NuMicro® Family

Arm® Cortex®-M4-based Microcontroller

M463/M467 Series
Technical Reference Manual

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro® microcontroller and microprocessor based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

TABLE OF CONTENTS

1 GENERAL DESCRIPTION 36
2 FEATURES 38
2.1 M463/M467 Series Features 38
3 PARTS INFORMATION 52
3.1 Package Type 52
3.1.1 M463/M467 Series 52
3.2 M463/M467 Series Naming Rule 53
3.3 M463/M467 Series Selection Guide 54
3.3.1 M463 Series Selection Guide 54
3.3.2 M467 Series Selection Guide 56
3.4 M463/M467 Series Features Comparison Table 59
4 PIN CONFIGURATION 60
4.1 Pin Configuration 60
4.1.1 M463 Series Pin Diagram 61
4.1.2 M467 Series Pin Diagram 68
4.1.3 M463 Series Multi-function Pin Diagram 76
4.1.4 M467 Series Multi-function Pin Diagram 102
4.2 Pin Mapping Table 140
4.2.1 M463 Series Mapping Table 140
4.2.2 M467 Series Mapping Table 144
4.3 Pin Functional Description 150
4.3.1 M463/M467 Series Summary Function Pin Description 150
5 BLOCK DIAGRAM 165
5.1 M463/M467 Series Block Diagram 165
6 FUNCTIONAL DESCRIPTION 166
6.1 ARM Cortex-M4 Core 166
6.2 System Manager 169
6.2.1 Overview 169
6.2.2 System Reset 169
6.2.3 System Power Distribution 175
6.2.4 Power Modes and Wake-up Sources 176
6.2.5 Power Modes and Power Level Transition 182
6.2.6 System Memory Map 183
6.2.7 SRAM Memory Organization 186
6.2.8 Bus Matrix 189
6.2.9 HIRC Auto Trim 189
6.2.10 Register Lock Control 190

6.2.11 Register Map 193
6.2.12 Register Description 196
6.2.13 System Timer (SysTick) 321
6.2.14 System Timer Control Register Map 322
6.2.15 Nested Vectored Interrupt Controller (NVIC) 326
6.2.16 System Control Register 360
6.3 Clock Controller 369
6.3.1 Overview 369
6.3.2 Clock Generator 372
6.3.3 System Clock and SysTick Clock 375
6.3.4 Peripherals Clock 376
6.3.5 Power-down Mode Clock 377
6.3.6 Clock Output 377
6.3.7 USB Clock Source 378
6.3.8 Register Map 379
6.3.9 Register Description 381
6.4 True Random Number Generator (TRNG) 448
6.4.1 Overview 448
6.4.2 Features 448
6.4.3 Block Diagram 448
6.4.4 Basic Configuration 448
6.4.5 Functional Description 448
6.4.6 Register Map 450
6.4.7 Register Description 451
6.5 Flash Memory Controller (FMC) 455
6.5.1 Overview 455
6.5.2 Features 455
6.5.3 Block Diagram 455
6.5.4 Functional Description 458
6.5.5 Register Map 497
6.5.6 Register Description 498
6.6 External Bus Interface (EBI) 522
6.6.1 Overview 522
6.6.2 Features 522
6.6.3 Block Diagram 523
6.6.4 Basic Configuration 523
6.6.5 Functional Description 523
6.6.6 Register Map 533
6.6.7 Register Description 534
6.7 HyperBus Interface Controller (HBI) 538
6.7.1 Overview 538

6.7.2 Features 538
6.7.3 Block Diagram 538
6.7.4 Basic Configuration 538
6.7.5 Functional Description 539
6.7.6 Register Map 546
6.7.7 Register Description 547
6.8 General Purpose I/O (GPIO) 555
6.8.1 Overview 555
6.8.2 Features 555
6.8.3 Block Diagram 556
6.8.4 Basic Configuration 557
6.8.5 Functional Description 557
6.8.6 Register Map 561
6.8.7 Register Description 567
6.9 PDMA Controller (PDMA) 589
6.9.1 Overview 589
6.9.2 Features 589
6.9.3 Block Diagram 589
6.9.4 Basic Configuration 589
6.9.5 Functional Description 590
6.9.6 Register Map 599
6.9.7 Register Description 601
6.10 Timer Controller (TMR) 651
6.10.1 Overview 651
6.10.2 Features 651
6.10.3 Block Diagram 652
6.10.4 Basic Configuration 657
6.10.5 Timer Functional Description 657
6.10.6 PWM Functional Description 663
6.10.7 Register Map 682
6.10.8 Register Description 688
6.11 Basic PWM Generator and Capture Timer (BPWM) 738
6.11.1 Overview 738
6.11.2 Features 738
6.11.3 Block Diagram 739
6.11.4 Basic Configuration 741
6.11.5 Functional Description 742
6.11.6 Register Map 758
6.11.7 Register Description 761
6.12 EPWM Generator and Capture Timer (EPWM) 794
6.12.1 Overview 794

6.12.2 Features 794
6.12.3 Block Diagram 795
6.12.4 Basic Configuration 799
6.12.5 Functional Description 799
6.12.6 Register Map 837
6.12.7 Register Description 845
6.13 Enhanced Quadrature Encoder Interface (EQEI) 928
6.13.1 Overview 928
6.13.2 Features 928
6.13.3 Block Diagram 928
6.13.4 Basic Configuration 930
6.13.5 Functional Description 930
6.13.6 Register Map 945
6.13.7 Register Description 946
6.14 Watchdog Timer (WDT) 960
6.14.1 Overview 960
6.14.2 Features 960
6.14.3 Block Diagram 960
6.14.4 Basic Configuration 961
6.14.5 Functional Description 961
6.14.6 Register Map 964
6.14.7 Register Description 965
6.15 Window Watchdog Timer (WWDT) 969
6.15.1 Overview 969
6.15.2 Features 969
6.15.3 Block Diagram 969
6.15.4 Basic Configuration 970
6.15.5 Functional Description 970
6.15.6 Register Map 974
6.15.7 Register Description 975
6.16 Real Time Clock (RTC) 980
6.16.1 Overview 980
6.16.2 Features 980
6.16.3 Block Diagram 981
6.16.4 Basic Configuration 982
6.16.5 Functional Description 982
6.16.6 Register Map 990
6.16.7 Register Description 992
6.17 Camera Capture Interface Controller (CCAP) 1028
6.17.1 Overview 1028
6.17.2 Features 1028

6.17.3 Block Diagram 1028
6.17.4 Basic Configuration 1028
6.17.5 Functional Description 1029
6.17.6 Register Map 1033
6.17.7 Register Description 1034
6.18 Enhanced Input Capture Timer (ECAP) 1050
6.18.1 Overview 1050
6.18.2 Features 1050
6.18.3 Block Diagram 1050
6.18.4 Basic Configuration 1051
6.18.5 Functional Description 1052
6.18.6 Register Map 1057
6.18.7 Register Description 1058
6.19 USCI - Universal Serial Control Interface Controller (USCI) 1069
6.19.1 Overview 1069
6.19.2 Features 1069
6.19.3 Block Diagram 1069
6.19.4 Functional Description 1069
6.20 USCI – UART Mode 1080
6.20.1 Overview 1080
6.20.2 Features 1080
6.20.3 Block Diagram 1080
6.20.4 Basic Configuration 1081
6.20.5 Functional Description 1081
6.20.6 Register Map 1090
6.20.7 Register Description 1091
6.21 USCI - SPI Mode 1113
6.21.1 Overview 1113
6.21.2 Features 1113
6.21.3 Block Diagram 1114
6.21.4 Basic Configuration 1114
6.21.5 Functional Description 1114
6.21.6 Register Map 1127
6.21.7 Register Description 1128
6.22 USCI - I²C Mode 1150
6.22.1 Overview 1150
6.22.2 Features 1150
6.22.3 Block Diagram 1150
6.22.4 Basic Configuration 1151
6.22.5 Functional Description 1151
6.22.6 Register Map 1170

6.22.7 Register Description 1171
6.23 UART Interface Controller (UART) 1189
6.23.1 Overview 1189
6.23.2 Features 1189
6.23.3 Block Diagram 1190
6.23.4 Basic Configuration 1193
6.23.5 Functional Description 1195
6.23.6 Register Map 1220
6.23.7 Register Description 1222
6.24 Ethernet MAC Controller (EMAC) 1258
6.24.1 Overview 1258
6.24.2 Features 1258
6.24.3 Architecture 1259
6.24.4 Basic Configuration 1271
6.24.5 Functional Description 1271
6.24.6 Descriptors 1296
6.24.7 Register Map 1320
6.24.8 Register Description 1323
6.25 Smart Card Host Interface (SC) 1400
6.25.1 Overview 1400
6.25.2 Features 1400
6.25.3 Block Diagram 1400
6.25.4 Basic Configuration 1402
6.25.5 Functional Description 1403
6.25.6 Register Map 1415
6.25.7 Register Description 1416
6.26 I²S Controller (I²S) 1441
6.26.1 Overview 1441
6.26.2 Features 1441
6.26.3 Block Diagram 1441
6.26.4 Basic Configuration 1441
6.26.5 Functional Description 1442
6.26.6 Register Map 1453
6.26.7 Register Description 1454
6.27 Serial Peripheral Interface (SPI) 1472
6.27.1 Overview 1472
6.27.2 Features 1472
6.27.3 Block Diagram 1473
6.27.4 Basic Configuration 1473
6.27.5 Functional Description 1474
6.27.6 Timing Diagram 1492

6.27.7 Programming Examples 1494
6.27.8 Register Map 1497
6.27.9 Register Description 1498
6.28 Quad Serial Peripheral Interface (QSPI) 1521
6.28.1 Overview 1521
6.28.2 Features 1521
6.28.3 Block Diagram 1522
6.28.4 Basic Configuration 1522
6.28.5 Functional Description 1523
6.28.6 Timing Diagram 1540
6.28.7 Programming Examples 1542
6.28.8 Register Map 1545
6.28.9 Register Description 1546
6.29 SPI Synchronous Serial Interface Controller (SPI Master Mode) 1561
6.29.1 Overview 1561
6.29.2 Features 1561
6.29.3 Block Diagram 1562
6.29.4 Basic Configuration 1562
6.29.5 Functional Description 1562
6.29.6 Mapping Address for SPI Flash Controller in AMBA System 1573
6.29.7 Register Map 1574
6.29.8 Register Description 1575
6.30 I²C Serial Interface Controller (I²C) 1593
6.30.1 Overview 1593
6.30.2 Features 1593
6.30.3 Block Diagram 1594
6.30.4 Basic Configuration 1594
6.30.5 Functional Description 1595
6.30.6 Register Map 1621
6.30.7 Register Description 1621
6.31 Secure Digital Host Controller (SDH) 1646
6.31.1 Overview 1646
6.31.2 Features 1646
6.31.3 Block Diagram 1647
6.31.4 Basic Configuration 1647
6.31.5 Functional Description 1648
6.31.6 Register Map 1650
6.31.7 Register Description 1651
6.32 Programmable Serial I/O (PSIO) 1671
6.32.1 Overview 1671
6.32.2 Features 1671

6.32.3 Block Diagram 1672
6.32.4 Basic Configuration 1673
6.32.5 Data Rate Limitation 1673
6.32.6 Functional Description 1675
6.32.7 Programing Flow 1686
6.32.8 Register Map 1706
6.32.9 Register Description 1707
6.33 USB 1.1 Device Controller (USBD) 1743
6.33.1 Overview 1743
6.33.2 Features 1743
6.33.3 Block Diagram 1744
6.33.4 Basic Configuration 1744
6.33.5 Functional Description 1744
6.33.6 Register Map 1752
6.33.7 Register Description 1756
6.34 High Speed USB 2.0 Device Controller (HSUSBD) 1788
6.34.1 Overview 1788
6.34.2 Features 1788
6.34.3 Block Diagram 1788
6.34.4 Basic Configuration 1788
6.34.5 Functional Description 1789
6.34.6 Registers Map 1792
6.34.7 Register Description 1797
6.35 High Speed USB 2.0 and USB 1.1 Host Controller (HSUSB and USBH) 1848
6.35.1 Overview 1848
6.35.2 Features 1848
6.35.3 Block Diagram 1849
6.35.4 Basic Configuration 1850
6.35.5 Functional Description 1850
6.35.6 Register Map 1852
6.35.7 Register Description 1854
6.36 USB 1.1 On-The-Go (OTG) 1905
6.36.1 Overview 1905
6.36.2 Features 1905
6.36.3 Block Diagram 1905
6.36.4 Basic Configuration 1906
6.36.5 Functional Description 1906
6.36.6 Register Map 1911
6.36.7 Register Description 1912
6.37 High Speed USB 2.0 On-The-Go (HSOTG) 1921
6.37.1 Overview 1921

6.37.2 Features 1921
6.37.3 Block Diagram 1921
6.37.4 Basic Configuration 1922
6.37.5 Functional Description 1922
6.37.6 Register Map 1927
6.37.7 Register Description 1928
6.38 Controller Area Network with Flexible Data-Rate (CAN FD) 1938
6.38.1 Overview 1938
6.38.2 Features 1938
6.38.3 Block Diagram 1938
6.38.4 Basic Configuration 1940
6.38.5 Functional Description 1941
6.38.6 Register Map 1969
6.38.7 Register Description 1971
6.39 CRC Controller (CRC) 2023
6.39.1 Overview 2023
6.39.2 Features 2023
6.39.3 Block Diagram 2023
6.39.4 Basic Configuration 2023
6.39.5 Functional Description 2024
6.39.6 Register Map 2026
6.39.7 Register Description 2027
6.40 Cryptographic Accelerator (CRYPTO) 2033
6.40.1 Overview 2033
6.40.2 Features 2033
6.40.3 Block Diagram 2034
6.40.4 Basic Configuration 2035
6.40.5 Functional Description 2035
6.40.6 Register Map 2075
6.40.7 Register Description 2089
6.41 Cryptographic Accelerator for M463 Series (CRYPTO_M463) 2187
6.41.1 Overview 2187
6.41.2 Features 2187
6.41.3 Block Diagram 2187
6.41.4 Basic Configuration 2188
6.41.5 Functional Description 2188
6.41.6 Register Map 2199
6.41.7 Register Description 2201
6.42 Key Store (KS) 2224
6.42.1 Overview 2224
6.42.2 Features 2224

6.42.3 Block Diagram 2225
6.42.4 Basic Configuration 2225
6.42.5 Functional Description 2225
6.42.6 Register Map 2233
6.42.7 Register Description 2234
6.43 Keypad Interface (KPI) 2243
6.43.1 Overview 2243
6.43.2 Features 2243
6.43.3 Block Diagram 2243
6.43.4 Basic Configuration 2243
6.43.5 Functional Description 2243
6.43.6 Register Map 2246
6.43.7 Register Description 2247
6.44 Enhanced 12-bit Analog-to-Digital Converter (EADC) 2258
6.44.1 Overview 2258
6.44.2 Features 2258
6.44.3 Block Diagram 2259
6.44.4 Basic Configuration 2259
6.44.5 Functional Description 2260
6.44.6 Register Map 2275
6.44.7 Register Description 2280
6.45 Digital to Analog Converter (DAC) 2315
6.45.1 Overview 2315
6.45.2 Features 2315
6.45.3 Block Diagram 2316
6.45.4 Basic Configuration 2316
6.45.5 Functional Description 2317
6.45.6 Register Map 2322
6.45.7 Register Description 2323
6.46 Analog Comparator Controller (ACMP) 2338
6.46.1 Overview 2338
6.46.2 Features 2338
6.46.3 Block Diagram 2340
6.46.4 Basic Configuration 2340
6.46.5 Functional Description 2341
6.46.6 Register Map 2347
6.46.7 Register Description 2348
6.47 Peripherals Interconnection 2360
6.47.1 Overview 2360
6.47.2 Peripherals Interconnect Matrix Table 2360
6.47.3 Functional Description 2360

7 APPLICATION CIRCUIT 2363
7.1 Power Supply Scheme 2363
7.2 Peripheral Application Scheme 2364
8 ELECTRICAL CHARACTERISTICS 2365
9 PACKAGE DIMENSIONS 2366
9.1 QFN48L 5x5mm, Thickness 0.80mm, Lead_Pitch 0.35mm, Lead_Length 0.40mm, EP Size 3.75x3.75mm 2366
9.2 LQFP48L 7x7mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm 2367
9.3 LQFP64L 7x7mm, Thickness 1.40mm, Lead_Pitch 0.40mm, Lead_Length 1.00mm 2368
9.4 LQFP128L 14x14mm, Thickness 1.40mm, Lead_Pitch 0.40mm, Lead_Length 1.00mm 2369
9.5 LQFP144L 20x20mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm 2370
9.6 LQFP176L 24x24mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm 2371
9.7 LQFP48L EPAD 7x7mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm, EP_SIZE 5.36x5.36mm 2372
9.8 LQFP64L EPAD 7x7mm, Thickness 1.40mm, Lead_Pitch 0.40mm, Lead_Length 1.00mm, EP_SIZE 5.48x5.48mm 2373
9.9 LQFP128L EPAD 14x14mm, Thickness 1.40mm, Lead_Pitch 0.40mm, Lead_Length 1.00mm, EP_SIZE 7.15x7.15mm 2374
9.10 LQFP144L EPAD 20x20mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm, EP_SIZE 5.9x5.9mm 2375
9.11 LQFP176L EPAD 24x24mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm, EP_SIZE 6.75x6.75mm 2376
10 ABBREVIATIONS 2377
10.1 Abbreviations 2377
11 REVISION HISTORY 2379

LIST OF FIGURES

Figure 4.1-1 M463 Series QFN48-Pin Diagram 61
Figure 4.1-2 M463 Series LQFP48-Pin Diagram 62
Figure 4.1-3 M463 Series LQFP64-Pin Diagram 63
Figure 4.1-4 M463 Series LQFP128-Pin Diagram 64
Figure 4.1-5 M463 Series LQFP48 EPAD Pin Diagram 65
Figure 4.1-6 M463 Series LQFP64 EPAD Pin Diagram 66
Figure 4.1-7 M463 Series LQFP128 EPAD Pin Diagram 67
Figure 4.1-8 M467 Series LQFP64-Pin Diagram 68
Figure 4.1-9 M467 Series LQFP128-Pin Diagram 69
Figure 4.1-10 M467 Series LQFP144-Pin Diagram 70
Figure 4.1-11 M467 Series LQFP176-Pin Diagram 71
Figure 4.1-12 M467 Series LQFP64 EPAD Pin Diagram 72
Figure 4.1-13 M467 Series LQFP128 EPAD Pin Diagram 73
Figure 4.1-14 M467 Series LQFP144 EPAD Pin Diagram 74
Figure 4.1-15 M467 Series LQFP176 EPAD Pin Diagram 75
Figure 4.1-16 M463YGCAE Multi-function Pin Diagram 76
Figure 4.1-17 M463LGCAE Multi-function Pin Diagram 79
Figure 4.1-18 M463SGCAE Multi-function Pin Diagram 82
Figure 4.1-19 M463KGCAE Multi-function Pin Diagram 86
Figure 4.1-20 M463L2GCAC Multi-function Pin Diagram 92
Figure 4.1-21 M463S2GCAC Multi-function Pin Diagram 95
Figure 4.1-22 M463K2GCAC Multi-function Pin Diagram 98
Figure 4.1-23 M467SJHAN Multi-function Pin Diagram 102
Figure 4.1-24 M467KJHAN Multi-function Pin Diagram 105
Figure 4.1-25 M467JJHAN Multi-function Pin Diagram 110
Figure 4.1-26 M467HJHAN Multi-function Pin Diagram 115
Figure 4.1-27 M467S2JHAE Multi-function Pin Diagram 121
Figure 4.1-28 M467K2JHAE Multi-function Pin Diagram 124
Figure 4.1-29 M467J2JHAE Multi-function Pin Diagram 129
Figure 4.1-30 M467H2JHAE Multi-function Pin Diagram 134
Figure 5.1-1 M463/M467 Series Block Diagram 165
Figure 6.1-1 Cortex-M4 Block Diagram 166
Figure 6.2-1 System Reset Sources 170
Figure 6.2-2 Reset Event State Machine 170
Figure 6.2-3 nRESET Reset Waveform(Except SPDx and DPDx mode) 172
Figure 6.2-4 Power-on Reset (POR) Waveform 173

Figure 6.2-5 Low Voltage Reset (LVR) Waveform 173
Figure 6.2-6 Brown-out Detector (BOD) Waveform 174
Figure 6.2-7 NuMicro M463/M467 series Power Distribution Diagram 176
Figure 6.2-8 Power Mode State Machine 178
Figure 6.2-9 NuMicro M463/M467 Series Power Distribution Diagram 182
Figure 6.2-10 SRAM Block Diagram 186
Figure 6.2-11 SRAM Memory Organization for 512 Kbyte Device 187
Figure 6.2-12 SRAM Memory Organization for 128 Kbyte Device 188
Figure 6.2-13 NuMicro M463/M467 Series Bus Matrix Diagram 189
Figure 6.3-1 Clock Generator Global View Diagram (1/3) 370
Figure 6.3-2 Clock Generator Global View Diagram (2/3) 371
Figure 6.3-3 Clock Generator Global View Diagram (3/3) 372
Figure 6.3-4 Clock Generator Block Diagram 374
Figure 6.3-5 System Clock Block Diagrams 375
Figure 6.3-6 HXT Stop Protect Procedure 376
Figure 6.3-7 SysTick Clock Control Block Diagram 376
Figure 6.3-8 Clock Output Block Diagram 377
Figure 6.3-9 USB Clock Source 378
Figure 6.4-1 True Random Number Generator Block Diagram 448
Figure 6.5-1 Flash Memory Controller Block Diagram 456
Figure 6.5-2 Data Flash Shared with APROM Examples (256/512/1024 Kbytes) 459
Figure 6.5-3 Address Operation Model 460
Figure 6.5-4 New XOM Setting Flow 467
Figure 6.5-5 OTP Memory Map 480
Figure 6.5-6 Flash Memory Map 481
Figure 6.5-7 System Memory Map with IAP Mode 482
Figure 6.5-8 LDROM with IAP Mode 483
Figure 6.5-9 APROM with IAP Mode 483
Figure 6.5-10 System Memory Map without IAP Mode 484
Figure 6.5-11 ISP Procedure Example 487
Figure 6.5-12 ISP 32-bit Programming Procedure 488
Figure 6.5-13 ISP 64-bit Programming Procedure 489
Figure 6.5-14 Multi-word Programming Time 489
Figure 6.5-15 Firmware in SRAM for Multi-word Programming 490
Figure 6.5-16 Firmware in Boot Loader for Multi-word Programming 490
Figure 6.5-17 Multi-word Programming Flow 492
Figure 6.5-18 Fast Flash Programming Verification Flow 493

Figure 6.5-19 Verification Flow 494
Figure 6.5-20 Flash CRC32 Checksum Calculation 494
Figure 6.6-1 EBI Block Diagram 523
Figure 6.6-2 Connection of 16-bit EBI Data Width with 16-bit Device 524
Figure 6.6-3 Connection of 8-bit EBI Data Width with 8-bit Device 525
Figure 6.6-4 Connection of 16-bit EBI Data Width with 16-bit Device in Separate mode 525
Figure 6.6-5 Connection of 8-bit EBI Data Width with 8-bit Device in Separate mode 526
Figure 6.6-6 Timing Control Waveform for 16-bit Data Width 527
Figure 6.6-7 Timing Control Waveform for 8-bit Data Width 528
Figure 6.6-8 Timing Control Waveform for Byte Write in 16-bit Data Mode 529
Figure 6.6-9 Timing Control Waveform for Insert Idle Cycle 530
Figure 6.6-10 Timing Control Waveform for 16-bit Data Width for Separate Mode 531
Figure 6.6-11 Timing Control Waveform for Continuous Data Access Mode 532
Figure 6.7-1 HyperBus Interface Controller Block Diagram 538
Figure 6.7-2 HyperRAM Read Transaction, Single Initial Latency Count 540
Figure 6.7-3 Read Transaction, Additional Latency Count 540
Figure 6.7-4 Write Transaction, Single Initial Latency Count 541
Figure 6.7-5 Data Placement during a Read Transaction 541
Figure 6.7-6 Data Placement during a Write Transaction 542
Figure 6.7-7 Chip Reset Command and Timing 543
Figure 6.7-8 Exit Timing for HS and DPD 543
Figure 6.7-9 Chip Select Setup Time to Next HBI_CK Rising Edge 544
Figure 6.7-10 HyperBus Initial Access Time 544
Figure 6.7-11 Chip Select Hold Time to After Last HBI_CK Falling Edge 544
Figure 6.7-12 Data Order for Write 545
Figure 6.7-13 Data Order for Read 545
Figure 6.8-1 GPIO Controller Block Diagram 556
Figure 6.8-2 Push-Pull Output 558
Figure 6.8-3 Open-Drain Output 558
Figure 6.8-4 Quasi-Bidirectional I/O Mode 559
Figure 6.8-5 GPIO Rising Edge Trigger Interrupt 560
Figure 6.8-6 GPIO Falling Edge Trigger Interrupt 560
Figure 6.9-1 PDMA Controller Block Diagram 589
Figure 6.9-2 Descriptor Table Entry Structure 590
Figure 6.9-3 Basic Mode Finite State Machine 592
Figure 6.9-4 Descriptor Table Link List Structure 593
Figure 6.9-5 Scatter-gather Mode Finite State Machine 593

Figure 6.9-6 Example of Single Transfer Type and Burst Transfer Type in Basic Mode 595
Figure 6.9-7 Example of PDMA Channel 0 Time-out Counter Operation 596
Figure 6.9-8 Stride Function Block Transfer 597
Figure 6.9-9 Block Transfer with Interval =3 and Repeat Count =2 598
Figure 6.10-1 Timer Controller Block Diagram 653
Figure 6.10-2 Clock Source of Timer Controller 654
Figure 6.10-3 PWM Generator Overview Block Diagram 655
Figure 6.10-4 PWM System Clock Source Control 655
Figure 6.10-5 PWM Counter Clock Source Control 656
Figure 6.10-6 PWM Independent Mode Architecture Diagram 656
Figure 6.10-7 PWM Complementary Mode Architecture Diagram 657
Figure 6.10-8 Continuous Counting Mode 659
Figure 6.10-9 Capture Mode on TMx_EXT Pin 660
Figure 6.10-10 Reset Counter Mode 661
Figure 6.10-11 Internal Timer Trigger 662
Figure 6.10-12 Inter-Timer Trigger Capture Timing 663
Figure 6.10-13 PWM Prescale Waveform in Up Count Type 663
Figure 6.10-14 PWM Up Count Type 664
Figure 6.10-15 PWM Down Count Type 664
Figure 6.10-16 PWM Up-Down Count Type 665
Figure 6.10-17 PWM Comparator Events in Up-Down Count Type 666
Figure 6.10-18 Period Loading Mode with Up Count Type 667
Figure 6.10-19 Immediately Loading Mode with Up Count Type 668
Figure 6.10-20 PWM Pulse Generation in Up-Down Count Type 668
Figure 6.10-21 PWM Pulse Generation in Up Count Type 669
Figure 6.10-22 PWM Pulse Generation in Down Count Type 669
Figure 6.10-23 PWM 0% to 100% Duty Cycle in Up Count Type and Up-Down Count Type 670
Figure 6.10-24 PWM Independent Mode Output Waveform 670
Figure 6.10-25 PWM Complementary Mode Output Waveform 671
Figure 6.10-26 PWMx_CH0 Output Control in Independent Mode 671
Figure 6.10-27 PWMx_CH0 and PWMx_CH1 Output Control in Complementary Mode 671
Figure 6.10-28 Dead-Time Insertion 672
Figure 6.10-29 PWM Output Mask Control Waveform 672
Figure 6.10-30 Brake Pin Noise Filter Block Diagram 673
Figure 6.10-31 Brake Event Block Diagram for PWMx_CH0 and PWMx_CH1 674
Figure 6.10-32 Edge Detector Brake Waveform for PWMx_CH0 and PWMx_CH1 675
Figure 6.10-33 Level Detector Brake Waveform for PWMx_CH0 and PWMx_CH1 676

Figure 6.10-34 Brake Source Block Diagram 677
Figure 6.10-35 System Fail Brake Block Diagram 678
Figure 6.10-36 PWMx_CH0 and PWMx_CH1 Polarity Control with Dead-Time Insertion 678
Figure 6.10-37 PWM Accumulate Interrupt Waveform 679
Figure 6.10-38 PWM Interrupt Architecture Diagram 679
Figure 6.10-39 PWM Accumulator Stop Mode Waveform 680
Figure 6.10-40 PWM Trigger EADC Block Diagram 681
Figure 6.11-1 BPWM Generator Overview Block Diagram 739
Figure 6.11-2 BPWM Clock Source Control 740
Figure 6.11-3 BPWM Clock Source Control 740
Figure 6.11-4 BPWM Independent Mode Architecture Diagram 741
Figure 6.11-5 BPWM_CH0 CLKPSC Waveform 742
Figure 6.11-6 BPWM Counter Clear Waveform 743
Figure 6.11-7 BPWM Up Counter Type 743
Figure 6.11-8 BPWM Down Counter Type 744
Figure 6.11-9 BPWM Up-Down Counter Type 744
Figure 6.11-10 BPWM CMPDAT Events in Up-Down Counter Type 745
Figure 6.11-11 Period Loading Mode with Up-Counter Type 746
Figure 6.11-12 Immediately Loading Mode with Up-Counter Type 747
Figure 6.11-13 Center Loading Mode with Up-Down-Counter Type 748
Figure 6.11-14 BPWM Pulse Generation (Left: Asymmetric Pulse, Right: Variety Pulse) 749
Figure 6.11-15 BPWM 0% to 100% Pulse Generation (Left: Up Counter Type, Right: Up-down Counter Type) 749
Figure 6.11-16 BPWM_CH0 Output Control 3 Steps 750
Figure 6.11-17 Mask Control Waveform Illustration 751
Figure 6.11-18 Initial State and Polarity Control 751
Figure 6.11-19 BPWM_CH0 and BPWM_CH1 Pair Interrupt Architecture Diagram 752
Figure 6.11-20 BPWM_CH0 and BPWM_CH1 Pair Trigger EADC0/1/2 Source Block Diagram 753
Figure 6.11-21 BPWM CH0~ CH5 Trigger EADC0/1/2 Block Diagram 753
Figure 6.11-22 BPWM Trigger EADC0/1/2 in Up-Down Counter Type Timing Waveform 754
Figure 6.11-23 BPWM_CH0 and BPWM_CH1 Pair Trigger EADC Source Block Diagram 754
Figure 6.11-24 BPWM CH0~ CH5 Trigger EADC Block Diagram 755
Figure 6.11-25 BPWM Trigger EADC in Up-Down Counter Type Timing Waveform 755
Figure 6.11-26 BPWM_CH0 Capture Block Diagram 756
Figure 6.11-27 Capture Operation Waveform 757
Figure 6.12-1 EPWM Generator Overview Block Diagram 796
Figure 6.12-2 EPWM Clock Source Control 796
Figure 6.12-3 EPWM Clock Source Control 797

Figure 6.12-4 EPWM Independent Mode Architecture Diagram 798
Figure 6.12-5 EPWM Complementary Mode Architecture Diagram 799
Figure 6.12-6 EPWM_CH0 Prescaler Waveform in Up Counter Type 800
Figure 6.12-7 EPWM Counter Waveform when Setting Clear Counter 800
Figure 6.12-8 EPWM Up Counter Type 801
Figure 6.12-9 EPWM Down Counter Type 801
Figure 6.12-10 EPWM Up-Down Counter Type 802
Figure 6.12-11 EPWM Compared point Events in Up-Down Counter Type 803
Figure 6.12-12 EPWM Double Buffering Illustration 804
Figure 6.12-13 Period Loading in Up-Count Mode 805
Figure 6.12-14 Immediately Loading in Up-Count Mode 806
Figure 6.12-15 Window Loading in Up-Count Mode 807
Figure 6.12-16 Center Loading in Up-Down-Count Mode 808
Figure 6.12-17 EPWM One-shot Mode Output Waveform 809
Figure 6.12-18 EPWM Pulse Generation 810
Figure 6.12-19 Software Event Output Force 810
Figure 6.12-20 EPWM 0% to 100% Pulse Generation 811
Figure 6.12-21 EPWM Independent Mode Waveform 812
Figure 6.12-22 EPWM Complementary Mode Waveform 812
Figure 6.12-23 EPWM Group Function Waveform 813
Figure 6.12-24 EPWM SYNC_IN Noise Filter Block Diagram 814
Figure 6.12-25 EPWM Counter Synchronous Function Block Diagram 815
Figure 6.12-26 EPWM Synchronous Function with Synchronize source from SYNC_IN Signal 815
Figure 6.12-27 EPWMx_CH0 Output Control in Independent Mode 816
Figure 6.12-28 EPWMx_CH0 and EPWMx_CH1 Output Control in Complementary Mode 816
Figure 6.12-29 Dead-Time Insertion 817
Figure 6.12-30 Dead-Time Independent Control 817
Figure 6.12-31 Illustration of Mask Control Waveform 818
Figure 6.12-32 Brake Noise Filter Block Diagram 819
Figure 6.12-33 Brake Block Diagram for EPWMx_CH0 and EPWMx_CH1 Pair 820
Figure 6.12-34 Edge Detector Waveform for EPWMx_CH0 and EPWMx_CH1 Pair 821
Figure 6.12-35 Level Detector Waveform for EPWMx_CH0 and EPWMx_CH1 Pair 821
Figure 6.12-36 Brake Source Block Diagram 822
Figure 6.12-37 Brake System Fail Block Diagram 822
Figure 6.12-38 EPWM LEB Function Waveform 823
Figure 6.12-39 Initial State and Polarity Control with Rising Edge Dead-Time Insertion 824
Figure 6.12-40 EPWMx_CH0 Accumulate Interrupt Waveform 825

Figure 6.12-41 EPWMx_CH0 and EPWMx_CH1 Pair Interrupt Architecture Diagram 826
Figure 6.12-42 Fault Detect Function Interrupt Architecture Diagram 827
Figure 6.12-43 EPWMx_CH0 and EPWMx_CH1 Pair Trigger EADC Events 828
Figure 6.12-44 EPWMx_CH0 Trigger EADC Block Diagram 828
Figure 6.12-45 EPWM Trigger EADC in Up-Down Counter Type Timing Waveform 829
Figure 6.12-46 EPWM_CH0 and EPWM_CH1 Pair Trigger DAC Block Diagram 829
Figure 6.12-47 EPWM_CH0 Capture Block Diagram 830
Figure 6.12-48 Capture Operation Waveform 831
Figure 6.12-49 Capture PDMA Operation Waveform of Channel 0 832
Figure 6.12-50 Accumulator PDMA Function Architecture 833
Figure 6.12-51 EPWM Accumulator Stop Mode Waveform 834
Figure 6.12-52 Fault Detect Function Architecture 835
Figure 6.12-53 EPWM Mask Function Waveform 836
Figure 6.12-54 EPWM Deglitch Function Waveform 836
Figure 6.13-1 EQEI Block Diagram 929
Figure 6.13-2 EQEI Decoder Unit 929
Figure 6.13-3 EQEI Clock Source Control 930
Figure 6.13-4 Noise Filter 931
Figure 6.13-5 Noise Filter Sampling Clock Selection 931
Figure 6.13-6 QEA/QEB/IDX Timing Requirement Through Noise Filter 932
Figure 6.13-7 X4 Counting Mode 933
Figure 6.13-8 X2 Counting Mode 934
Figure 6.13-9 Compare Operation 935
Figure 6.13-10 Phase Counting Mode, Mode1 936
Figure 6.13-11 Phase Counting Mode, Mode2 936
Figure 6.13-12 X1 Phase Counting Mode, Mode1, Rising Trigger 936
Figure 6.13-13 X1 Phase Counting Mode, Mode1, Falling Trigger 937
Figure 6.13-14 X2 Phase Counting Mode, Mode1 937
Figure 6.13-15 X1 Phase Counting Mode, Mode2, Rising Trigger 937
Figure 6.13-16 X1 Phase Counting Mode, Mode2, Falling Trigger 938
Figure 6.13-17 X2 Phase Counting Mode, Mode2 938
Figure 6.13-18 X2 Direction Count Mode 939
Figure 6.13-19 x2 Direction Up Count Mode 939
Figure 6.13-20 X2 Direction Down Count Mode 939
Figure 6.13-21 EQEI Swap Function for QA and QB 940
Figure 6.13-22 EQEI Unit Timer Counter Control Flow Chart 941
Figure 6.13-23 EQEI_CNT Reload/Reset Control 942

Figure 6.13-24 Trigger Control of Capturing EQEI Counter 942
Figure 6.13-25 Capture and Latch EQEI Counter 943
Figure 6.13-26 Enhanced Quadrature Encoder Interface Interrupt Architecture Diagram 944
Figure 6.13-27 EQEI Phase Variation for CHA and CHB 944
Figure 6.14-1 Watchdog Timer Clock Control 960
Figure 6.14-2 Watchdog Timer Block Diagram 961
Figure 6.14-3 Watchdog Timer Time-out Interval and Reset Period Timing 962
Figure 6.15-1 WWDT Clock Control 969
Figure 6.15-2 WWDT Block Diagram 969
Figure 6.15-3 WWDT Reset and Reload Behavior 971
Figure 6.15-4 WWDT Reload Counter When CNTDAT > CMPDAT 971
Figure 6.15-5 WWDT Reload Counter When WWDT_CNT < WINCMP 972
Figure 6.15-6 WWDT Interrupt and Reset Signals 972
Figure 6.16-1 RTC Block Diagram 981
Figure 6.16-2 Dynamic Rate Definition 987
Figure 6.16-3 Backup I/O Control Diagram 989
Figure 6.17-1 Camera Capture Interface Controller Functional Block Diagram 1028
Figure 6.18-1 Input Capture Timer/Counter Architecture 1050
Figure 6.18-2 Input Capture Timer/Counter Clock Source Control 1052
Figure 6.18-3 Noise Filter Sampling Clock Selection 1053
Figure 6.18-4 Input Capture Timer/Counter Function Block 1054
Figure 6.18-5 Input Capture Timer/Counter Interrupt Architecture Diagram 1056
Figure 6.19-1 USCI Block Diagram 1069
Figure 6.19-2 Input Conditioning for USCIx_DAT[1:0] and USCIx_CTL[1:0] 1070
Figure 6.19-3 Input Conditioning for USCIx_CLK 1071
Figure 6.19-4 Block Diagram of Data Buffering 1072
Figure 6.19-5 Data Access Structure 1073
Figure 6.19-6 Transmit Data Path 1073
Figure 6.19-7 Receive Data Path 1074
Figure 6.19-8 Protocol-Relative Clock Generator 1075
Figure 6.19-9 Basic Clock Divider Counter 1076
Figure 6.19-10 Block of Timing Measurement Counter 1076
Figure 6.19-11 Sample Time Counter 1077
Figure 6.19-12 Event and Interrupt Structure 1078
Figure 6.20-1 USCI-UART Mode Block Diagram 1080
Figure 6.20-2 UART Signal Connection for Full-Duplex Communication 1081
Figure 6.20-3 UART Standard Frame Format 1082

Figure 6.20-4 UART Bit Timing (Data Sample Time) 1084
Figure 6.20-5 UART Auto Baud Rate Control 1086
Figure 6.20-6 Incoming Data Wake-Up 1087
Figure 6.20-7 nCTS Wake-Up Case 1 1087
Figure 6.20-8 nCTS Wake-Up Case 2 1087
Figure 6.21-1 SPI Master Mode Application Block Diagram 1113
Figure 6.21-2 SPI Slave Mode Application Block Diagram 1113
Figure 6.21-3 USCI SPI Mode Block Diagram 1114
Figure 6.21-44-Wire Full-Duplex SPI Communication Signals (Master Mode) 1115
Figure 6.21-54-Wire Full-Duplex SPI Communication Signals (Slave Mode) 1115
Figure 6.21-6 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x0) 1116
Figure 6.21-7 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x1) 1117
Figure 6.21-8 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x2) 1117
Figure 6.21-9 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x3) 1117
Figure 6.21-1016-bit Data Length in One Word Transaction with MSB First Format 1118
Figure 6.21-11 Word Suspend Interval between Two Transaction Words 1118
Figure 6.21-12 Auto Slave Select (SUSPITV ≧ 0x3) 1119
Figure 6.21-13 Auto Slave Select (SUSPITV < 0x3) 1120
Figure 6.21-14 One Output Data Channel Half-duplex (SPI Master Mode) 1121
Figure 6.21-15 One Input Data Channel Half-duplex (SPI Master Mode) 1121
Figure 6.21-16 SPI Timing in Master Mode 1123
Figure 6.21-17 SPI Timing in Master Mode (Alternate Phase of Serial Bus Clock) 1123
Figure 6.21-18 SPI Timing in Slave Mode 1124
Figure 6.21-19 SPI Timing in Slave Mode (Alternate Phase of Serial Bus Clock) 1124
Figure 6.22-1 I²C Bus Timing 1150
Figure 6.22-2 USCI I²C Mode Block Diagram 1151
Figure 6.22-3 I²C Protocol 1152
Figure 6.22-4 START and STOP Conditions 1152
Figure 6.22-5 Bit Transfer on the I²C Bus 1153
Figure 6.22-6 Acknowledge on the I²C Bus 1154
Figure 6.22-7 Arbitration Lost 1155
Figure 6.22-8 Control I²C Bus according to Current I²C Status 1157
Figure 6.22-9 Master Transmits Data to Slave with a 7-bit Address 1158
Figure 6.22-10 Master Reads Data from Slave with a 7-bit Address 1158
Figure 6.22-11 Master Transmits Data to Slave by 10-bit Address 1158
Figure 6.22-12 Master Reads Data from Slave by 10-bit Address 1158
Figure 6.22-13 Master Transmitter Mode Control Flow with 7-bit Address 1159

Figure 6.22-14 Master Receiver Mode Control Flow with 7-bit Address 1160
Figure 6.22-15 Master Transmitter Mode Control Flow with 10-bit Address 1161
Figure 6.22-16 Master Receiver Mode Control Flow with 10-bit Address 1162
Figure 6.22-17 Save Mode Control Flow with 7-bit Address 1163
Figure 6.22-18 Save Mode Control Flow with 10-bit Address 1164
Figure 6.22-19 GC Mode with 7-bit Address..... 1165
Figure 6.22-20 Setup Time Wrong Adjustment..... 1167
Figure 6.22-21 Hold Time Wrong Adjustment..... 1167
Figure 6.22-22 I²C Time-out Count Block Diagram 1168
Figure 6.22-23 EEPROM Random Read..... 1169
Figure 6.22-24 Protocol of EEPROM Random Read..... 1169
Figure 6.23-1 UART Clock Control Diagram..... 1191
Figure 6.23-2 UART Block Diagram..... 1192
Figure 6.23-3 Auto-Baud Rate Measurement 1198
Figure 6.23-4 Transmit Delay Time Operation..... 1199
Figure 6.23-5 UART nCTS Wake-up Case1..... 1200
Figure 6.23-6 UART nCTS Wake-up Case2..... 1200
Figure 6.23-7 UART Data Wake-up..... 1201
Figure 6.23-8 UART Received Data FIFO reached threshold wake-up 1201
Figure 6.23-9 UART RS-485 AAD Mode Address Match Wake-up..... 1202
Figure 6.23-10 UART Received Data FIFO threshold time-out wake-up 1202
Figure 6.23-11 Auto-Flow Control Block Diagram 1206
Figure 6.23-12 UART nCTS Auto-Flow Control Enabled..... 1206
Figure 6.23-13 UART nRTS Auto-Flow Control Enabled..... 1207
Figure 6.23-14 UART nRTS Auto-Flow with Software Control 1207
Figure 6.23-15 IrDA Control Block Diagram 1208
Figure 6.23-16 IrDA TX/RX Timing Diagram 1209
Figure 6.23-17 Structure of LIN Frame 1209
Figure 6.23-18 Structure of LIN Byte 1210
Figure 6.23-19 Break Detection in LIN Mode..... 1212
Figure 6.23-20 LIN Frame ID and Parity Format 1212
Figure 6.23-21 LIN Sync Field Measurement 1214
Figure 6.23-22 UART_BAUD Update Sequence in AR mode if SLVDUEN is 1 1215
Figure 6.23-23 UART_BAUD Update Sequence in AR mode if SLVDUEN is 0 1215
Figure 6.23-24 RS-485 nRTS Driving Level in Auto Direction Mode..... 1217
Figure 6.23-25 RS-485 nRTS Driving Level with Software Control 1218
Figure 6.23-26 Structure of RS-485 Frame 1219

Figure 6.24-1 EMAC Block Diagram 1259
Figure 6.24-2 Descriptor Ring and Chain Structure 1260
Figure 6.24-3 Tx DMA Operation by default Mode 1264
Figure 6.24-4 Tx DMA Operation in OSF Mode 1266
Figure 6.24-5 Receive DMA Operation 1269
Figure 6.24-6 Networked Time Synchronization 1272
Figure 6.24-7 System Time Update Using Fine Method 1274
Figure 6.24-8 Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction 1278
Figure 6.24-9 RMII Block Diagram 1294
Figure 6.24-10 Transmission Bit Ordering 1295
Figure 6.24-11 Receive Bit Ordering 1295
Figure 6.24-12 MAC Interrupt Masking Scheme 1296
Figure 6.24-13 Rx/Tx Descriptors in Same-Endian Mode for 64-Bit, Little-Endian Format Data Bus 1297
Figure 6.24-14 Transmit Descriptor Format in Little-Endian Mode With a 32-Bit Data Bus 1297
Figure 6.24-15 Receive Descriptor Format in Little-Endian Mode with a 32-Bit Data Bus 1301
Figure 6.24-16 Descriptor Fields when DMA Clears the Own Bit 1305
Figure 6.24-17 Descriptor Fields when DMA Clears the Own Bit 1306
Figure 6.24-18 Receive Timestamp Fields 1307
Figure 6.24-19 Transmit Descriptor Fields - Alternate (Enhanced) Format 1309
Figure 6.24-20 Transmit Descriptor Fetch (Read) for Alternate (Enhanced) Format 1309
Figure 6.24-21 Receive Descriptor Fields - Alternate (Enhanced) Format 1314
Figure 6.25-1 SC Clock Control Diagram (8-bit Pre-scale Counter in Clock Controller) 1401
Figure 6.25-2 SC Controller Block Diagram 1401
Figure 6.25-3 SC Data Character 1403
Figure 6.25-4 SC Activation Sequence 1404
Figure 6.25-5 SC Warm Reset Sequence 1406
Figure 6.25-6 SC Deactivation Sequence 1407
Figure 6.25-7 Basic Operation Flow 1408
Figure 6.25-8 Initial Character TS 1409
Figure 6.25-9 SC Error Signal 1410
Figure 6.25-10 Transmit Direction Block Guard Time Operation 1413
Figure 6.25-11 Receive Direction Block Guard Time Operation 1413
Figure 6.25-12 Extra Guard Time Operation 1413
Figure 6.26-1 I^2S Controller Block Diagram 1441
Figure 6.26-2 I^2S Clock Control Diagram 1442
Figure 6.26-3 Master Mode Interface Block Diagram 1443

Figure 6.26-4 Slave Mode Interface Block Diagram 1443
Figure 6.26-5 I²S Channel Width and Data Width (CHWIDTH≦DATWIDTH) 1443
Figure 6.26-6 I²S Channel Width and Data Width (CHWIDTH > DATWIDTH)..... 1444
Figure 6.26-7 I²S Data Format Timing Diagram (FORMAT = 0x0 ; CHWIDTH≦DATWIDTH) . 1444
Figure 6.26-8 MSB Justified Data Format (FORMAT = 0x1 ; CHWIDTH > DATWIDTH)..... 1444
Figure 6.26-9 LSB Justified Data Format (FORMAT = 0x2 ; CHWIDTH > DATWIDTH)..... 1445
Figure 6.26-10 Standard PCM Audio Timing Diagram (FORMAT = 0x4 ; CHWIDTH≦DATWIDTH) 1445
Figure 6.26-11 PCM with MSB Justified Data Format (FORMAT = 0x5 ; CHWIDTH > DATWIDTH) 1445
Figure 6.26-12 PCM with LSB Justified Data Format (FORMAT = 0x6 ; CHWIDTH > DATWIDTH) 1446
Figure 6.26-13 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM Standard Data Format; FORMAT=0x4) 1446
Figure 6.26-14 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM with MSB Justified; FORMAT=0x5)..... 1447
Figure 6.26-15 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM with LSB Justified; FORMAT=0x6)..... 1447
Figure 6.26-16 I²S Interrupts 1448
Figure 6.26-17 FIFO Contents for Various 2-channel Audio Modes..... 1449
Figure 6.26-18 FIFO Contents for Various 4-channel Audio Modes..... 1450
Figure 6.26-19 FIFO Contents for Various 6-channel Audio Modes (Part-1) 1451
Figure 6.26-20 FIFO Contents for Various 6-channel Audio Modes (Part-2) 1452
Figure 6.27-1 SPI Block Diagram..... 1473
Figure 6.27-2 SPI Peripheral Clock..... 1475
Figure 6.27-3 SPI Full-Duplex Master Mode Application Block Diagram 1475
Figure 6.27-4 SPI Full-Duplex Slave Mode Application Block Diagram 1476
Figure 6.27-5 32-bit in One Transaction 1477
Figure 6.27-6 Automatic Slave Selection (SSACTPOL = 0, SUSPITV > 0x2) 1478
Figure 6.27-7 Automatic Slave Selection (SSACTPOL = 0, SUSPITV < 0x3) 1478
Figure 6.27-8 Byte Reorder Function..... 1478
Figure 6.27-9 Timing Waveform for Byte Suspend..... 1479
Figure 6.27-10 SPI Half-Duplex Master Mode Application Block Diagram..... 1479
Figure 6.27-11 SPI Half-Duplex Slave Mode Application Block Diagram..... 1479
Figure 6.27-12 FIFO Threshold Comparator 1481
Figure 6.27-13 Transmit FIFO Buffer Example for 4~16 bits of data length..... 1482
Figure 6.27-14 Transmit FIFO Buffer Example for 17~32 bits of data length..... 1482
Figure 6.27-15 Receive FIFO Buffer Example for 16 bits of Data Length 1483
Figure 6.27-16 Receive FIFO Buffer Example for 32 bits of Data Length 1484

Figure 6.27-17 TX Underflow Event and Slave Under Run Event 1484
Figure 6.27-18 TX Underflow Event (SPI Slave 3-Wire Mode Enabled) 1485
Figure 6.27-19 Slave Mode Bit Count Error and Effective Bit Number of Uncompleted RX data 1486
Figure 6.27-20 I²S Data Format Timing Diagram 1488
Figure 6.27-21 MSB Justified Data Format Timing Diagram 1488
Figure 6.27-22 PCM Mode A Timing Diagram 1488
Figure 6.27-23 PCM Mode B Timing Diagram 1489
Figure 6.27-24 FIFO Contents for Various I²S Modes 1490
Figure 6.27-25 SPI Timing in Master Mode 1493
Figure 6.27-26 SPI Timing in Master Mode (Alternate Phase of SPIx_CLK) 1493
Figure 6.27-27 SPI Timing in Slave Mode 1494
Figure 6.27-28 SPI Timing in Slave Mode (Alternate Phase of SPIx_CLK) 1494
Figure 6.28-1 QSPI Block Diagram 1522
Figure 6.28-2 QSPI Peripheral Clock 1523
Figure 6.28-3 QSPI Full-Duplex Master Mode Application Block Diagram 1524
Figure 6.28-4 QSPI Full-Duplex Slave Mode Application Block Diagram 1524
Figure 6.28-532-bit in One Transaction 1525
Figure 6.28-6 Automatic Slave Selection (SSACTPOL = 0, SUSPITV > 0x2) 1526
Figure 6.28-7 Automatic Slave Selection (SSACTPOL = 0, SUSPITV < 0x3) 1526
Figure 6.28-8 Byte Reorder Function 1527
Figure 6.28-9 Timing Waveform for Byte Suspend 1527
Figure 6.28-10 QSPI Half-Duplex Master Mode Application Block Diagram 1528
Figure 6.28-11 QSPI Half-Duplex Slave Mode Application Block Diagram 1528
Figure 6.28-12 Two-bit Transfer Mode System Architecture 1530
Figure 6.28-13 Two-bit Transfer Mode Timing (Master Mode) 1530
Figure 6.28-14 Bit Sequence of Dual Output Mode 1531
Figure 6.28-15 Bit Sequence of Dual Input Mode 1531
Figure 6.28-16 Bit Sequence of Quad Output Mode 1532
Figure 6.28-17 Bit Sequence of Quad Input Mode 1533
Figure 6.28-18 Timing Diagram of QSPI Output Data for Transmit Double Transfer Rate Mode 1533
Figure 6.28-19 FIFO Threshold Comparator 1534
Figure 6.28-20 Transmit FIFO Buffer Example 1535
Figure 6.28-21 Receive FIFO Buffer Example 1536
Figure 6.28-22 TX Underflow Event and Slave Under Run Event 1536
Figure 6.28-23 Two-bit Transfer Mode FIFO Buffer Example 1537
Figure 6.28-24 TX Underflow Event (QSPI Slave 3-Wire Mode Enabled) 1537

Figure 6.28-25 Slave Mode Bit Count Error and Effective Bit Number of Uncompleted RX Data 1538
Figure 6.28-26 Slave Time-out Event 1538
Figure 6.28-27 QSPI Timing in Master Mode 1541
Figure 6.28-28 QSPI Timing in Master Mode (Alternate Phase of QSPIx_CLK) 1541
Figure 6.28-29 QSPI Timing in Slave Mode 1542
Figure 6.28-30 QSPI Timing in Slave Mode (Alternate Phase of QSPIx_CLK) 1542
Figure 6.29-1 SPIM Block Diagram 1562
Figure 6.29-2 SPIM Timing Diagram for TX and RX Transfers 1563
Figure 6.29-3 SPIM_SS Active Setup Time Relative to SPIM_CLK, and SPIM_SS Deselect Time 1563
Figure 6.29-4 TYPE_1 Program Command Flow with Quad Mode Write 1565
Figure 6.29-5 TYPE_2 Program Command Flow with Quad Mode Write 1566
Figure 6.29-6 TYPE_3 Program Command Flow with Quad Mode Write 1567
Figure 6.29-7 Fast Read Quad I/O Command 0xEB with Continuous Read Mode Disable 1567
Figure 6.29-8 Fast Read Quad I/O Command 0xEB with Continuous Read Mode Enable 1568
Figure 6.29-9 DTR Fast read command 0x0D 1571
Figure 6.29-10 DTR Fast Read Dual I/O Command 0xBD 1571
Figure 6.29-11 DTR Fast Read Quad I/O Command 0xED 1572
Figure 6.30-1 I²C Controller Block Diagram 1594
Figure 6.30-2 I²C Bus Timing 1595
Figure 6.30-3 I²C Protocol 1596
Figure 6.30-4 START and STOP Conditions 1596
Figure 6.30-5 Bit Transfer on the I²C Bus 1597
Figure 6.30-6 Acknowledge on the I²C Bus 1597
Figure 6.30-7 Master Transmits Data to Slave by 7-bit 1598
Figure 6.30-8 Master Reads Data from Slave by 7-bit 1598
Figure 6.30-9 Master Transmits Data to Slave by 10-bit 1598
Figure 6.30-10 Master Reads Data from Slave by 10-bit 1599
Figure 6.30-11 Control I²C Bus according to the Current I²C Status 1599
Figure 6.30-12 Master Transmitter Mode Control Flow 1600
Figure 6.30-13 Master Receiver Mode Control Flow 1601
Figure 6.30-14 Slave Mode Control Flow 1602
Figure 6.30-15 GC Mode 1603
Figure 6.30-16 Arbitration Lost 1604
Figure 6.30-17 Bus Management Packet Protocol Diagram Element Key 1606
Figure 6.30-187-bit Addressable Device to Host Communication 1607
Figure 6.30-197-bit Addressable Device Responds to an ARA 1607

Figure 6.30-20 Bus Management ALERT Function 1608
Figure 6.30-21 Bus Management Time Out Timing 1609
Figure 6.30-22 Bus Clock Low Time Out Timing 1609
Figure 6.30-23 Two-level BUFFER Mode Slave Read Control 1611
Figure 6.30-24 Two-level BUFFER Mode Slave Write Control 1612
Figure 6.30-25 Two-level BUFFER Mode Slave Read Limitation 1613
Figure 6.30-26 Setup Time Wrong Adjustment 1614
Figure 6.30-27 Hold Time Wrong Adjustment 1614
Figure 6.30-28 I²C Data Shifting Direction 1615
Figure 6.30-29 I²C Time-out Count Block Diagram 1617
Figure 6.30-30 I²C Wake-Up Related Signals Waveform 1618
Figure 6.30-31 EEPROM Random Read 1619
Figure 6.30-32 Protocol of EEPROM Random Read 1620
Figure 6.31-1 SD Host Controller Block Diagram 1647
Figure 6.31-2 PAD (Physical Address Descriptor) Table Format 1649
Figure 6.32-1 PSIO Block Diagram 1672
Figure 6.32-2 PSIO Clock Control Diagram (8-bit Pre-scale Counter in Clock Controller) 1673
Figure 6.32-3 Basic Operation 1675
Figure 6.32-4 Slot Controller Normal Repeat Mode 1676
Figure 6.32-5 Slot Controller Normal Repeat Mode with Infinite Loops 1677
Figure 6.32-6 Slot Controller Repeat Mode with Whole Repeat Mode 1678
Figure 6.32-7 Initial State, Interval State and IDLE Flag 1678
Figure 6.32-8 Switch I/O Mode 1679
Figure 6.32-9 PIN Delay Time and Devise Response Time 1680
Figure 6.32-10 Data Width Larger Than 16-bit 1680
Figure 6.32-11 Data Width Between 16-bit and 8-bit 1681
Figure 6.32-12 Data Width Smaller Than 8-bit 1681
Figure 6.32-13 OUTPUT DATA Buffer Example 1682
Figure 6.32-14 INPUT DATA Buffer Example 1682
Figure 6.32-15 Configurable Interrupt Flag 1683
Figure 6.32-16 MISMATIF, TERRIF and SCnIF 1684
Figure 6.32-17 Data Transfer with PDMA 1685
Figure 6.32-18 Normal Programing Example 1688
Figure 6.32-19 Programing Example with PDMA and Repeat Mode 1690
Figure 6.32-20 PDMA with OUTPUT DATA and Endless SLOT_PERIOD_LOOP_CNT 1693
Figure 6.32-21 PDMA with OUTPUT DATA and SLOT CONTROLLER WHOLE REPEAT Mode with 3 Pins 1697
Figure 6.32-22 PDMA with INPUT DATA and SLOT CONTROLLER WHOLE REPEAT Mode 1699

Figure 6.32-23 PDMA with INPUT DATA and Endless SLOT_PERIOD_LOOP_CNT 1701
Figure 6.32-24 PDMA with INPUT DATA and SLOT CONTROLLER WHOLE REPEAT Mode with 2 Pins 1705
Figure 6.33-1 USB Block Diagram 1744
Figure 6.33-2 NEVWK Interrupt Operation Flow 1746
Figure 6.33-3 Endpoint SRAM Structure 1747
Figure 6.33-4 Setup Transaction Followed by Data IN Transaction 1748
Figure 6.33-5 Data Out Transfer 1748
Figure 6.33-6 LPM State Transition Diagram 1750
Figure 6.34-1 USB Device Controller Block Diagram 1788
Figure 6.34-2 USB BC 1.2 Detection Flow Block Diagram 1791
Figure 6.35-1 USB 2.0 HS/FS Host Controller Block Diagram 1849
Figure 6.36-1 USB OTG Block Diagram 1905
Figure 6.36-2 USB Device Mode 1907
Figure 6.36-3 USB Host Mode 1907
Figure 6.36-4 USB ID Dependent Mode 1908
Figure 6.37-1 HS USB OTG Block Diagram 1922
Figure 6.37-2 USB Device Mode 1923
Figure 6.37-3 USB Host Mode 1924
Figure 6.38-1 CAN FD Controller Block Diagram 1940
Figure 6.38-2 Transceiver Delay Measurement 1944
Figure 6.38-3 Pin Controller in Bus Monitor Mode 1945
Figure 6.38-4 Pin Control in Loop Back Modes 1947
Figure 6.38-5 Standard Message ID Filter Path 1950
Figure 6.38-6 Extended Message ID Filter Path 1951
Figure 6.38-7 Rx FIFO Status 1952
Figure 6.38-8 Rx FIFO Overflow Handling 1953
Figure 6.38-9 Example of Mixed Configuration Dedicated Tx Buffers / Tx FIFO 1957
Figure 6.38-10 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue 1957
Figure 6.38-11 Message RAM configuration 1959
Figure 6.38-12 Bit Timing 1967
Figure 6.39-1 CRC Generator Block Diagram 2023
Figure 6.39-2 CHECKSUM Bit Order Reverse Functional Block 2024
Figure 6.39-3 Write Data Bit Order Reverse Functional Block 2025
Figure 6.40-1 Cryptographic Accelerator Block Diagram 2035
Figure 6.40-2 PRNG Function Diagram 2038
Figure 6.40-3 Electronic Codebook Mode 2040
Figure 6.40-4 Cipher Block Chaining Mode 2041

Figure 6.40-5 Cipher Feedback Mode 2042
Figure 6.40-6 Output Feedback Mode 2043
Figure 6.40-7 Counter Mode 2044
Figure 6.40-8 CBC-CS1 Encryption 2044
Figure 6.40-9 CBC-CS1 Decryption 2045
Figure 6.40-10 CCM Encryption and Decryption Flow 2046
Figure 6.40-11 GCM Encryption and Decryption Flow 2048
Figure 6.40-12 Main Hierarchy Chart of ECC 2054
Figure 6.41-1 Cryptographic Accelerator Block Diagram 2188
Figure 6.41-2 PRNG Function Diagram 2189
Figure 6.41-3 Electronic Codebook Mode 2192
Figure 6.41-4 Cipher Block Chaining Mode 2193
Figure 6.41-5 Cipher Feedback Mode 2194
Figure 6.41-6 Output Feedback Mode 2195
Figure 6.41-7 Counter Mode 2196
Figure 6.41-8 CBC-CS1 Encryption 2196
Figure 6.41-9 CBC-CS1 Decryption 2197
Figure 6.42-1 Key Store Block Diagram 2225
Figure 6.43-1 Keypad Controller Block Diagram 2243
Figure 6.43-2 Keypad Connection 2244
Figure 6.43-3 Keypad Interface 2244
Figure 6.43-46 x8 Keypad Scan Timing Diagram 2245
Figure 6.43-5 Key Array Scan Delay Time Diagram 2245
Figure 6.44-1 EADC Converter Block Diagram 2259
Figure 6.44-2 Sample Module 0~3 Block Diagram 2261
Figure 6.44-3 Sample Module 4~15 Block Diagram 2262
Figure 6.44-4 Sample Module 16~18 Block Diagram 2262
Figure 6.44-5 EADC Clock Control 2263
Figure 6.44-6 Example EADC Conversion Timing Diagram, n=0~18 2264
Figure 6.44-7 Sample Module Conversion Priority Arbitrator Diagram 2265
Figure 6.44-8 Specific Sample Module EADC EOC Signal for ADINT0~3 Interrupt 2266
Figure 6.44-9 EPWM-triggered EADC Start Conversion 2267
Figure 6.44-10 External triggered EADC Start Conversion 2267
Figure 6.44-11 Conversion Start Delay Timing Diagram 2268
Figure 6.44-12 EADC0_ST De-bounce Timing Diagram 2268
Figure 6.44-13 EADC Extend Sampling Timing Diagram 2269
Figure 6.44-14 EADC Conversion Result Monitor Logics Diagram 2271

Figure 6.44-15 Input of Fully-Differential EADC 2271
Figure 6.44-16 EADC Controller Interrupts 2273
Figure 6.44-17 Calibration Mode with 16 Times Average 2274
Figure 6.45-1 Digital-to-Analog Converter Block Diagram 2316
Figure 6.45-2 Data Holding Register Format 2317
Figure 6.45-3 DAC Conversion Started by Software Write Trigger 2318
Figure 6.45-4 DAC Conversion Started by Hardware Trigger Event 2318
Figure 6.45-5 DAC0 and DAC1 Group and Ungroup Update Example 2319
Figure 6.45-6 DAC PDMA Under-Run Condition Example 2320
Figure 6.45-7 DAC Continuous Conversion with Software PDMA Mode 2320
Figure 6.45-8 DAC Interrupt Source 2321
Figure 6.46-1 ACMP01 Block Diagram 2340
Figure 6.46-2 ACMP23 Block Diagram 2340
Figure 6.46-3 Comparator Hysteresis Function of ACMP0 2341
Figure 6.46-4 Window Latch Mode 2342
Figure 6.46-5 Example of Filter Function 2342
Figure 6.46-6 ACMP01 Interrupt Diagram 2343
Figure 6.46-7 ACMP23 Interrupt Diagram 2343
Figure 6.46-8 Comparator0 Reference Voltage Block Diagram 2344
Figure 6.46-9 Example of Window Compare Mode 2345
Figure 6.46-10 Example of Window Compare Mode 2345

LIST OF TABLE

Table 4.1-1 M463YGCAE Multi-function Pin Table 78
Table 4.1-2 M463LGCAE Multi-function Pin Table 81
Table 4.1-3 M463SGCAE Multi-function Pin Table 85
Table 4.1-4 M463KGCAE Multi-function Pin Table 91
Table 4.1-5 M463L2GCAC Multi-function Pin Table 94
Table 4.1-6 M463S2GCAC Multi-function Pin Table 97
Table 4.1-7 M463K2GCAC Multi-function Pin Table 101
Table 4.1-8 M467SJHAN Multi-function Pin Table 104
Table 4.1-9 M467KJHAN Multi-function Pin Table 109
Table 4.1-10 M467JJHAN Multi-function Pin Table 114
Table 4.1-11 M467HJHAN Multi-function Pin Table 120
Table 4.1-12 M467S2JHAE Multi-function Pin Table 123
Table 4.1-13 M467K2JHAE Multi-function Pin Table 128
Table 4.1-14 M467J2JHAE Multi-function Pin Table 133
Table 4.1-15 M467H2JHAE Multi-function Pin Table 139
Table 6.2-1 Reset Value of Registers 172
Table 6.2-2 Power Mode Table 177
Table 6.2-3 Power Mode Difference Table 177
Table 6.2-4 Power Mode Definition Table 178
Table 6.2-5 Clocks in Power Modes 179
Table 6.2-6 Re-Entering Power-down Mode Condition 182
Table 6.2-7 Address Space Assignments for On-Chip Controllers 186
Table 6.2-8 Exception Model 327
Table 6.2-9 Interrupt Number Table 330
Table 6.2-10 Priority Grouping 364
Table 6.3-1 Clock Stable Count Value Table 373
Table 6.3-2 Each Clock Source Enable Bit and Corresponding Stable Flag Table 373
Table 6.3-3 Symbol definition of PLL Output Frequency formula 414
Table 6.5-1 FMC Features Comparison Table at Different Chip 455
Table 6.5-2 Block Address Range 458
Table 6.5-3 ISP Command List 486
Table 6.5-4 FMC Control Registers for Flash Programming 488
Table 6.5-5 Lock Effect Table with Four Protections for CPU 496
Table 6.5-6 Lock Effect Table with Four Protections for ICE/ ICP/Writer 496
Table 6.6-1 EBI Address Mapping 523
Table 6.6-2 Timing Control Parameter 527

Table 6.9-1 Channel Priority Table 591
Table 6.10-1 PWM Pulse Generation Event Priority in Up Count Type 669
Table 6.10-2 PWM Pulse Generation Event Priority in Down Count Type 670
Table 6.10-3 PWM Pulse Generation Event Priority in Up-Down Count Type 670
Table 6.11-1 BPWM Clock Source Control Registers Setting Table 741
Table 6.11-2 BPWM Pulse Generation Event Priority for Up-Counter 749
Table 6.11-3 BPWM Pulse Generation Event Priority for Down-Counter 750
Table 6.11-4 BPWM Pulse Generation Event Priority for Up-Down-Counter 750
Table 6.12-1 EPWM Clock Source Control Registers Setting Table 796
Table 6.12-2 EPWM Pulse Generation Event Priority for Up-Counter 811
Table 6.12-3 EPWM Pulse Generation Event Priority for Down-Counter 811
Table 6.12-4 EPWM Pulse Generation Event Priority for Up-Down-Counter 811
Table 6.13-1 Direction of Counting 935
Table 6.13-2 Swap Function Table 940
Table 6.14-1 Watchdog Timer Time-out Interval Period Selection 962
Table 6.15-1 WWDT Prescale Value Selection 970
Table 6.15-2 CMPDAT Setting Limitation 973
Table 6.16-1 RTC Features Comparison Table at Different Chip 981
Table 6.16-2 RTC Read/Write Enable 983
Table 6.16-312/24 Hour Time Scale Selection 984
Table 6.16-4 Register Value After Powered On 985
Table 6.16-5 Dynamic Pattern Source Selection 987
Table 6.16-6 Tamper Control Bit Effection for Pair 0 987
Table 6.16-7 Tamper Control Bit Effection for Pair 1 and 2 988
Table 6.18-1 Typical Case of Noise Filter Settings 1053
Table 6.19-1 Input Signals for Different Protocols 1070
Table 6.19-2 Output Signals for Different Protocols 1071
Table 6.19-3 Data Transfer Events and Interrupt Handling 1078
Table 6.19-4 Protocol-specific Events and Interrupt Handling 1078
Table 6.20-1 Input Signals for UART Protocol 1082
Table 6.20-2 Output Signals for UART Protocol 1082
Table 6.20-3 Baud Rate Relationship 1085
Table 6.21-1 Serial Bus Clock Configuration 1116
Table 6.22-1 Relationship between I^2C Baud Rate and PCLK 1167
Table 6.23-1 M463/M467 Series UART Features 1190
Table 6.23-2 UART Interrupt 1193
Table 6.23-3 UART Interface Controller Pin 1195

Table 6.23-4 UART Controller Baud Rate Equation Table 1196
Table 6.23-5 UART Controller Baud Rate Parameter Setting Example Table 1196
Table 6.23-6 UART Controller Baud Rate Register Setting Example Table..... 1196
Table 6.23-7 Baud Rate Compensation Example Table 1 1197
Table 6.23-8 Baud Rate Compensation Example Table 2 1198
Table 6.23-9 UART Controller Interrupt Source and Flag List..... 1204
Table 6.23-10 UART Line Control of Word and Stop Length Setting 1205
Table 6.23-11 UART Line Control of PARITY bit Setting..... 1205
Table 6.23-12 LIN Header Selection in Master Mode..... 1210
Table 6.24-1 Minimum PTP Clock Frequency Example 1277
Table 6.24-2 Ordinary Clock: PTP Messages for Snapshot 1279
Table 6.24-3 End to End Transparent Clock: PTP Messages for Snapshot..... 1280
Table 6.24-4 Peer-to-Peer Transparent Clock: PTP Messages for Snapshot..... 1280
Table 6.24-5 Message Format Defined in IEEE 1588-2008 1281
Table 6.24-6 IPv4-UDP PTP Frame Fields Required for Control and Status 1282
Table 6.24-7 IPv6-UDP PTP Frame Fields Required for Control and Status 1282
Table 6.24-8 Ethernet PTP Frame Fields Required for Control And Status 1283
Table 6.24-9 CRC Replacement Based on TDES0 Bit 27 and Bit 24 1287
Table 6.24-10 Transmit Checksum Offload Engine Functions for Different Packet Types 1289
Table 6.24-11 Receive Checksum Offload Engine (Type 2) Functions for Different Packet Types 1291
Table 6.24-12 MDIO Frame Structure..... 1293
Table 6.24-13 Transmit Descriptor 0..... 1299
Table 6.24-14 Transmit Descriptor 1..... 1300
Table 6.24-15 Transmit Descriptor 2..... 1300
Table 6.24-16 Transmit Descriptor 3..... 1300
Table 6.24-17 Receive Descriptor 0..... 1303
Table 6.24-18 Receive Descriptor 0 when COE (Type 2) is Enabled..... 1303
Table 6.24-19 Receive Descriptor 1..... 1304
Table 6.24-20 Receive Descriptor 2..... 1304
Table 6.24-21 Receive Descriptor 3..... 1305
Table 6.24-22 Transmit Timestamp Status – Normal Descriptor Format Case (TDES0)..... 1306
Table 6.24-23 Transmit Timestamp Control – Normal Descriptor Format Case (TDES1) 1306
Table 6.24-24 Transmit Descriptor Fields (TDES2)..... 1307
Table 6.24-25 Transmit Descriptor Fields (TDES3)..... 1307
Table 6.24-26 Receive Descriptor Fields (RDES2) 1307
Table 6.24-27 Receive Descriptor Fields (RDES3) 1308
Table 6.24-28 Transmit Descriptor Word 0 (TDES0)..... 1312

Table 6.24-29 Transmit Descriptor Word 1 (TDES1) 1312
Table 6.24-30 Transmit Descriptor 2 (TDES2) 1313
Table 6.24-31 Transmit Descriptor 3 (TDES3) 1313
Table 6.24-32 Transmit Descriptor 6 (TDES6) 1313
Table 6.24-33 Transmit Descriptor 7 (TDES7) 1313
Table 6.24-34 Receive Descriptor Fields (RDES0) 1316
Table 6.24-35 Receive Descriptor Fields 1 (RDES1) 1316
Table 6.24-36 Receive Descriptor Fields 2 (RDES2) 1317
Table 6.24-37 Receive Descriptor Fields 3 (RDES3) 1317
Table 6.24-38 Receive Descriptor Fields 4 (RDES4) 1319
Table 6.24-39 Receive Descriptor Fields 6 (RDES6) 1319
Table 6.24-40 Receive Descriptor Fields 7 (RDES7) 1319
Table 6.25-1 SC Host Controller Pin Description 1402
Table 6.25-2 UART Pin Description 1402
Table 6.25-3 Timer0/Timer1/Timer2 Operation Mode 1413
Table 6.27-1 SPI/I^{2}S Interface Controller Pin Description (SPI0~SPI3) 1474
Table 6.27-2 Dummy Data Number for I^{2}S / PCM Master Mode and Monaural Mode 1491
Table 6.27-3 Dummy Data Number for I^{2}S / PCM Master Mode and Stereo Mode 1491
Table 6.27-4 Dummy Data Number for I^{2}S Slave Mode and Monaural Mode 1491
Table 6.27-5 Dummy Data Number for PCM Slave Mode and Monaural Mode 1492
Table 6.27-6 Dummy Data Number for I^{2}S Slave Mode and Stereo Mode 1492
Table 6.27-7 Dummy Data Number for PCM Slave Mode and Stereo Mode 1492
Table 6.29-1 Functional Description for Slave Select Active Level and Slave Active Enable ... 1563
Table 6.29-2 Mapping Address for SPI Flash Controller in AMBA System 1573
Table 6.30-1 Reserved SMBus Address 1605
Table 6.30-2 Relationship between I^{2}C Baud Rate and PCLK 1614
Table 6.30-3 I^{2}C Status Code Description 1616
Table 6.32-1 PSIO Output Data Rate when Bus Traffic is Heavy 1673
Table 6.32-2 PSIO Output Data Rate With Bus Is Free 1674
Table 6.32-3 PSIO Input Data Rate with Bus Is Free 1674
Table 6.32-4 PIN OUTPUT DELAY and PIN INPUT DELAY 1680
Table 6.33-1 USB Link Power Management (Lx) States 1749
Table 6.34-1 Scatter Gather Descriptor Format 1790
Table 6.38-1 DLC coding in CAN FD format 1943
Table 6.38-2 Rx Buffer / FIFO Element Size 1953
Table 6.38-3 Example Filter Configuration for Rx Buffers 1954
Table 6.38-4 Possible Configuration for Frame Transmission 1955

Table 6.38-5 Tx Buffer / FIFO / Queue Element Size 1956
Table 6.38-6 Rx Buffer and FIFO Element 1960
Table 6.38-7 Rx Buffer and FIFO Element Description 1962
Table 6.38-8 Tx Buffer and FIFO Element 1962
Table 6.38-9 Tx Buffer Element Description 1963
Table 6.38-10 Tx Event FIFO Element 1964
Table 6.38-11 Tx Event FIFO Element Description 1964
Table 6.38-12 Standard Message ID Filter Element 1965
Table 6.38-13 Standard Message ID Filter Element Description 1966
Table 6.38-14 Extended Message ID Filter Element 1966
Table 6.38-15 Extended Message ID Filter Element Description 1967
Table 6.40-1 Each Engine Error Conditions and Error Flag 2036
Table 6.40-2 DMA Enable Bit Table 2036
Table 6.40-3 DMA Cascade Bit Table 2037
Table 6.40-4 ECC Parameters and Corresponding Registers Table 2054
Table 6.40-5 Required Input Data of Various Operations 2055
Table 6.40-6 RSA Accelerator Support Situation 2072
Table 6.41-1 Each Engine Error Conditions and Error Flag 2188
Table 6.41-2 DMA Enable Bit Table 2188
Table 6.41-3 DMA Cascade Bit Table 2189
Table 6.42-1 Detailed Information for SRAM, Flash and OTP for M467 Series 2227
Table 6.42-2 Detailed Information for SRAM, Flash and OTP for M463 Series 2228
Table 6.42-3 Information of OTP Key Storage 2228
Table 6.42-4 Byte Counts for Key Size at SRAM and Flash 2231
Table 6.42-5 CPU and Crypto Engine Access Key for All Situations 2232
Table 6.44-1 Setting of Accumulation and Conversion Result Precision 2270
Table 6.44-2 Setting of Averaging and Conversion Result Precision 2270
Table 6.44-3 EADC Differential Model Channel Selection 2272
Table 6.46-1 Truth Table of Window Compare Logic 2345
Table 6.47-1 Peripherals Interconnect Matrix table 2360
Table 10.1-1 List of Abbreviations 2378

1 GENERAL DESCRIPTION

The NuMicro M463/M467 series is a 32-bit microcontroller based on Arm Cortex-M4F core, with DSP instruction set and single-precision floating-point unit (FPU), targeted for IoT, Industrial, and consumer applications. The M463/M467 series runs up to 200 MHz, and features 1.7 V to 3.6 V wide operating voltage, -40 °C to 85 °C /105°C /125°C wide operating temperature, a variety of packages choice, and excellent high immunity characteristics by ESD HBM 2 KV and EFT 4.4 KV.

As the new smart function added on home appliances, the M463/M467 series provides up to 1024 KB dual-bank of Flash memory for code storage and 512 KB SRAM for run time operation. The dual bank design of 1024 KB Flash memory supports the Firmware update through the Over-The-Air (FOTA) process. Additionally, in response to the code security requirements, the M463/M467 series supports Execute-Only Memory (XOM) function to protect confidential program code information from stealing in the run-time. In order to reduce the data access overhead of CPU core to peripherals, up to 32 channels of peripheral direct memory access (PDMA) is provided.

The M463/M467 series supports plenty of peripherals, including Ethernet 10/100 MAC, hardware crypto engine, key store, true random number generator (TRNG), programmable audio PLL, HyperBus interface, 4 sets of CAN FD, USB HS OTG, USB FS OTG, up to 24 channels of 16-bit PWM, 10 sets of UART, 4 sets of SPI/I^{2}S, 2 sets of Quad-SPI, 5 sets of I^{2}C, 1 set of USCI, 1 set of PSIO, 4 sets of EQEI and a real-time clock (RTC).

The M463/M467 series also provides rich analog peripherals including 4 sets of analog comparators, up to 28 channels of 12-bit SAR ADC, and 2 channels of 12-bit DAC.

For the development, Nuvoton provides the NuMaker-M467HJ, NuMaker-M463KG evaluation board, and Nuvoton Nu-Link debugger. The 3rd Party IDE such as Keil MDK, IAR EWARM, Eclippse IDE with GNU GCC compilers are also supported.

Product Line	Core (MHz)	Ethernet	Crypto + TRNG	CAN FD	USB OTG	HBI	EBI	UART	I^{2}C	QSPI/SPI	PWM	ADC	DAC	ACMP	EQEI
M463 Series	200	-	√	2	HS x1	-	√	8+1	5	SPI x4 QSPI x2	24	16	-	2	2
M467 Series	200	√	√	4	HS x1 FS x1	√	√	10+3	5	SPI x4 QSPI x2	24	28	2	4	4

This series supports eleven package choices which are designed for home appliance PCB demands.

- QFN48: Body Size 5 mm x 5 mm, Load pitch: 0.35 mm
- LQFP48: Body Size 7 mm x 7 mm, Load Pitch 0.5 mm
- LQFP64: Body Size 7 mm x 7 mm, Load Pitch 0.4 mm
- LQFP128: Body Size 14 mm x 14 mm, Load Pitch 0.4 mm
- LQFP144: Body Size 20 mm x 20 mm, Load Pitch 0.5 mm
- LQFP176: Body Size 24 mm x 24 mm, Load Pitch 0.5 mm
- LQFP48 EPAD: Body Size 7 mm x 7 mm, Load Pitch 0.5 mm, EP Size 5.36 x 5.36 mm
- LQFP64 EPAD: Body Size 7 mm x 7 mm, Load Pitch 0.4 mm, EP Size 5.48 x 5.48 mm
- LQFP128 EPAD: Body Size 14 mm x 14 mm, Load Pitch 0.4 mm, EP Size 7.15 x 7.15 mm
- LQFP144 EPAD: Body Size 20 mm x 20 mm, Load Pitch 0.5 mm, EP Size 5.9 x 5.9 mm
- LQFP176 EPAD: Body Size 24 mm x 24 mm, Load Pitch 0.5 mm, EP Size 6.75 x 6.75 mm

The NuMicro M463/M467 series is suitable for a wide range of applications such as:

- IoT Gateway
- Industrial Control
- Telecom
- Data Center

2 FEATURES

2.1 M463/M467 Series Features

Core and System	
ARM Cortex-M4	• Arm Cortex-M4 processor, running up to 200 MHz • Built-in Memory Protection Unit (MPU) • Built-in Nested Vectored Interrupt Controller (NVIC) • Hardware IEEE 754 compliant Floating-point Unit (FPU) • DSP extension with hardware divider and single-cycle 32-bit hardware multiplier • 24-bit system tick timer • Programmble and maskable interrupt • Low Power Sleep mode by WFI and WFE instructions
Brown-out Detector (BOD)	• Eight-level BOD with brown-out interrupt and reset option
Low Voltage Reset (LVR)	• LVR with 1.5V threshold voltage level
Security	• 96-bit Unique ID (UID) • 128-bit Unique Customer ID (UCID)
Memories	
Boot Loader	• Factory pre-loaded mask ROM for secure boot procedure and ISP procedure • Uses SHA-256 and ECDSA-P256 to validate data in APROM and LDROM • NuMicro ISP Programming Tool for firmware upgrade via UART, CAN FD, SPI, I^2C and high speed USB device
Flash	• Up to 1024 Kbytes on-chip Application ROM (APROM) • Dual bank 1024 Kbytes APROM for Over-The-Air (OTA) upgrade • Four eXecute-Only-Memory (XOM) regions for code protection • Embedded with 4 Kbytes cache, with performance at zero wait cycle in continuous address read access • 8 Kbytes on-chip Flash for user-defined loader (LDROM) • 3 Kbytes One Time Programmable (OTP) ROM for data security • All on-chip Flash support 4 Kbytes page erase • Fast Flash programming verification with CRC32 • On-chip Flash programming with In-Chip Programming (ICP), In-System Programming (ISP) and In-Application Programming (IAP) capabilities • Configurable boot up sources including boot loader, user-defined

	loader (LDROM) or Application ROM (APROM) • Data Flash with configurable memory size • 2-wired ICP Flash updating through SWD interface • 32-bit/64-bit and multi-word Flash programming function
SRAM	• Up to 512 Kbytes on-chip SRAM includes: – 128 Kbytes SRAM located in bank 0 with programmable size of SPD retention The first 64 Kbytes supports hardware parity check; Exception (NMI) generated upon a parity check error – 128 Kbytes SRAM located in bank 1 with SPD retention – 256 Kbytes SRAM located in bank 2 that can be used as cache for external SPI Flash memory • Supports byte-, half-word- and word-access • PDMA operation
Cyclic Redundancy Calculation (CRC)	• Supports 8-bits, 16-bits and 32-bits configurable polynomials, CRC-CCITT, CRC-8, CRC-16 and CRC-32 polynomials included • Programmable initial value • Supports order reverse setting and one's complement setting for input data and CRC checksum • Supports 8-bit, 16-bit, and 32-bit data width • Supports using PDMA to write data to perform CRC operation
Peripheral DMA (PDMA)	• Up to 32 independent and configurable channels for automatic data transfer between memories and peripherals • Basic and Scatter-Gather transfer modes • Each channel supports circular buffer management using Scatter-Gather Transfer mode • Stride function for rectangle image data movement • Supports Fixed-priority and Round-robin priority modes • Single and burst transfer types • Byte-, half-word- and word tranfer unit with count up to 65536 • Incremental or fixed source and destination address
Clocks	
External Clock Source	• 4~24 MHz High-speed eXternal crystal oscillator (HXT) for precise timing operation • 32.768 kHz Low-speed eXternal crystal oscillator (LXT) for RTC function and low-power system operation • Supports clock failure detection for external crystal oscillators and exception generation (NMI)
Internal Clock Source	• 48 MHz High-speed Internal RC oscillator (HIRC48) for crystal-less USB

	• 12 MHz High-speed Internal RC oscillator (HIRC) can optionally be used as a system clock • 10 kHz Low-speed Internal RC oscillator (LIRC) for watchdog timer and wakeup operation • Up to 400 MHz on-chip PLL, sourced from HIRC or HXT, allowing CPU operation up to the maximim CPU frequency without the need for a high-frequency crystal • Programmable Audio PLL (PLLFN), sourced from HIRC or HXT
Real-Time Clock (RTC)	• Real-Time Clock with a separate power domain and independent V_{BAT} pin • Supports 80 bytes of battery-powered backup registers, which can be cleared by tamper pins • Supports 6 static and dynamic tamper pins • Able to wake up CPU from any reduced power mode • Supports frequency compensation of RTC clock source • Supports Alarm registers (second, minute, hour, day, month, year) • Supports RTC Time Tick and Alarm Match interrupt • Automatic leap year recognition • Supports 1 Hz clock output for calibration
Timers	
32-bit Timer	**TIMER** • Four sets of 32-bit timers with 24-bit up counter and one 8-bit pre-scale counter from independent clock source • One-shot, Periodic, Toggle-output and Continuous Counting operation modes • Supports event counting function to count the event from external pins • Supports external capture pin for interval measurement and resetting 24-bit up counter • Supports chip wake-up function, if a timer interrupt signal is generated **PWM** • Eight 16-bit PWM counters with 12-bit clock prescale • Supports 12-bit deadband (dead time) • Up, down or up-down PWM counter type • Supports brake function • Supports mask function and tri-state output for each PWM channel
Enhanced PWM (EPWM)	• Twelve 16-bit counters with 12-bit clock prescale for twelve 200 MHz PWM output channels • Up to 12 independent input capture channels with 16-bit resolution counter

	• Supports dead time with maximum divided 12-bit prescale • Up, down or up-down PWM counter type • Supports complementary mode for 3 complementary paired PWM output channels • Synchronous function for phase control • Counter synchronous start function • Brake function with auto recovery mechanism • Mask function and tri-state output for each PWM channel • Trigger EADC or DAC to start conversion immediately • Trigger EADC to start conversion after a short delay • Supports External Pin Trigger Function
Basic PWM (BPWM)	• Two 16-bit counters with 12-bit clock prescale for twelve 200 MHz PWM output channels • Up to 6 independent input capture channels with 16-bit resolution counter • Up, down or up-down PWM counter type • Counter synchronous start function • Complementary mode for 3 complementary paired PWM output channels • Mask function and tri-state output for each PWM channel • Able to trigger EADC to start conversion
Watchdog	• 18-bit free running up counter for WDT time-out interval • Supports multiple clock sources from LIRC (default selection), HCLK/2048 and LXT with 8 selectable time-out period • Able to wake up system from Power-down or Idle mode • Time-out event to trigger interrupt or reset system • Supports four WDT reset delay periods, including 1026, 130, 18 or 3 WDT_CLK reset delay period • Configured to force WDT enabled on chip power-on or reset
Window Watchdog	• Clock sourced from HCLK/2048 or LIRC; the window set by 6-bit counter with 11-bit prescale • Suspended in Idle/Power-down mode
Analog Interfaces	
Enhanced Analog-to-Digital Converter (EADC)	• Up to three sets of 12-bit, 19-ch 5 MSPS SAR EADC with up to 16 single-ended input channels or 8 differential input pairs; 10-bit accuracy is guaranteed • Three internal channels for V_{BAT}, band-gap VBG input and Temperature sensor input • Supports external V_{REF} pin or internal reference voltage • Supports Power-down mode

	• Supports calibration capability • Conversion can be triggered by software, external pin, Timer 0~3 overflow pulse or EPWM/BPWM • Configurable EADC sampling time • Up to 19 sample modules • Double data buffers for sample module 0~3 • PDMA operation • Supports Averageing mode and Oversampling mode, where Oversampling mode provides up to 16-bit precision • Supports early interrupt with delay counter feature
Digital-to-Analog Converter (DAC)	• Up to two sets of 12-bit, 1 MSPS voltage type DAC with 8-bit mode and 8μs rail-to-rail settle time • Maximum output voltage AV_{DD} -0.2V at buffer mode • Conversion can be triggered by software, Timer0~3, EPWM, external pin • Supports group mode for synchronized data update of two DACs • PDMA operation
Analog Comparator (ACMP)	• Up to four rail-to-rail Analog Comparators • Supports four multiplexed I/O pins at positive input • Supports I/O pins, band-gap, DAC, and 16-level Voltage divider from AV_{DD} or V_{REF} at negative input • Supports four programmable power modes for power saving • Supports wake up from Power-down by interrput • Supports triggers for brake events and cycle-by-cycle control for PWM • Supports window compare mode and window latch mode • Supports programmable hysteresis window: 0mV, 20mV and 40mV • Supports offset calibration
Communication Interfaces	
Low-power UART	• Low-power UARTs with up to 10 MHz baud rate • Auto-Baud Rate measurement and baud rate compensation function • Supports low power UART (LPUART): baud rate clock from LXT (32.768 kHz) with 9600 bps in Power-down mode even system clock is stopped • 16-byte FIFOs with programmable level trigger • Auto flow control (nCTS and nRTS) • Supports IrDA (SIR) function • Supports LIN function on UART0 and UART1 • Supports RS-485 9-bit mode and direction control

	• Supports nCTS, incoming data, Received Data FIFO reached threshold and RS-485 Address Match (AAD mode) wake-up function in idle mode • Supports hardware or software enables to program nRTS pin to control RS-485 transmission direction • 8-bit receiver FIFO time-out detection function • Supports break error, frame error, parity error and receive/transmit FIFO overflow detection function • Supports PDMA operation • Supports Single-wire function mode • Supports TX and RX swap function mode
Smart Card Interface	Up to three sets of Smart Card interfaces • ISO-7816-3 compliant with ISO-7816-3 T=0, T=1 • Supports full duplex UART function • 4-byte FIFOs with programmable level trigger • Programmable guard time selection (11 ETU ~ 266 ETU) • One 24-bit and two 8 bit time-out counters for Answer to Request (ATR) and waiting times processing • Auto inverse convention function • Stop clock level and clock stop (clock keep) function • Transmitter and receiver error retry function • Supports hardware activation, deactivation and warm reset sequence process • Supports hardware auto deactivation sequence after card removal
I^2C	• Up to five sets of I^2C devices with Master/Slave mode • Supports Standard mode (100 kbps), Fast mode (400 kbps), Fast mode plus (1 Mbps) and High speed mode (Up to 3.4Mbps) • Supports 10 bits mode • Programmable clocks allowing for versatile rate control • Supports multiple address recognition (four slave address with mask option) • Supports SMBus and PMBus • Supports multi-address power-down wake-up function • PDMA operation • Supports pin swap function • Supports setup/hold time programmable • Supports two level buffer mode
SPI Master (SPI Flash)	• Supports SPI master mode • One slave/device select line for external SPI Flash component • Maximum 32 Mbytes external SPI Flash memory with standard (1-bit), dual (2-bit) and quad (4-bit) transfer mode • 16 Kbytes cache memory for enhancing program execution

	performance • 64-bit key length for code protection • DMA mode for code transfer between SPI Flash memory and SRAM • SPI Master function with 8-, 16-, 24-, and 32-bit length of transaction and burst mode operation, which can transmit/receive data up to four successive transactions in one transfer • Supports Double Transfer Rate (DTR) mode • Supports eXcute-In-Place (XIP)
Quad SPI	• Up to two sets of Quad SPI with Master/Slave mode • Master mode up to 100 MHz (V_{DD} = 2.7V ~ 3.6V) • Slave mode up to 50 MHz (V_{DD} = 2.7V ~ 3.6V) • Supports Dual and Quad I/O Transfer mode • Supports one data channel half-duplex transfer • Supports double data rate mode (Master TX DIR Mode Only) • Supports receive-only mode • Configurable bit length of a transfer word from 8 to 32-bit • Provides separate 8-level depth transmit and receive FIFO buffers • Supports MSB first or LSB first transfer sequence • Supports the byte reorder function • Supports Byte or Word Suspend mode • Supports 3-wired, no slave select signal, bi-direction interface • PDMA operation
SPI/I²S	• Up to four sets of SPI/I²S controllers with Master/Slave mode • Provides separate 4-level of 32-bit (or 8-level of 16-bit) transmit and receive FIFO buffers **SPI** • Master mode up to 100 MHz (V_{DD} = 2.7V ~ 3.6V) • Slave mode up to 50 MHz (V_{DD} = 2.7V ~ 3.6V) • Configurable bit length of a transfer word from 4 to 32-bit • MSB first or LSB first transfer sequence • Byte reorder function • Supports Byte or Word Suspend mode • Supports one data channel half-duplex transfer • Supports receive-only mode **I²S** • Supports mono and stereo audio data with 8-, 16-, 24- and 32-bit audio data sizes • Supports PCM mode A, PCM mode B, I²S and MSB justified data format • PDMA operation

I²S	• Up to two sets of I²S interfaces with Master/Slave mode • Supports mono and stereo audio data with 8-, 16-, 24- and 32-bit word sizes • Two 16-level FIFO data buffers, one for transmitting and the other for receiving • Supports I²S protocols: Philips standard, MSB-justified, and LSB-justified data format • Supports PCM protocols: PCM standard, MSB-justified, and LSB-justified data format • PCM protocol supports TDM multi-channel transmission in one audio sample; the number of data channel can be set as 2, 4, 6 or 8 • PDMA operation
Universal Serial Control Interface (USCI)	• Configured as UART, SPI or I²C function • Supports single byte TX and RX buffer mode **UART** • Supports one transmit buffer and two receive buffers for data payload • Supports hardware auto flow control function and programmable flow control trigger level • 9-bit Data Transfer • Baud rate detection by built-in capture event of baud rate generator • Supports wake-up function • PDMA operation **SPI** • Supports Master or Slave mode operation • Supports one transmit buffer and two receive buffer for data payload • Configurable bit length of a transfer word from 4 to 16-bit • Supports MSB first or LSB first transfer sequence • Supports Word Suspend function • Supports 3-wire, no slave select signal, bi-direction interface • Supports wake-up function: input slave select transition • Supports one data channel half-duplex transfer • PDMA operation **I²C** • Supports master and slave device capability • Supports one transmit buffer and two receive buffer for data payload • Communication in standard mode (100 kbps), fast mode (up to 400 kbps), and Fast mode plus (1 Mbps) • Supports 10-bit mode

	• Supports 10-bit bus time out capability • Supports bus monitor mode • Supports power-down wake-up by data toggle or address match • Supports multiple address recognition • Supports device address flag • Programmable setup/hold time
Controller Area Network with Feasibility Data Rate (CAN FD)	• Up to four sets of CAN FD controllers • Compliant with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015 • Compliant with CAN FD version 1.0 with up to 64 data bytes supported • Supports CAN Error logging, AUTOSAR and SAE J1938 • Built-in 1.5K word (32-bit) Message SRAM for each CAN FD controller
Secure Digital Host Controller (SDHC)	• Compliant with SD Memory Card Specification Version 2.0 • Supports 50 MHz to achieve 200 Mbps at 3.3V operation • Supports dedicated DMA master with Scatter-Gather function to accelerate the data transfer between system memory and SD/SDHC/SDIO card
External Bus Interface (EBI)	• Supports up to three memory banks with individual adjustment of timing parameter • Supports dedicated external chip select pin with polarity control and up to 1 Mbytes addressing space for each bank • 8-/16-bit data width • Supports byte write in 16-bit data width mode • Configurable idle cycle for different access condition: Idle of Write command finish (W2X) and Idle of Read-to-Read (R2R) • Supports Address/Data multiplexed mode • Supports address bus and data bus separate mode • Supports LCD interface i80 mode • PDMA operation
Programmable Serial I/O (PSIO)	• Supports up to 8 PSIO pins • Supports 6 clock sources with clock divider • Supports 4 slot controllers for timing sequence control • Supports 8 check points to connect with slots in each pin • Supports 8 check point actions in each check point • Supports four I/O modes, input, output, open-drain, and quasi • Supports switch I/O mode in different check points • Supports four kinds of Interrupt trigger conditions • PDMA operation

GPIO	• Supports four I/O modes: Quasi bi-direction, Push-Pull output, Open-Drain output and Input only with high impendence mode • Selectable TTL/Schmitt trigger input • Configured as interrupt source with edge/level trigger setting • Supports independent pull-up/pull-down control • Supports software selectable slew rate control • Supports 5V-tolerance function except analog I/O • Configurable I/O mode of all pins after reset default to Quasi-bidirection mode or input mode
KeyPad Interface (KPI)	• Matrix keypad interface with up to 6x8 array • Programmable de-bounce time • Generates interrupt and update press/release status of all keys once key press or release detected
Control Interfaces	
Enhanced Quadrature Encoder Interface (EQEI)	• Two EQEI phase inputs (EQEI_A, EQEI_B) and one Index input (EQEI_INDEX) • Supports 2/4 times free-counting mode and 2/4 compare-counting mode • Supports encoder pulse width measurement mode with ECAP • Supports swap function for input signals (EQEI_A, EQEI_B) • Supports for detecting the occurrence of phase error • Supports one times index signal reset function
Enhanced Capture (ECAP)	• Up to four sets of Enhanced input Capture units • Supports three input channels with independent capture counter hold register • 24-bit Input Capture up-counting timer/counter supports captured events reset and/or reload capture counter • Supports rising edge, falling edge and both edge detector options with noise filter in front of input ports • Supports compare-match function
Advanced Connectivity	
USB 2.0 Full Speed with on-chip transceiver	**USB 2.0 Full Speed OTG (On-The-Go)** • On-chip USB 2.0 full speed OTG transceiver • Compliant with USB OTG Supplement 2.0 • Configurable as host-only, device-only, ID-dependent or OTG device **USB 2.0 Full Speed Host Controller** • Compliant with USB Revision 1.1 Specification • Compatible with OHCI (Open Host Controller Interface) Revision

	1.0 • Supports full-speed (12Mbps) and low-speed (1.5Mbps) USB devices • Supports Control, Bulk, Interrupt, Isochronous and Split transfers • Integrated a port routing logic to route full/low speed device to OHCI controller • Supports an integrated Root Hub • Supports port power control and port overcurrent detection • Built-in DMA **USB 2.0 Full Speed Device Controller** • Compliant with USB Revision 2.0 Specification • Supports suspend function when no bus activity exists for 3 ms • 25 configurable endpoints for configurable Isochronous, Bulk, Interrupt and Control transfer types • 1.5 Kbytes configurable RAM for endpoint buffer • Remote wake-up capability • Start of Frame (SOF) locked clock pulse generation for crystal-less feature • USB 2.0 link power management • Supports double buffer function
USB 2.0 High Speed with on-chip transceiver	**USB 2.0 High Speed OTG (On-The-Go)** • On-chip USB 2.0 high speed OTG transceiver • Compliant with USB OTG Supplement 2.0 • Configurable as host-only, device-only, ID-dependent or OTG device **USB 2.0 High Speed Host Controller** • Compliant with USB Revision 2.0 Specification • Compatible with EHCI (Enhanced Host Controller Interface) Revision 1.0 • Compatible with OHCI (Open Host Controller Interface) Revision 1.0 • Supports high-speed (480Mbps), full-speed (12Mbps) and low-speed (1.5Mbps) USB devices • Integrated a port routing logic to route full/low speed device to OHCI controller • Supports an integrated Root Hub • Built-in DMA **USB 2.0 High Speed Device Controller** • Compliant with USB Revision 2.0 Specification • Supports 12 configurable endpoints; each can be Isochronous, Bulk or Interrupt and either IN or OUT direction • Maximum packet size up to 1024 bytes • Three different operation modes of an in-endpoint: Auto Validation

	mode, Manual Validation mode and Fly mode • Suspend, resume and remote wake-up capability • Built-in DMA • Supports Battery Charging 1.2 (BC1.2) • Supports Link Power Management (LPM)
Ethernet MAC	• Compliant with IEEE Std 802.3-2008 for Ethernet MAC • Compliant with IEEE Std 1588-2008 for precision networked clock synchronization. • Compliant with RMII specification version 1.2 from RMII consortium • Full-duplex operation – IEEE 802.3x flow control automatic transmission of zero-quanta Pause frame on flow control input de-assertion – Forwarding of received Pause frames to the user application • Half-duplex operation – CSMA/CD Protocol support – Flow control using backpressure support – Frame bursting and frame extension in 1000 Mbps half-duplex operation – Programmable frame length to support Standard or Jumbo Ethernet frames with up to 16 Kbytes of size • IEEE 802.1Q VLAN tag detection for reception frames • Receive module for checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame (Type 1) • Enhanced Receive module for checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams (Type 2) • CRC replacement, Source Address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted frames with per-frame control • Serial management interface (MDC/MDIO) master mode for PHY device configuration and management • Supports Magic Packet recognition to wake system up from Power-down mode
HyperBus Interface (HBI)	• Supports HyperRAM 2.0 with HyperRAM 1.0 • Up to 90 MHz HyperBus clock Rate • Maximum 1440 Mbits/s data rate • Configurable reset timing control • Configurable chip select timing control for HyperRAM Hybrid Sleep or DPD mode
Digital Camera Interface	
Camera Capture Interface	• Supports CCIR601, CCIR656 and 4-bit interfaces for CMOS

(CCAP)	sensor • Up to 320x240 (QVGA) resolution • Color format for data input supports YUV4:2:2 and RGB565 • Color format for data output supports YUV4:2:2, RGB565, RGB555 and Y-only • Supports 1-bit Y(luminance) output with 8-bit threshold setting for 2-level image thresholding • Supports the CROP function to crop input image to the required size for digital application • Supports image scaling-down
Cryptography Accelerator	
Elliptic Curve Cryptography (ECC)	• Hardware ECC accelerator • Supports 163-bit ~ 571-bit key length • Supports both prime field GF(p) and binary field GF(2m) • Supports NIST P-192, P-224, P-256, P-384 and P-521 curve sizes • Supports NIST B-163, B-233, B-283, B-409 and B-571 curve sizes • Supports NIST K-163, K-233, K-283, K-409 and K-571 curve sizes • Supports point multiplication, addition and doubling operations in GF(p) and GF(2m) • Supports modulus division, multiplication, addition and subtraction operations in GF(p) • Supports Curve25519 • Improved side-channel attack protection • Supports Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves
Rivest–Shamir–Adleman Cryptography (RSA)	• Supports both encryption and decryption with up to 4096 bits • CRT decryption with up to 4096 bits • Improved side-channel attack protection ability
Advanced Encryption Standard (AES)	• Hardware AES accelerator • Supports 128-bit, 192-bit and 256-bit key length and key expander, and is compliant with FIPS 197 • Supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2 and CBC-CS3 block cipher modes • Compliant with NIST SP800-38A and addendum • Supports CCM mode, GCM mode and GMAC function • Improved side-channel attack protection ability
Secure Hash Algorithm (SHA)	• Hardware SHA accelerator • Supports SHA-160, SHA-224, SHA-256, SHA-384, SHA-512 and and SHA-512/t • Supports SHA3-224, SHA3-256, SHA3-384, SHA3-512,

	SHAKE128 and SHAKE256 • Compliant with FIPS 180/180-2
keyed-Hash Message Authentication Code (HMAC)	• Hardware HMAC accelerator • Supports HMAC-SHA-160, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 • Supports HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, and HMAC-SHA3-512 • Compliant with FIPS 180/180-2
Pseudo Random Number Generator (PRNG)	• Hardware PRNG accelerator • Supports take seed from TRNG
True Random Number Generator (TRNG)	• 800 random bits per second
Key store	• Supports programming interface for key management • Supports multiple key size from 128 bits to 4096 bits • Supports 32 SRAM keys, 32 Flash keys and 8 OTP keys at most • Supports crypto engine access key in key store directly • Supports ECDH operation with ECC and PRNG engine • Supports to store middle data for RSA CRT and SCAP mode • Supports revoke operation for each key • Supports erase key in SRAM/Flash and revoke key in OTP while tamper detected • Supports auto verify function • Supports lock function for OTP keys • Supports data remanence prevention at SRAM

3 PARTS INFORMATION

3.1 Package Type

Package is Halogen-free, RoHS-compliant and TSCA-compliant.

3.1.1 M463/M467 Series

Part No.	QFN48 (5x5mm)	LQFP48 (7x7mm)	LQFP64 (7x7mm)	LQFP128 (14x14mm)	LQFP144 (20x20mm)	LQFP176 (24x24mm)
M463	M463YGCAE	M463LGCAE	M463SGCAE	M463KGCAE		
M467			M467SJHAN	M467KJHAN	M467JJHAN	M467HJHAN

Part No.	LQFP48 EPAD (7x7mm)	LQFP64 EPAD (7x7mm)	LQFP128 EPAD (14x14mm)	LQFP144 EPAD (20x20mm)	LQFP176 EPAD (24x24mm)
M463	M463L2GCAC	M463S2GCAC	M463K2GCAC		
M467		M467S2JHAE	M467K2JHAE	M467J2JHAE	M467H2JHAE

3.2 M463/M467 Series Naming Rule

M4	60	H	J	H	A	E
Core	Series	Package	Flash Size	SRAM Size	Revision	Temperature
Cortex-M4F	63: CAN FD/ USB HS 67: Ethernet/ Crypto	Y: QFN48 (5x5 mm) L: LQFP48 (7x7 mm) L2: LQFP48 EPAD (7x7 mm) S: LQFP64 (7x7 mm) S2: LQFP64 EPAD (7x7 mm) K: LQFP128 (14x14 mm) K2: LQFP128 EPAD (14x14 mm) J: LQFP144 (20x20 mm) J2: LQFP144 EPAD (20x20 mm) H: LQFP176 (24x24 mm) H2: LQFP176 EPAD (24x24 mm)	J: 1024 Kbytes G: 256 Kbytes	H: 512 Kbytes C: 128 Kbytes	A	N:-40˚C ~ 85˚C E:-40˚C ~ 105˚C C:-40˚C ~ 125˚C

3.3 M463/M467 Series Selection Guide

3.3.1 M463 Series Selection Guide

3.3.1.1 M463 Series

PART NUMBER		M463			
		KGCAE	SGCAE	LGCAE	YGCAE
System Frequency (MHz)		200			
Flash (KB)		256			
SRAM (KB)		128			
LDROM (KB)		8			
XOM (regions)		4			
PDMA		16-ch			
I/O		100	44	33	33
RTC (V_{BAT})		√			
32-bit Timer		4			
16-bit EPWM		12			
16-bit BPWM		12			
Connectivity	UART	8			
	QSPI	2			
	SPI/I^2S	4			
	I^2C	5			
	CAN FD	2			
	PSIO	1			
	USCI	1			
	SD Host	1			
	ISO-7816-3	1			
USB High Speed OTG with PHY		√			
LCD Parallel Data Bus (External Bus Interface)		√			
EQEI		2			
ECAP		2			
KPI		6x8			
12-bit ADC		16	16	12	12
Analog Comparator		2			
Crypto		AES-256			
TRNG		√			
Operating Temperature		-40˚C ~ 105˚C			
Package		LQFP 128 (14x14mm)	LQFP 64 (7x7mm)	LQFP 48 (7x7mm)	QFN 48 (5x5mm)

3.3.1.2 M463 Series with EPAD

PART NUMBER		M463		
		K2GCAC	S2GCAC	L2GCAC
System Frequency (MHz)		200		
Flash (KB)		256		
SRAM (KB)		128		
LDROM (KB)		8		
XOM (regions)		4		
PDMA		16-ch		
I/O		100	44	33
RTC (V_{BAT})		√		
32-bit Timer		4		
16-bit EPWM		12		
16-bit BPWM		12		
Connectivity	UART	8		
	QSPI	2		
	SPI/I^2S	4		
	I^2C	5		
	CAN FD	2		
	PSIO	1		
	USCI	1		
	SD Host	1		
	ISO-7816-3	1		
USB High Speed OTG with PHY		√		
LCD Parallel Data Bus (External Bus Interface)		√		
EQEI		2		
ECAP		2		
KPI		6x8		
12-bit ADC		16	16	12
Analog Comparator		2		
Crypto		AES-256		
TRNG		√		
Operating Temperature		-40˚C ~ 125˚C		
Package		LQFP 128 EPAD (14x14mm)	LQFP 64 EPAD (7x7mm)	LQFP 48 EPAD (7x7mm)

3.3.2 M467 Series Selection Guide

3.3.2.1 M467 Series

PART NUMBER		M467			
		HJHAN	JJHAN	KJHAN	SJHAN
System Frequency (MHz)		200			
Flash (KB)		1024			
SRAM (KB)		512			
LDROM (KB)		8			
XOM (regions)		4			
PDMA		2 set, each with 16-ch			
I/O		146	114	100	44
RTC (V_{BAT})		√			
32-bit Timer		4			
16-bit EPWM		12			
16-bit BPWM		12			
Connectivity	UART	10			9
	QSPI	2			
	SPI/I^2S	4			
	SPI Master	1			
	I^2S	2			
	I^2C	5			
	CAN FD	4			
	PSIO	1			
	USCI	1			
	SD Host	2			
	ISO-7816-3	3			
USB High Speed OTG with PHY		√			
USB Full Speed OTG with PHY		√			
LCD Parallel Data Bus (External Bus Interface)		√			
HyperBus Interface		√			-
EQEI		4			
ECAP		4			
KPI		6x8			
12-bit ADC		28			20
12-bit DAC		2			
Analog Comparator		4			

Crypto	AES-256, ECC-571, SHA-512, HMAC-512, RSA-4096, SM2			
TRNG	√			
Ethernet 10/100 Mac	√			
Camera Capture Interface	√			
Operating Temperature	-40˚C ~ 85˚C			
Package	LQFP 176 (24x24mm)	LQFP 144 (20x20mm)	LQFP 128 (14x14mm)	LQFP 64 (7x7mm)

3.3.2.2 M467 Series with EPAD

PART NUMBER		**M467**			
		H2JHAE	**J2JHAE**	**K2JHAE**	**S2JHAE**
System Frequency (MHz)		200			
Flash (KB)		1024			
SRAM (KB)		512			
LDROM (KB)		8			
XOM (regions)		4			
PDMA		2 set, each with 16-ch			
I/O		146	114	100	44
RTC (V_{BAT})		√			
32-bit Timer		4			
16-bit EPWM		12			
16-bit BPWM		12			
Connectivity	**UART**	10			9
	QSPI	2			
	SPI/I²S	4			
	SPI Master	1			
	I²S	2			
	I²C	5			
	CAN FD	4			
	PSIO	1			
	USCI	1			
	SD Host	2			
	ISO-7816-3	3			
USB High Speed OTG with PHY		√			
USB Full Speed OTG with PHY		√			
LCD Parallel Data Bus (External Bus Interface)		√			

HyperBus Interface	√			-
EQEI	4			
ECAP	4			
KPI	6x8			
12-bit ADC	28			20
12-bit DAC	2			
Analog Comparator	4			
Crypto	AES-256, ECC-571, SHA-512, HMAC-512, RSA-4096, SM2			
TRNG	√			
Ethernet 10/100 Mac	√			
Camera Capture Interface	√			
Operating Temperature	-40˚C ~ 105˚C			
Package	LQFP 176 EPAD (24x24mm)	LQFP 144 EPAD (20x20mm)	LQFP 128 EPAD (14x14mm)	LQFP 64 EPAD (7x7mm)

3.4 M463/M467 Series Features Comparison Table

Section	Sub-section	M463	M467
System Manager	Figure 6.2-11 SRAM Memory Organization for 512 Kbyte Device	-	●
	Figure 6.2-12 SRAM Memory Organization for 128 Kbyte Device	●	-
FMC	6.5.4.2 Dual Bank Architecture	-	●
	6.5.4.4 Physical and Virtual Address Concept	-	●
	6.5.4.5 APROM Reboot Address Operation Model Selection	-	●
Cryptographic Accelerator	6.40 Cryptographic Accelerator (CRYPTO)	-	●
	6.41 Cryptographic Accelerator for M463 Series (CRYPTO_M463)	●	-
RTC	6.16.5.12 Spare Registers and Tamper Detector for M463 Series	●	-
	6.16.5.13 Spare Registers and Tamper Detector for M467 Series	-	●

4 PIN CONFIGURATION

4.1 Pin Configuration

Users can find pin configuration information in the M463/M467 series Multi-function Pin diagram sections or by using NuTool - PinConfigure. The NuTool - PinConfigure contains all NuMicro Family chip series with all part number, and helps users configure GPIO multi-function correctly and handily.

4.1.1 M463 Series Pin Diagram

4.1.1.1 M463 Series QFN48-Pin Diagram

Corresponding Part Number: M463YGCAE

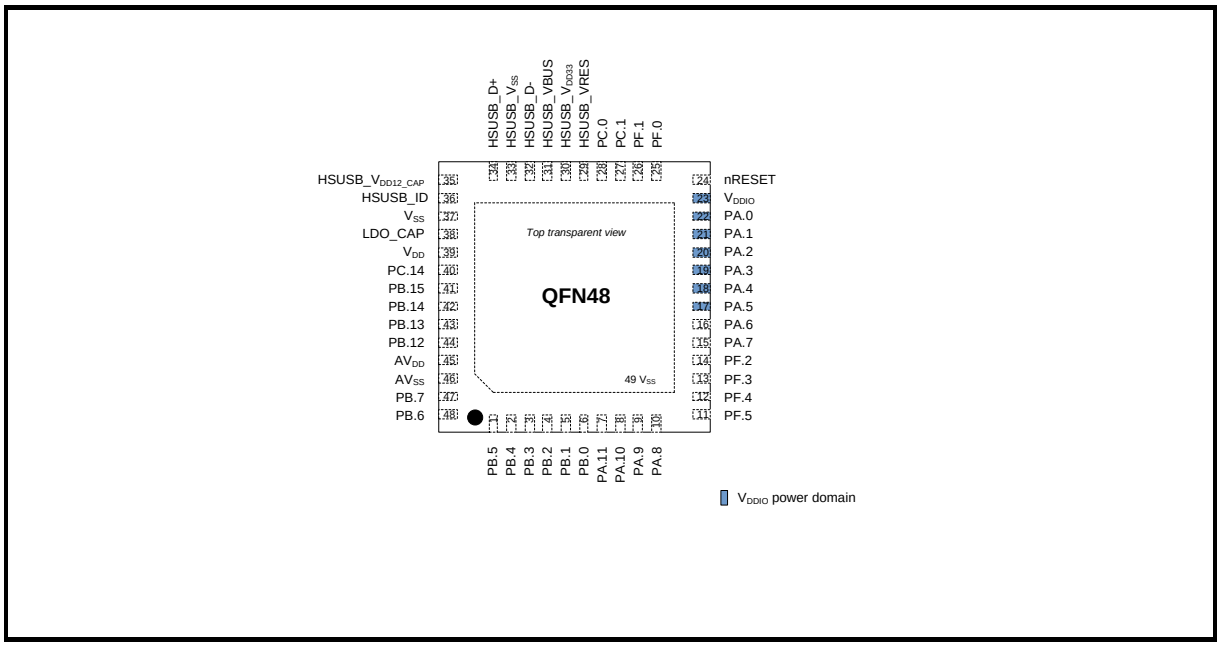


Figure 4.1-1 M463 Series QFN48-Pin Diagram

4.1.1.2 M463 Series LQFP48-Pin Diagram

Corresponding Part Number: M463LGCAE

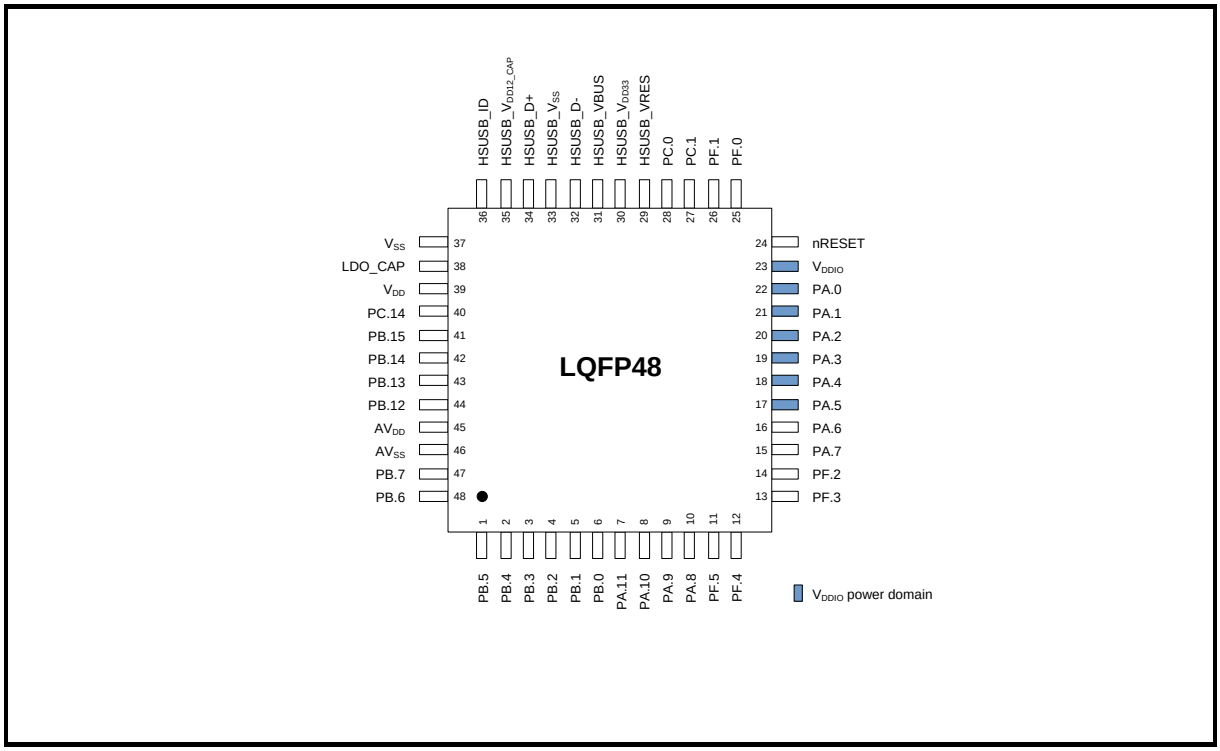


Figure 4.1-2 M463 Series LQFP48-Pin Diagram

4.1.1.3 M463 Series LQFP64-Pin Diagram

Corresponding Part Number: M463SGCAE

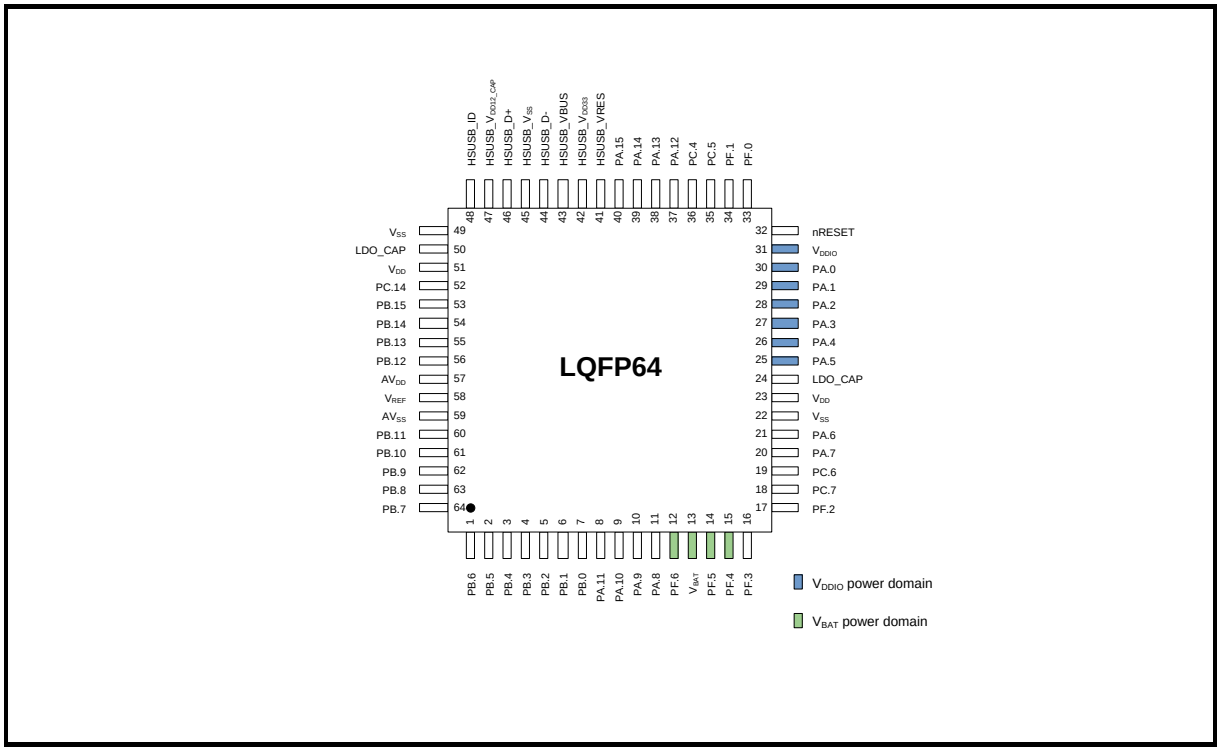


Figure 4.1-3 M463 Series LQFP64-Pin Diagram

4.1.1.4 M463 Series LQFP128-Pin Diagram

Corresponding Part Number: M463KGCAE

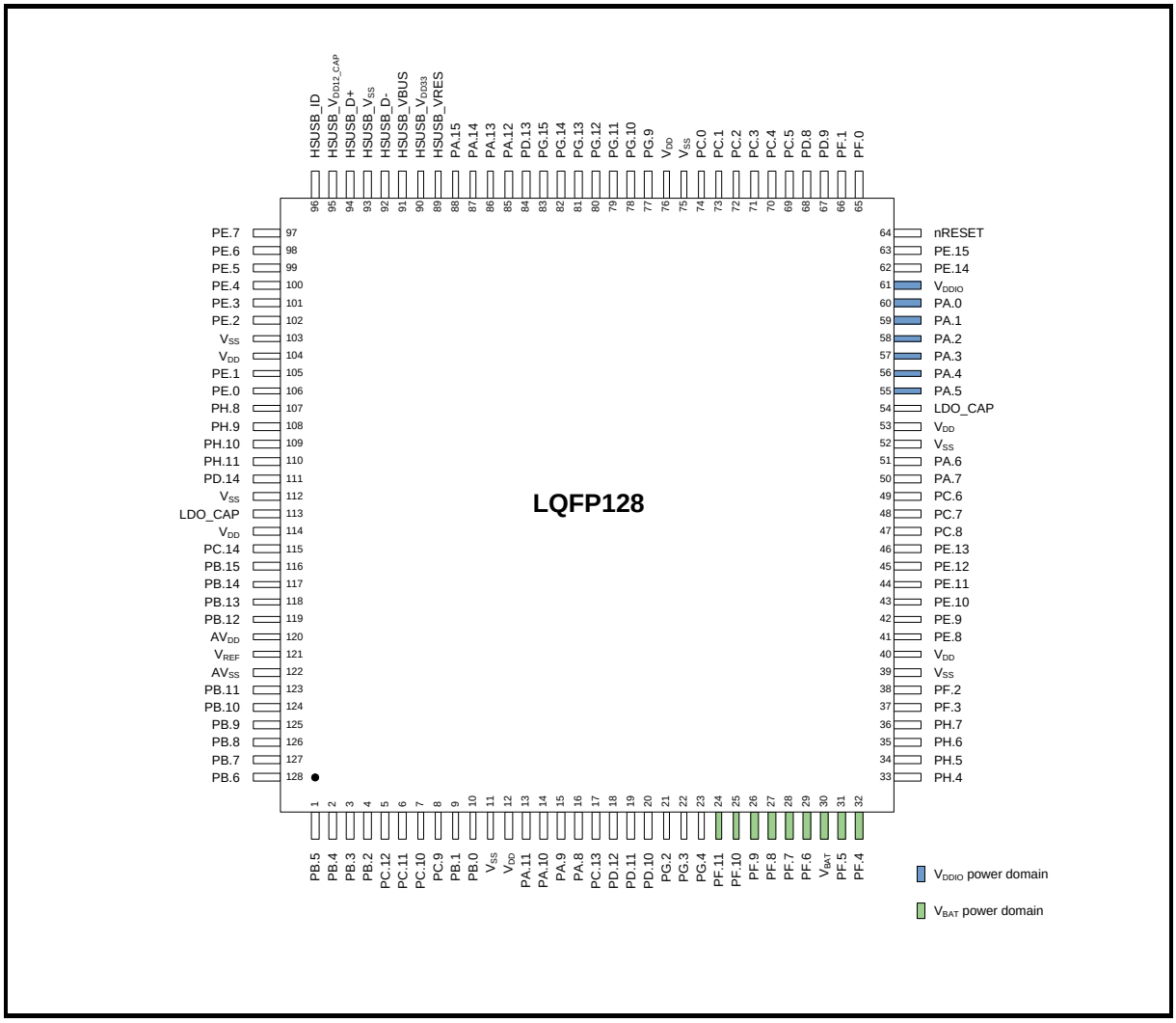


Figure 4.1-4 M463 Series LQFP128-Pin Diagram

4.1.1.5 M463 Series LQFP48 EPAD Pin Diagram

Corresponding Part Number: M463L2GCAC

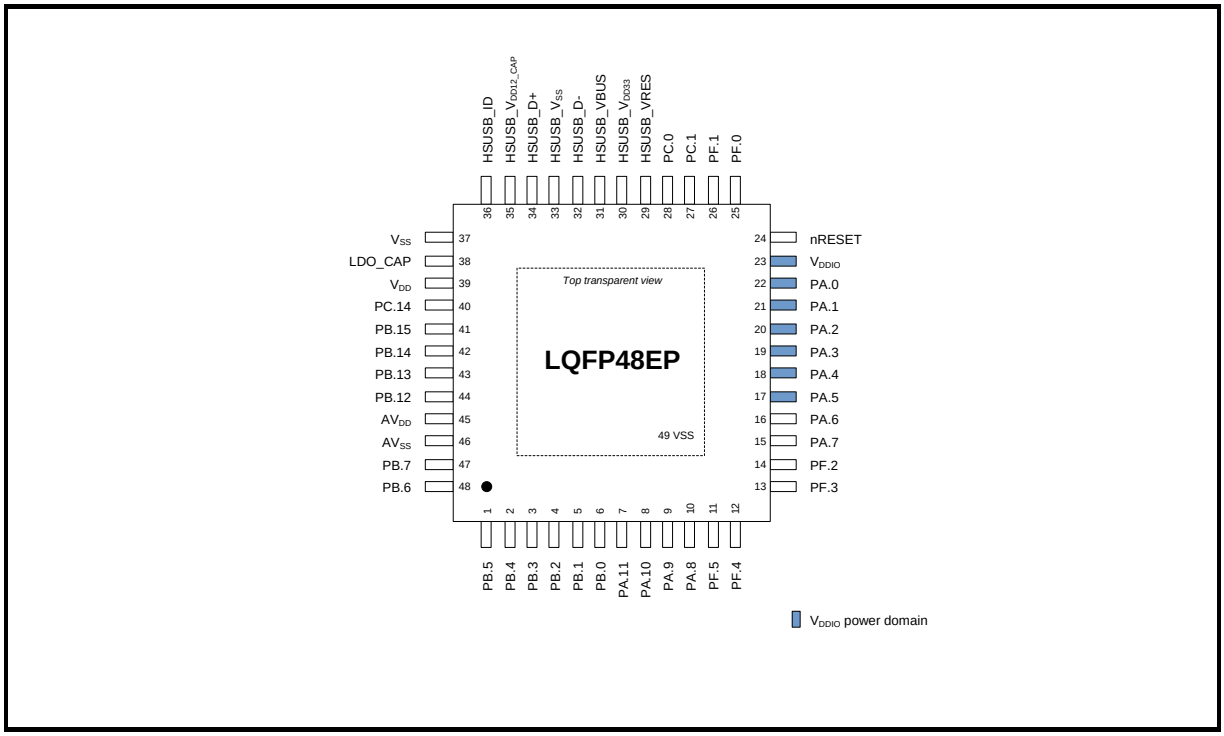


Figure 4.1-5 M463 Series LQFP48 EPAD Pin Diagram

4.1.1.6 M463 Series LQFP64 EPAD Pin Diagram

Corresponding Part Number: M463S2GCAC

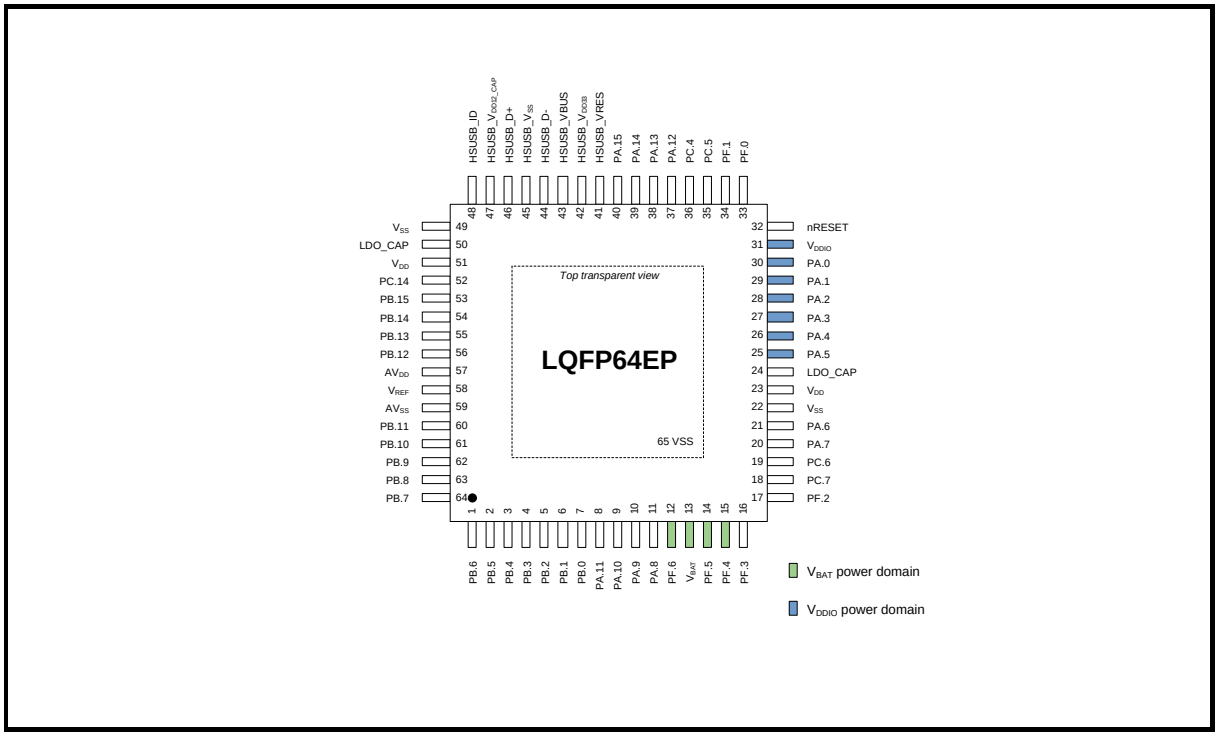


Figure 4.1-6 M463 Series LQFP64 EPAD Pin Diagram

4.1.1.7 M463 Series LQFP128 EPAD Pin Diagram

Corresponding Part Number: M463K2GCAC

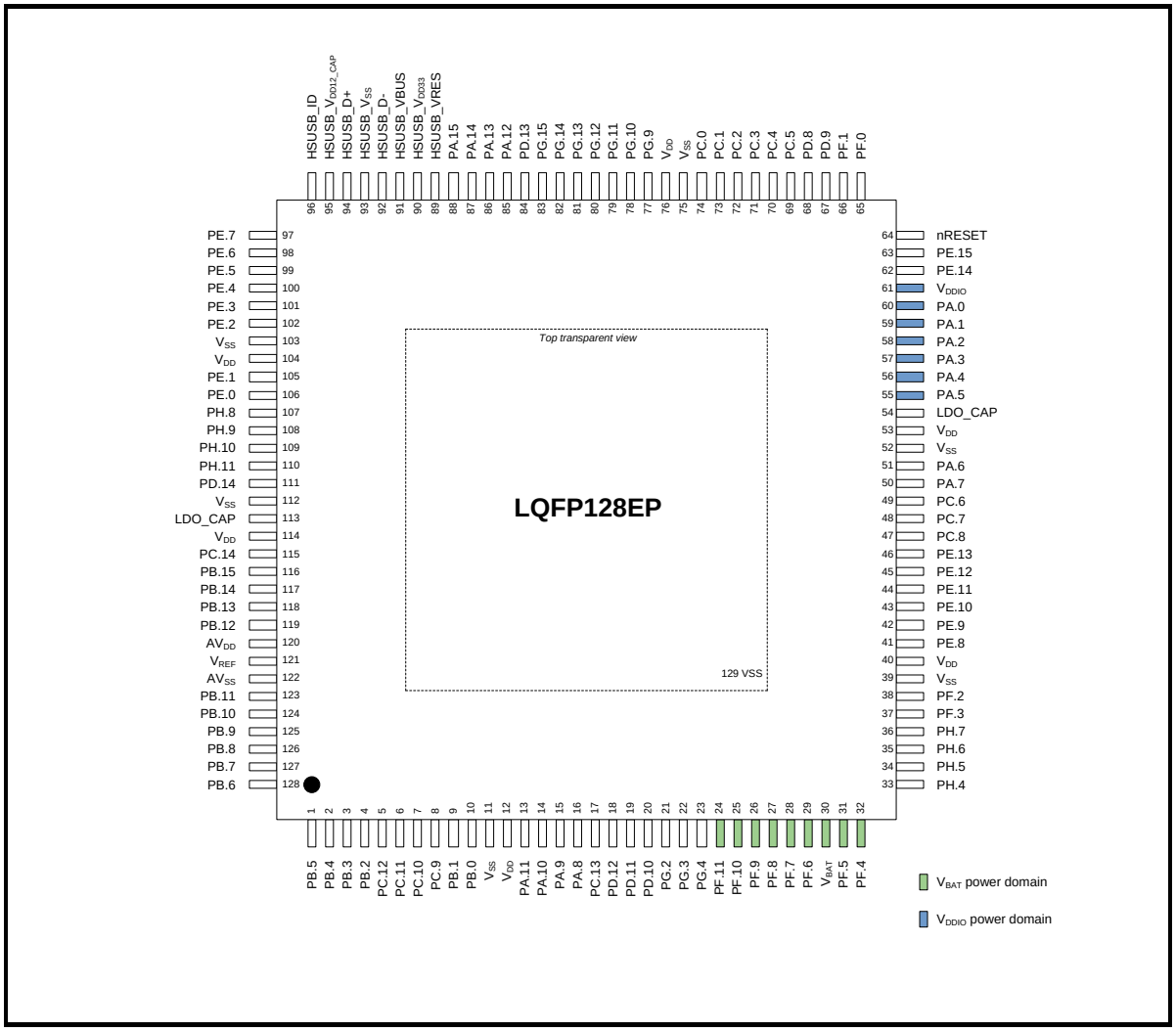


Figure 4.1-7 M463 Series LQFP128 EPAD Pin Diagram

4.1.2 M467 Series Pin Diagram

4.1.2.1 M467 Series LQFP64-Pin Diagram

Corresponding Part Number: M467SJHAN

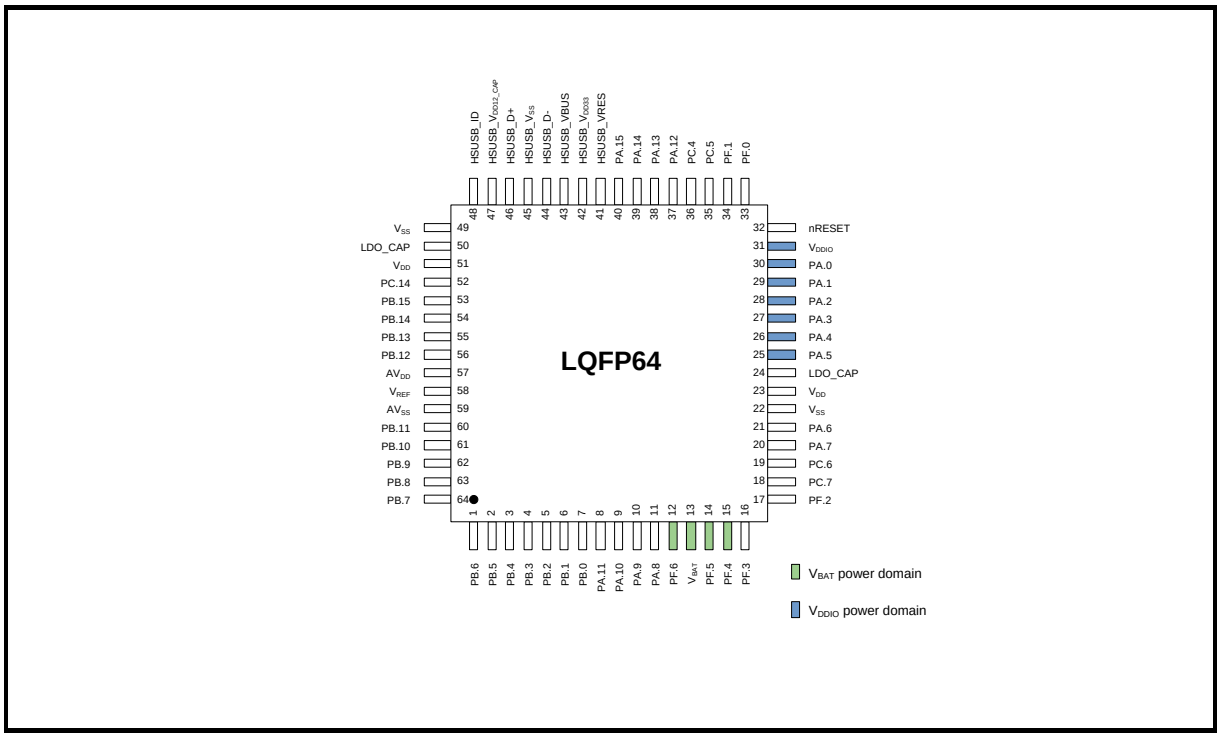


Figure 4.1-8 M467 Series LQFP64-Pin Diagram

4.1.2.2 M467 Series LQFP128-Pin Diagram

Corresponding Part Number: M467KJHAN

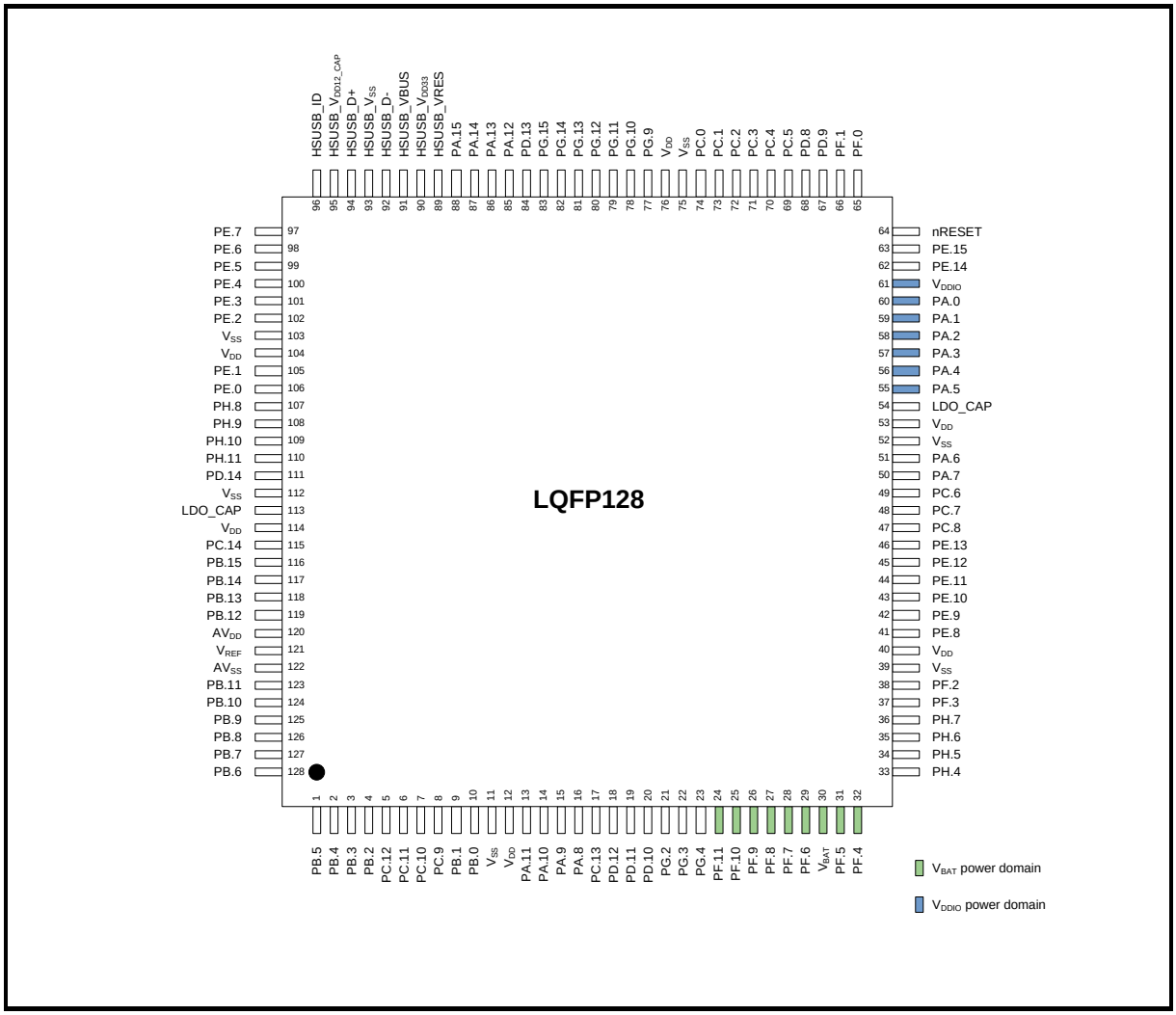


Figure 4.1-9 M467 Series LQFP128-Pin Diagram

4.1.2.3 M467 Series LQFP144-Pin Diagram

Corresponding Part Number: M467JJHAN

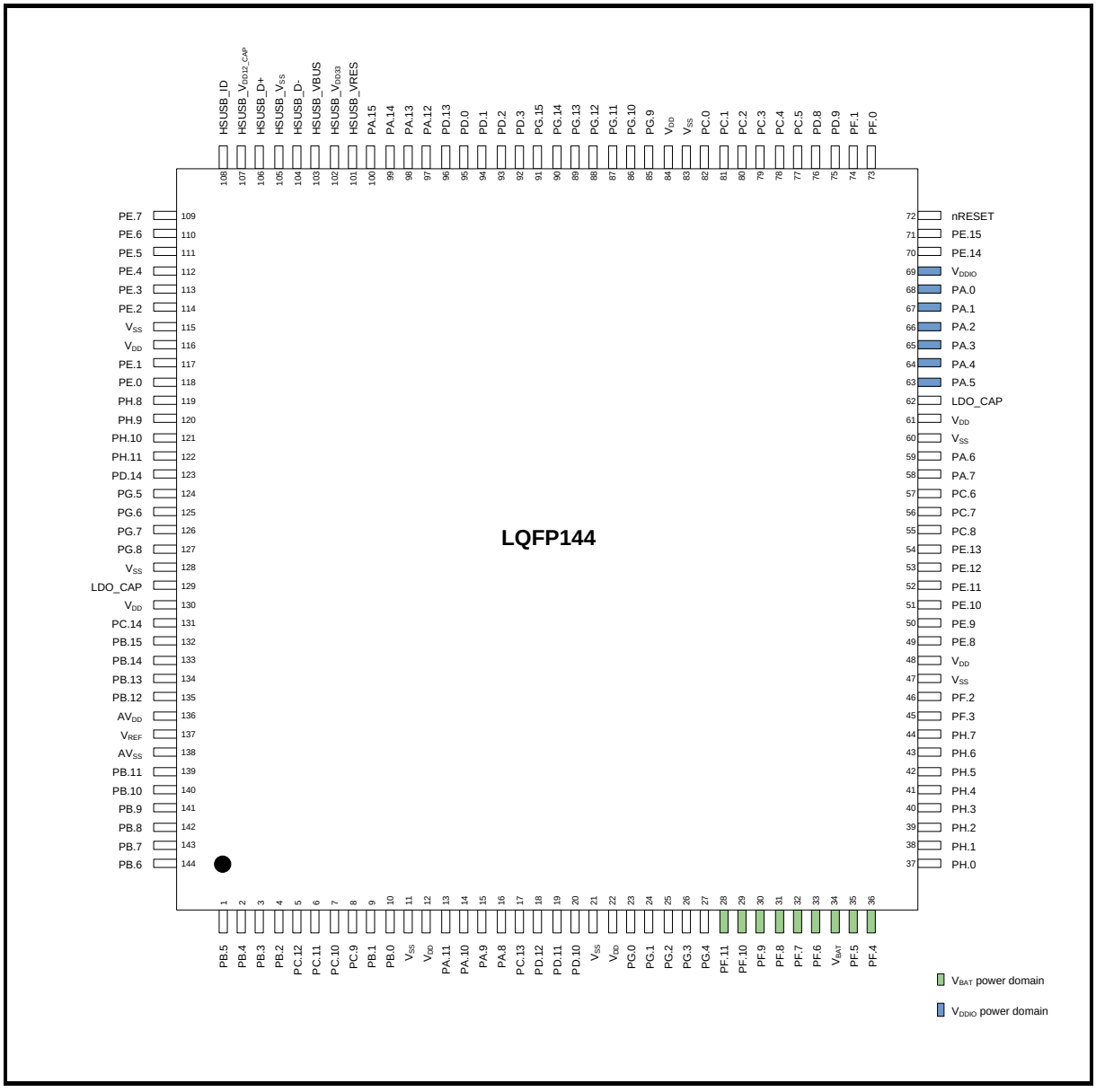


Figure 4.1-10 M467 Series LQFP144-Pin Diagram

4.1.2.4 M467 Series LQFP176-Pin Diagram

Corresponding Part Number: M467HJHAN

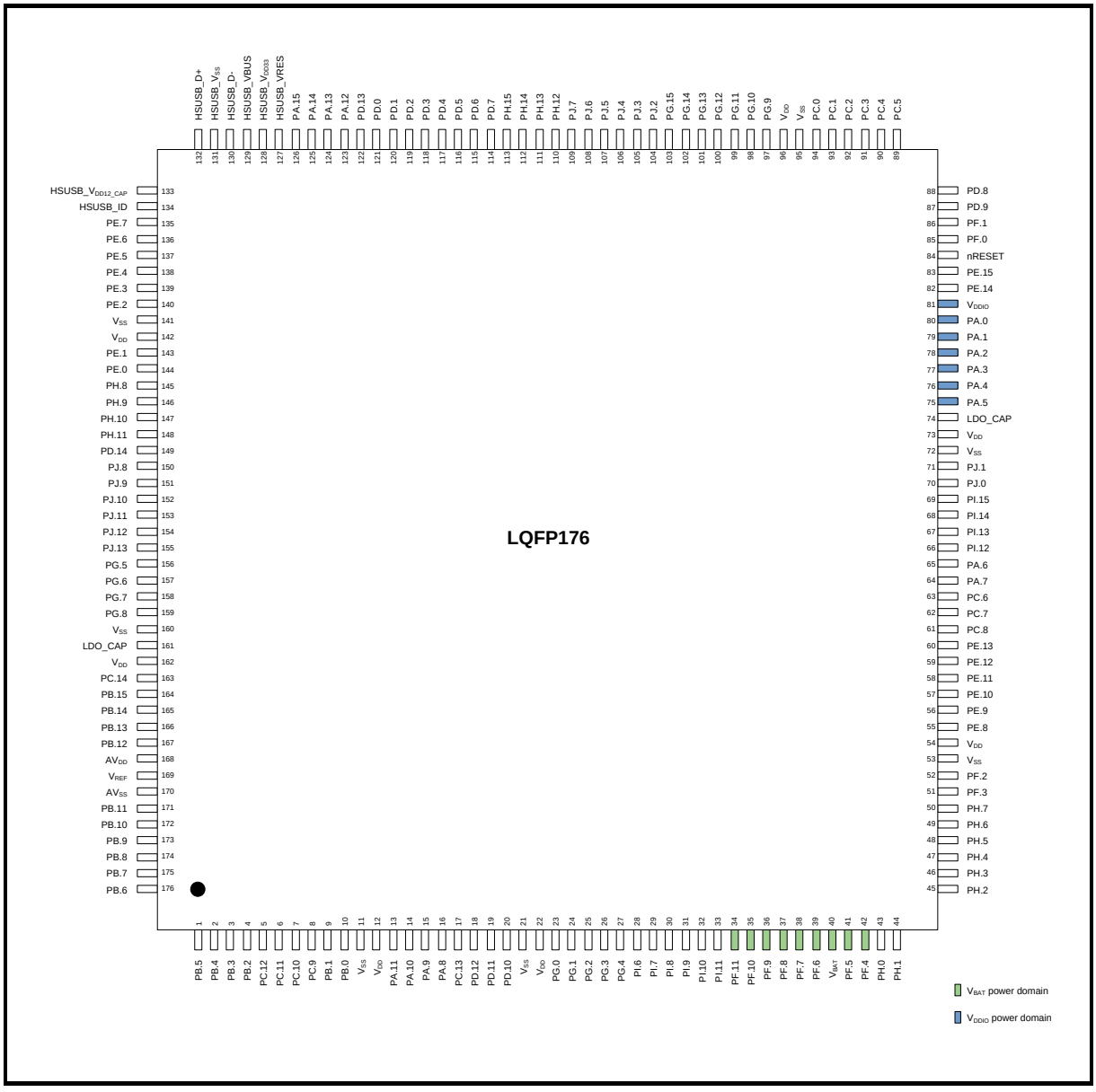


Figure 4.1-11 M467 Series LQFP176-Pin Diagram

4.1.2.5 M467 Series LQFP64 EPAD Pin Diagram

Corresponding Part Number: M467S2JHAE

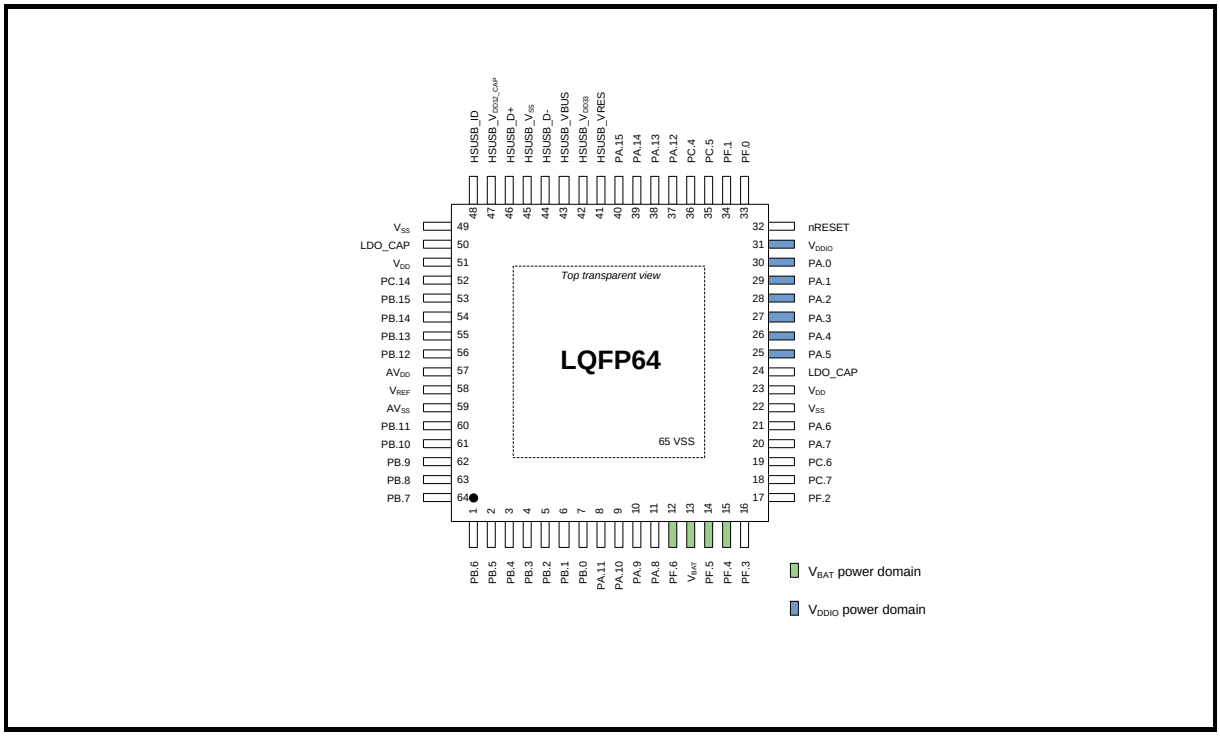


Figure 4.1-12 M467 Series LQFP64 EPAD Pin Diagram

4.1.2.6 M467 Series LQFP128 EPAD Pin Diagram

Corresponding Part Number: M467K2JHAE

Note: It recommends to connect EPAD to V_{SS}.

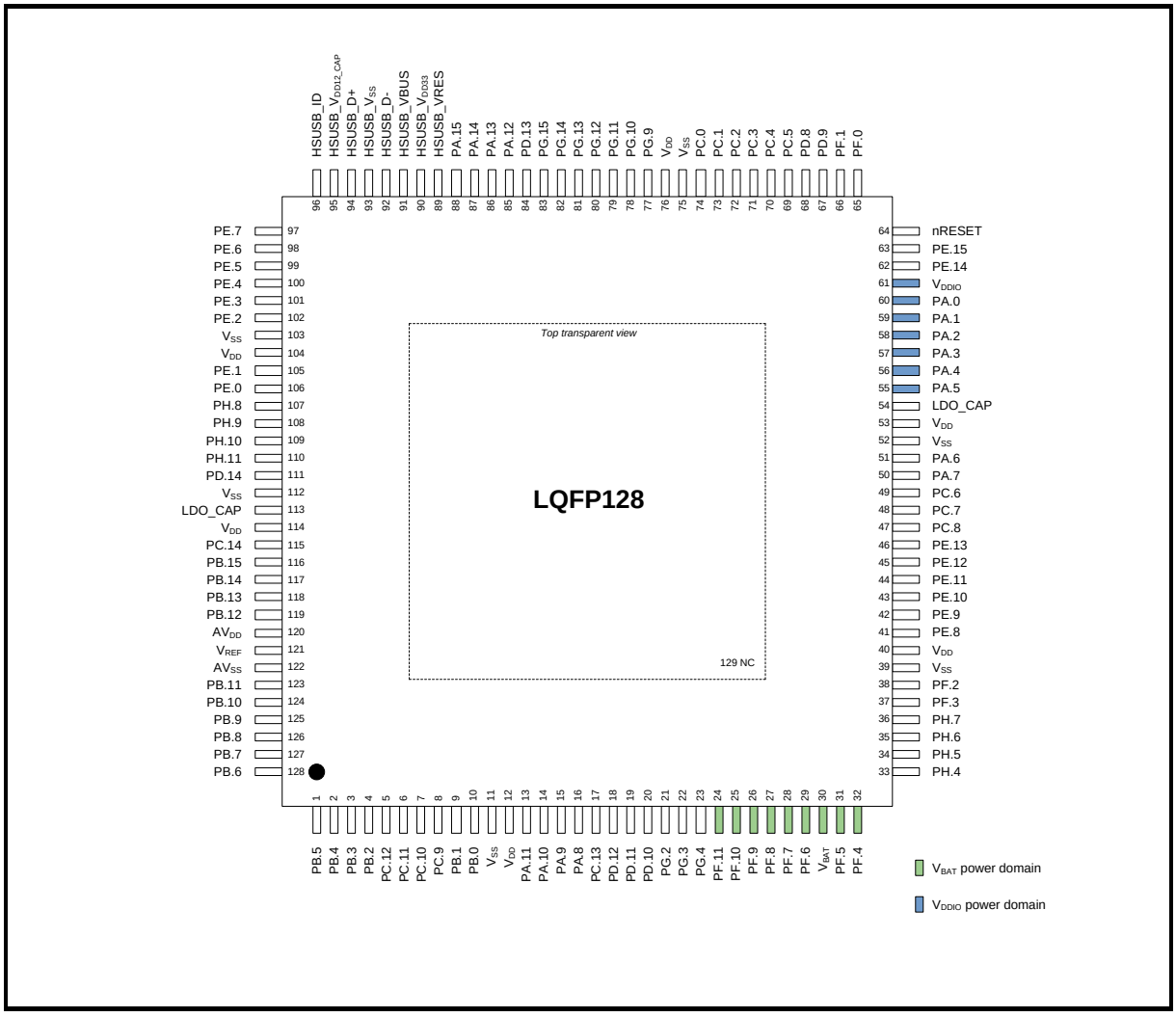


Figure 4.1-13 M467 Series LQFP128 EPAD Pin Diagram

4.1.2.7 M467 Series LQFP144 EPAD Pin Diagram

Corresponding Part Number: M467J2JHAE

Note: It recommends to connect EPAD to V_{SS}.

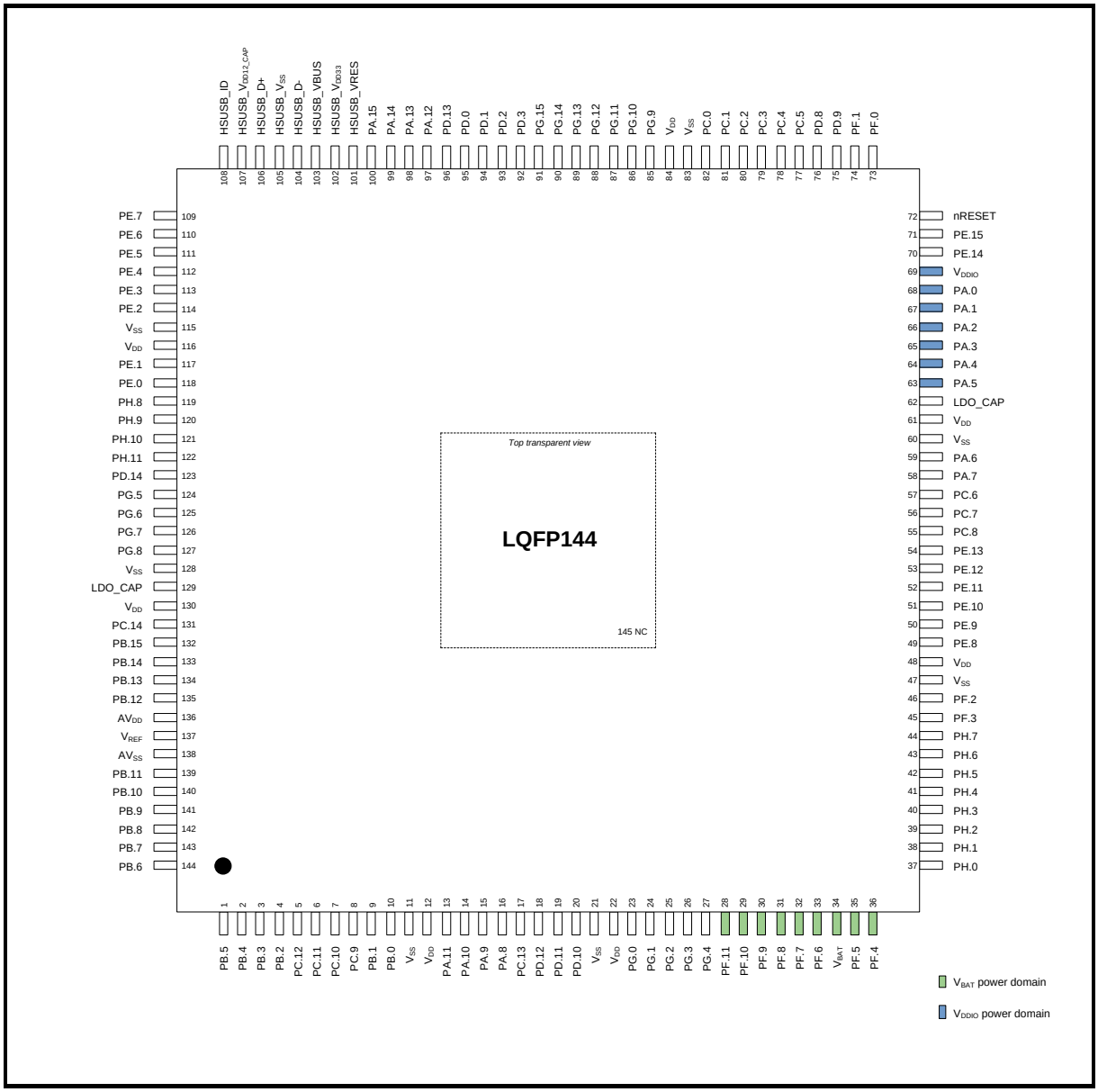


Figure 4.1-14 M467 Series LQFP144 EPAD Pin Diagram

4.1.2.8 M467 Series LQFP176 EPAD Pin Diagram

Corresponding Part Number: M467H2JHAE

Note: It recommends to connect EPAD to V_{SS}.

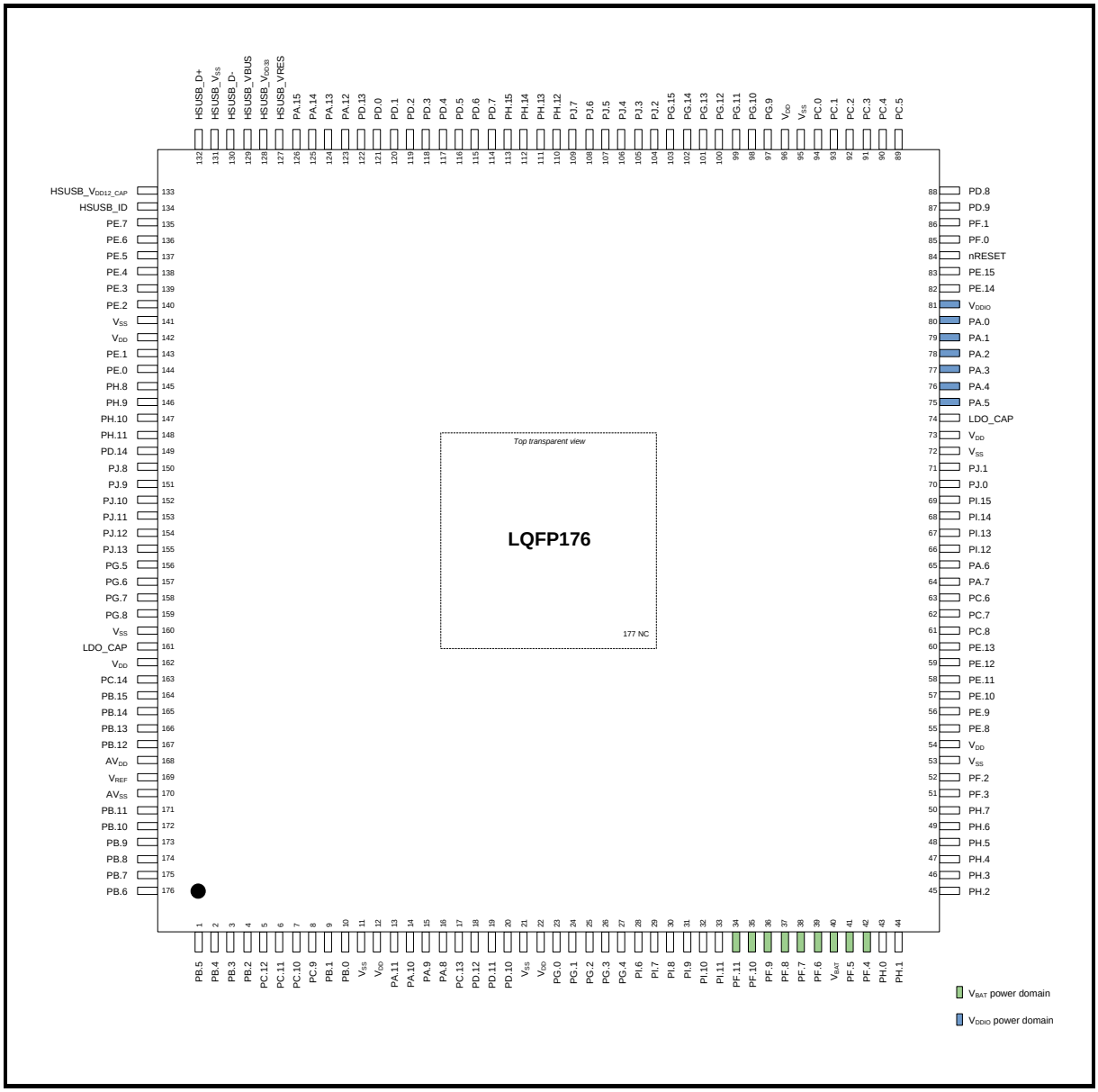


Figure 4.1-15 M467 Series LQFP176 EPAD Pin Diagram

4.1.3 M463 Series Multi-function Pin Diagram

4.1.3.1 M463 Series QFN48-Pin Multi-function Pin Diagram

Corresponding Part Number: M463YGCAE

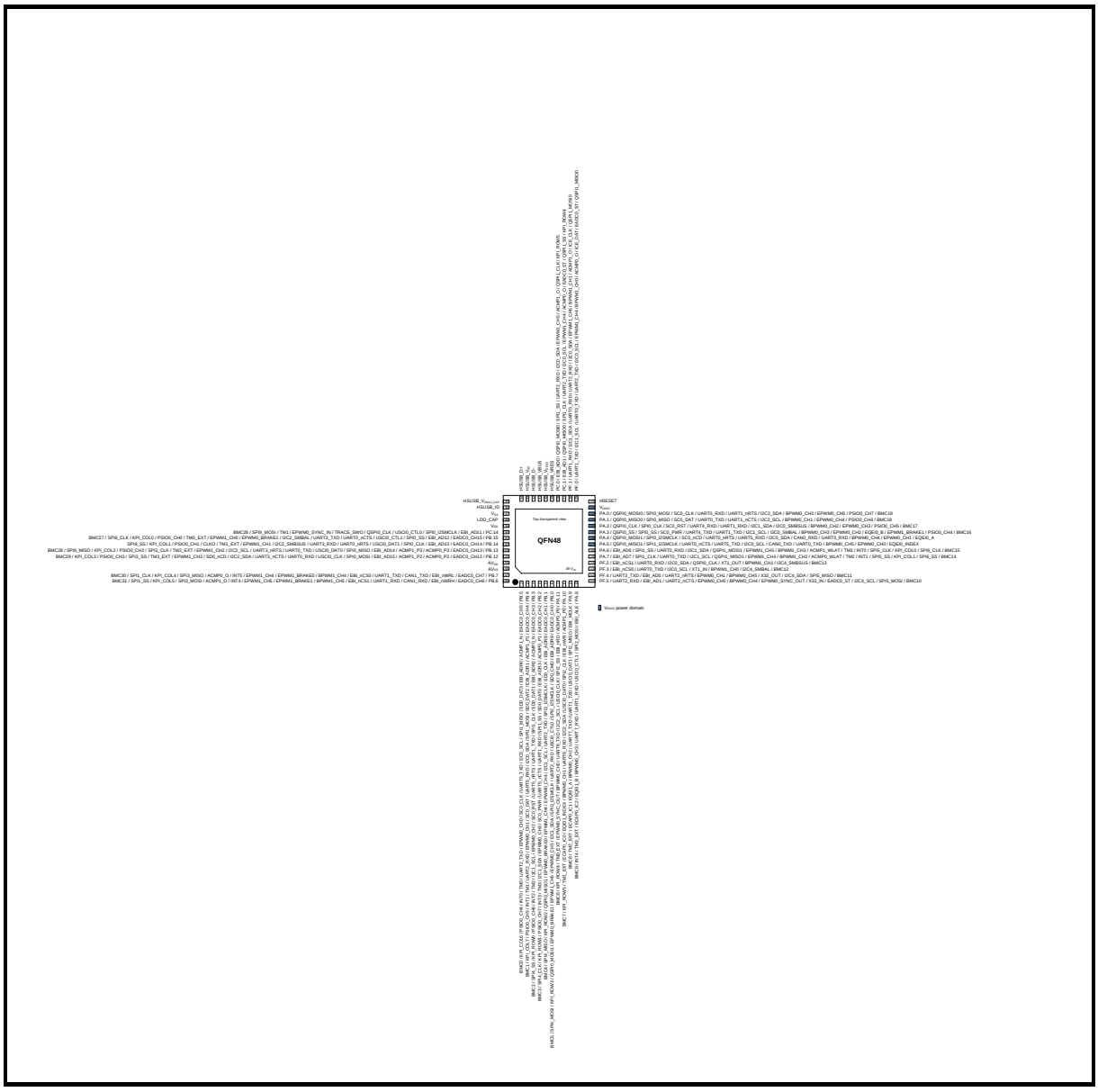


Figure 4.1-16 M463YGCAE Multi-function Pin Diagram

Pin	Type	M463YGCAE Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS

Pin	Type	M463YGCAE Pin Function
		/ SC0_PWR / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1 / KPI_ROW2
6	I/O	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1 / KPI_ROW3
7	I/O	PA.11 / ACMP0_P0 / EBI_nRD / SPI2_SS / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / KPI_ROW4
8	I/O	PA.10 / ACMP1_P0 / EBI_nWR / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / TM1_EXT / KPI_ROW5
9	I/O	PA.9 / EBI_MCLK / SPI2_MISO / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / TM2_EXT
10	I/O	PA.8 / EBI_ALE / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / TM3_EXT / INT4
11	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL
12	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / I2C4_SDA
13	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL
14	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS
15	I/O	PA.7 / EBI_AD7 / SPI1_CLK / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
16	I/O	PA.6 / EBI_AD6 / SPI1_SS / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
17	I/O	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / UART0_nCTS / UART5_TXD / I2C0_SCL / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
18	I/O	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
19	I/O	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
20	I/O	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / PSIO0_CH5
21	I/O	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / PSIO0_CH6
22	I/O	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / PSIO0_CH7
23	P	V_{DDIO}
24	I	nRESET
25	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / UART2_TXD / I2C0_SCL / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST / QSPI1_MISO0
26	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / UART2_RXD / I2C0_SDA / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / QSPI1_MOSI0
27	I/O	PC.1 / EBI_AD1 / QSPI0_MISO0 / SPI1_CLK / UART2_TXD / I2C0_SCL / EPWM1_CH4 / ACMP0_O / EADC0_ST / QSPI1_SS / KPI_ROW4
28	I/O	PC.0 / EBI_AD0 / QSPI0_MOSI0 / SPI1_SS / UART2_RXD / I2C0_SDA / EPWM1_CH5 / ACMP1_O /

Pin	Type	M463YGCAE Pin Function
		QSPI1_CLK / KPI_ROW5
29	A	HSUSB_VRES
30	P	HSUSB_VDD33
31	I/O	HSUSB_VBUS
32	A	HSUSB_D-
33	P	HSUSB_VSS
34	A	HSUSB_D+
35	A	HSUSB_VDD12_CAP
36	I	HSUSB_ID
37	P	V_{SS}
38	A	LDO_CAP
39	P	V_{DD}
40	I/O	PC.14 / EBI_AD11 / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / TM1
41	I/O	PB.15 / EADC0_CH15 / EBI_AD12 / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM0_BRAKE1 / EPWM1_CH0 / TM0_EXT / PSIO0_CH0 / KPI_COL0
42	I/O	PB.14 / EADC0_CH14 / EBI_AD13 / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM1_CH1 / TM1_EXT / CLKO / PSIO0_CH1 / KPI_COL1
43	I/O	PB.13 / EADC0_CH13 / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / PSIO0_CH2 / KPI_COL2
44	I/O	PB.12 / EADC0_CH12 / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / PSIO0_CH3 / KPI_COL3
45	P	AV_{DD}
46	P	AV_{SS}
47	I/O	PB.7 / EADC0_CH7 / EBI_nWRL / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / ACMP0_O / KPI_COL4 / SPI1_CLK
48	I/O	PB.6 / EADC0_CH6 / EBI_nWRH / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / ACMP1_O / KPI_COL5 / SPI1_SS
49	P	V_{SS}

Table 4.1-1 M463YGCAE Multi-function Pin Table

4.1.3.2 M463 Series LQFP48-Pin Multi-function Pin Diagram

Corresponding Part Number: M463LGCAE

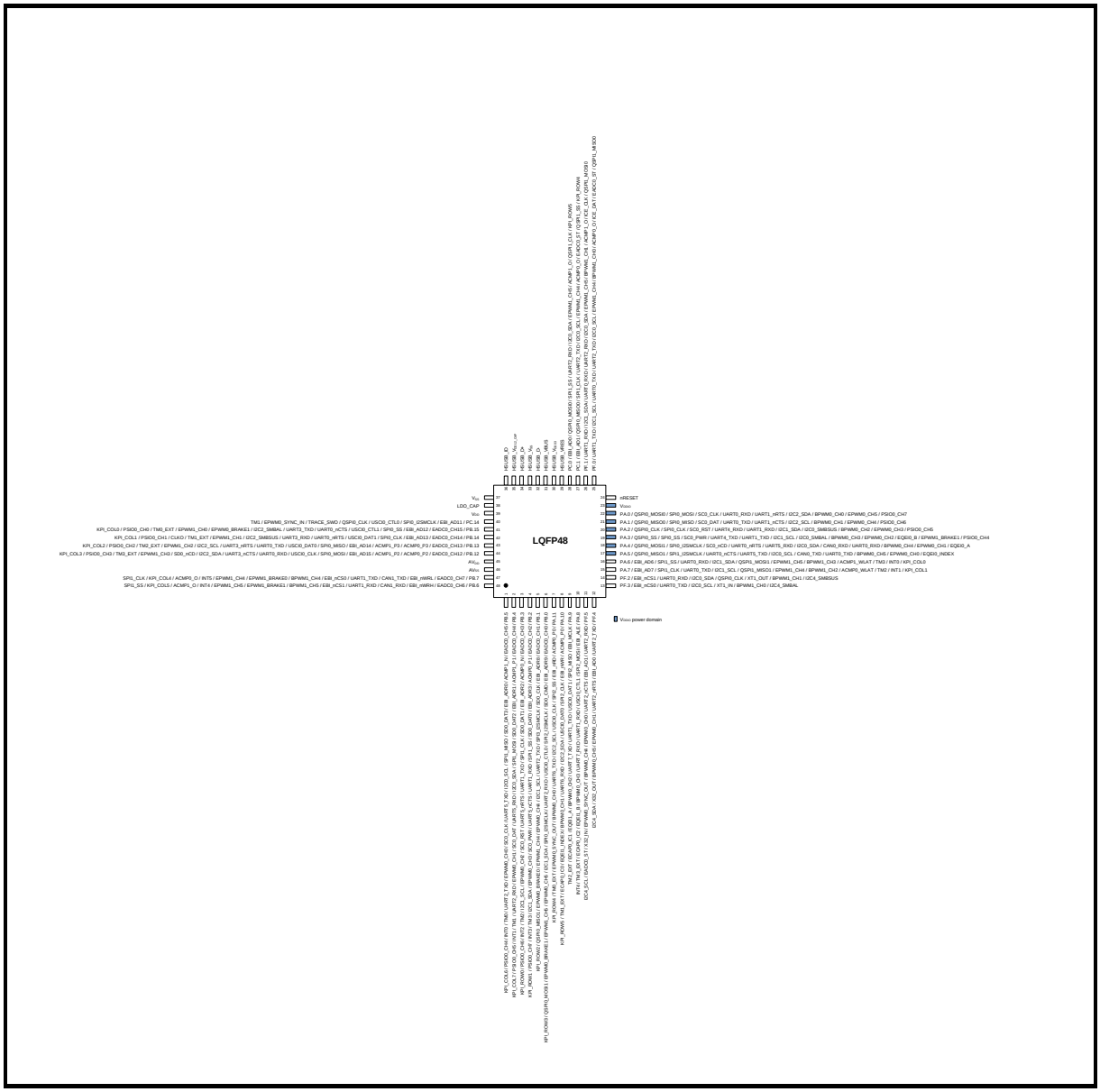


Figure 4.1-17 M463LGCAE Multi-function Pin Diagram

Pin	Type	M463LGCAE Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1

Pin	Type	M463LGCAE Pin Function
5	I/O	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1 / KPI_ROW2
6	I/O	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1 / KPI_ROW3
7	I/O	PA.11 / ACMP0_P0 / EBI_nRD / SPI2_SS / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / KPI_ROW4
8	I/O	PA.10 / ACMP1_P0 / EBI_nWR / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / TM1_EXT / KPI_ROW5
9	I/O	PA.9 / EBI_MCLK / SPI2_MISO / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / TM2_EXT
10	I/O	PA.8 / EBI_ALE / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / TM3_EXT / INT4
11	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL
12	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / I2C4_SDA
13	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL
14	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS
15	I/O	PA.7 / EBI_AD7 / SPI1_CLK / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
16	I/O	PA.6 / EBI_AD6 / SPI1_SS / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
17	I/O	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
18	I/O	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
19	I/O	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
20	I/O	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / PSIO0_CH5
21	I/O	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / PSIO0_CH6
22	I/O	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / PSIO0_CH7
23	P	V_{DDIO}
24	I	nRESET
25	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / UART2_TXD / I2C0_SCL / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST / QSPI1_MISO0
26	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / UART2_RXD / I2C0_SDA / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / QSPI1_MOSI0
27	I/O	PC.1 / EBI_AD1 / QSPI0_MISO0 / SPI1_CLK / UART2_TXD / I2C0_SCL / EPWM1_CH4 / ACMP0_O / EADC0_ST / QSPI1_SS / KPI_ROW4
28	I/O	PC.0 / EBI_AD0 / QSPI0_MOSI0 / SPI1_SS / UART2_RXD / I2C0_SDA / EPWM1_CH5 / ACMP1_O / QSPI1_CLK / KPI_ROW5

Pin	Type	M463LGCAE Pin Function
29	A	HSUSB_VRES
30	P	HSUSB_VDD33
31	I/O	HSUSB_VBUS
32	A	HSUSB_D-
33	P	HSUSB_VSS
34	A	HSUSB_D+
35	A	HSUSB_VDD12_CAP
36	I	HSUSB_ID
37	P	V_{SS}
38	A	LDO_CAP
39	P	V_{DD}
40	I/O	PC.14 / EBI_AD11 / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / TM1
41	I/O	PB.15 / EADC0_CH15 / EBI_AD12 / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM0_BRAKE1 / EPWM1_CH0 / TM0_EXT / PSIO0_CH0 / KPI_COL0
42	I/O	PB.14 / EADC0_CH14 / EBI_AD13 / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM1_CH1 / TM1_EXT / CLKO / PSIO0_CH1 / KPI_COL1
43	I/O	PB.13 / EADC0_CH13 / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / PSIO0_CH2 / KPI_COL2
44	I/O	PB.12 / EADC0_CH12 / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / PSIO0_CH3 / KPI_COL3
45	P	AV_{DD}
46	P	AV_{SS}
47	I/O	PB.7 / EADC0_CH7 / EBI_nWRL / CAN1_TXD / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / ACMP0_O / KPI_COL4 / SPI1_CLK
48	I/O	PB.6 / EADC0_CH6 / EBI_nWRH / CAN1_RXD / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / ACMP1_O / KPI_COL5 / SPI1_SS

Table 4.1-2 M463LGCAE Multi-function Pin Table

4.1.3.3 M463 Series LQFP64-Pin Multi-function Pin Diagram

Corresponding Part Number: M463SGCAE

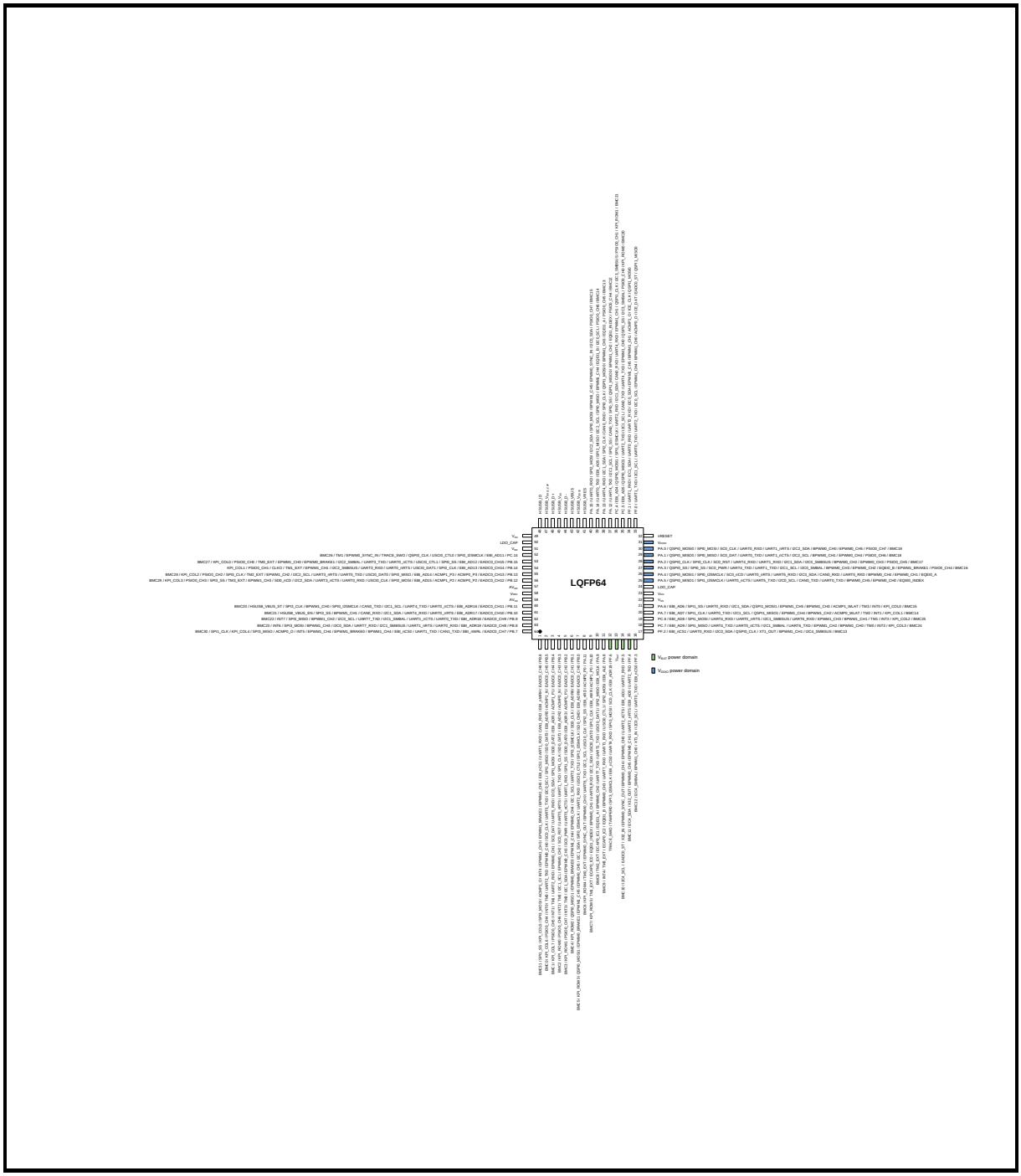


Figure 4.1-18 M463SGCAE Multi-function Pin Diagram

Pin	Type	M463SGCAE Pin Function
1	I/O	PB.6 / EADC0_CH6 / EBI_nWRH / CAN1_RXD / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / ACMP1_O / SPI3_MOSI / KPI_COL5 / SPI1_SS / BMC31
2	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD /

Pin	Type	M463SGCAE Pin Function
		SC0_CLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6 / BMC0
3	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7 / BMC1
4	I/O	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0 / BMC2
5	I/O	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1 / BMC3
6	I/O	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1 / KPI_ROW2 / BMC4
7	I/O	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1 / KPI_ROW3 / BMC5
8	I/O	PA.11 / ACMP0_P0 / EBI_nRD / SPI2_SS / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / KPI_ROW4 / BMC6
9	I/O	PA.10 / ACMP1_P0 / EBI_nWR / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / TM1_EXT / KPI_ROW5 / BMC7
10	I/O	PA.9 / EBI_MCLK / SPI2_MISO / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / TM2_EXT / BMC8
11	I/O	PA.8 / EBI_ALE / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / TM3_EXT / INT4 / BMC9
12	I/O	PF.6 / EBI_ADR19 / SC0_CLK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / SPI3_I2SMCLK / TAMPER0 / TRACE_SWO
13	P	V_{BAT}
14	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / BMC10
15	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / I2C4_SDA / BMC11
16	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / BMC12
17	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / BMC13
18	I/O	PC.7 / EBI_AD9 / SPI1_MISO / UART4_TXD / UART0_nCTS / I2C1_SMBAL / UART6_TXD / EPWM1_CH2 / BPWM1_CH0 / TM0 / INT3 / KPI_COL3 / BMC24
19	I/O	PC.6 / EBI_AD8 / SPI1_MOSI / UART4_RXD / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / EPWM1_CH3 / BPWM1_CH1 / TM1 / INT2 / KPI_COL2 / BMC25
20	I/O	PA.7 / EBI_AD7 / SPI1_CLK / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1 / BMC14
21	I/O	PA.6 / EBI_AD6 / SPI1_SS / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0 / BMC15
22	P	V_{SS}
23	P	V_{DD}
24	A	LDO_CAP
25	I/O	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
26	I/O	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A

Pin	Type	M463SGCAE Pin Function
27	I/O	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4 / BMC16
28	I/O	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / PSIO0_CH5 / BMC17
29	I/O	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / PSIO0_CH6 / BMC18
30	I/O	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / PSIO0_CH7 / BMC19
31	P	V_{DDIO}
32	I	nRESET
33	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / UART2_TXD / I2C0_SCL / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST / QSPI1_MISO0
34	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / UART2_RXD / I2C0_SDA / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / QSPI1_MOSI0
35	I/O	PC.5 / EBI_AD5 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / QSPI1_SS / I2C3_SMBAL / PSIO0_CH0 / KPI_ROW0 / BMC20
36	I/O	PC.4 / EBI_AD4 / QSPI0_MOSI1 / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / QSPI1_CLK / I2C3_SMBSUS / PSIO0_CH1 / KPI_ROW1 / BMC21
37	I/O	PA.12 / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SPI0_SS / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / PSIO0_CH4 / BMC12
38	I/O	PA.13 / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SPI0_CLK / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / PSIO0_CH5 / BMC13
39	I/O	PA.14 / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SPI0_MISO / BPWM1_CH4 / EQEI1_B / I2C0_SCL / PSIO0_CH6 / BMC14
40	I/O	PA.15 / UART0_RXD / SPI2_MOSI / I2C2_SDA / SPI0_MOSI / BPWM1_CH5 / EPWM0_SYNC_IN / I2C0_SDA / PSIO0_CH7 / BMC15
41	A	HSUSB_VRES
42	P	HSUSB_VDD33
43	I/O	HSUSB_VBUS
44	A	HSUSB_D-
45	P	HSUSB_VSS
46	A	HSUSB_D+
47	A	HSUSB_VDD12_CAP
48	I	HSUSB_ID
49	P	V_{SS}
50	A	LDO_CAP
51	P	V_{DD}
52	I/O	PC.14 / EBI_AD11 / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / TM1 / BMC26
53	I/O	PB.15 / EADC0_CH15 / EBI_AD12 / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM0_BRAKE1 / EPWM1_CH0 / TM0_EXT / PSIO0_CH0 / KPI_COL0 / BMC27
54	I/O	PB.14 / EADC0_CH14 / EBI_AD13 / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD /

Pin	Type	M463SGCAE Pin Function
		I2C2_SMBSUS / EPWM1_CH1 / TM1_EXT / CLKO / PSIO0_CH1 / KPI_COL1
55	I/O	PB.13 / EADC0_CH13 / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / SPI3_CLK / PSIO0_CH2 / KPI_COL2 / BMC28
56	I/O	PB.12 / EADC0_CH12 / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / SPI3_SS / PSIO0_CH3 / KPI_COL3 / BMC29
57	P	AV_{DD}
58	A	V_{REF}
59	P	AV_{SS}
60	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / HSUSB_VBUS_ST / BMC20
61	I/O	PB.10 / EADC0_CH10 / EBI_ADR17 / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / HSUSB_VBUS_EN / BMC21
62	I/O	PB.9 / EADC0_CH9 / EBI_ADR18 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / INT7 / BMC22
63	I/O	PB.8 / EADC0_CH8 / EBI_ADR19 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / INT6 / BMC23
64	I/O	PB.7 / EADC0_CH7 / EBI_nWRL / CAN1_TXD / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / ACMP0_O / SPI3_MISO / KPI_COL4 / SPI1_CLK / BMC30

Table 4.1-3 M463SGCAE Multi-function Pin Table

4.1.3.4 M463 Series LQFP128-Pin Multi-function Pin Diagram

Corresponding Part Number: M463KGCAE

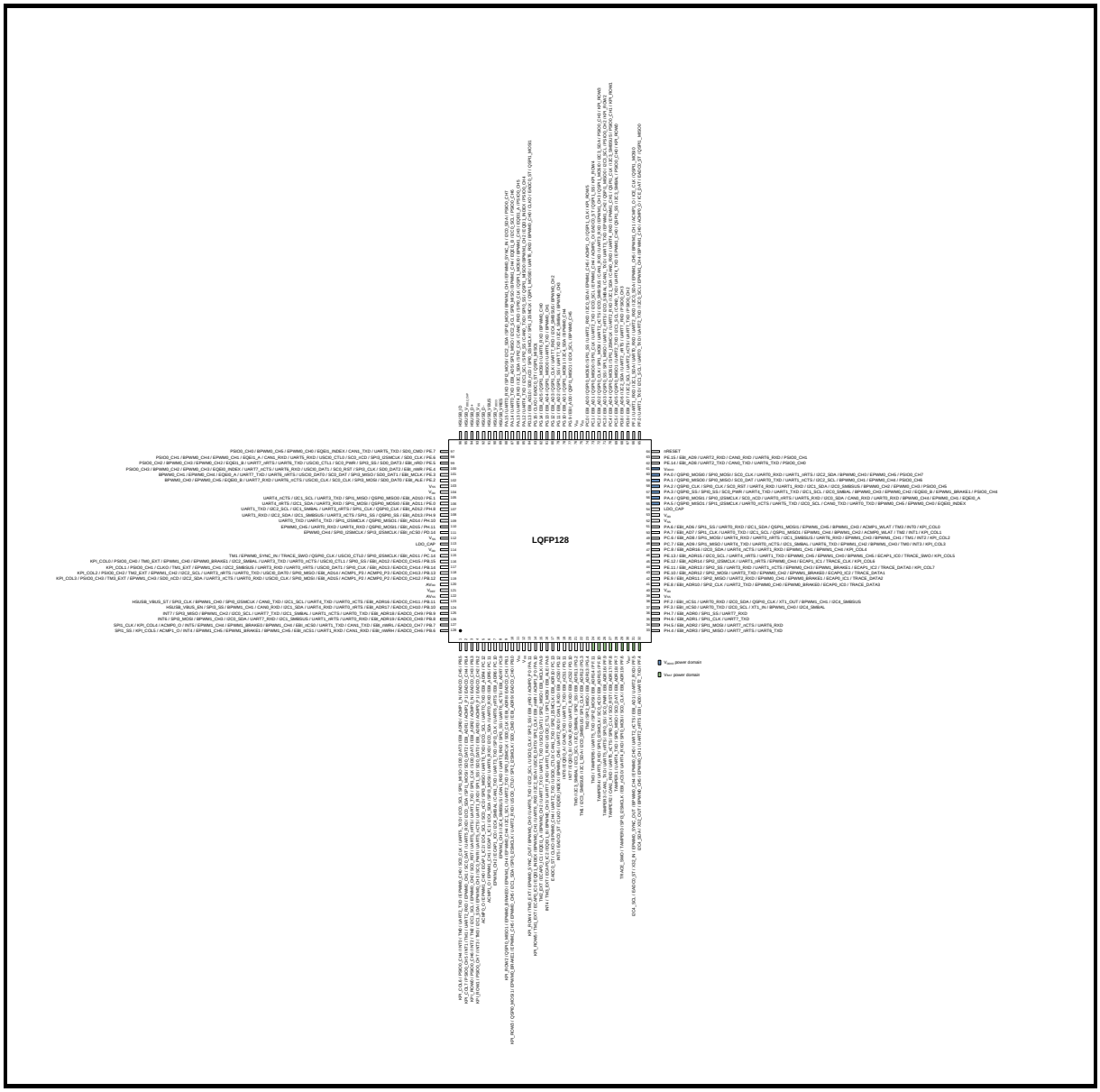


Figure 4.1-19 M463KGCAE Multi-function Pin Diagram

Pin	Type	M463KGCAE Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1

Pin	Type	M463KGCAE Pin Function
5	I/O	PC.12 / EBI_ADR4 / UART0_TXD / I2C0_SCL / UART6_TXD / SPI3_MISO / SC0_nCD / I2C4_SCL / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	I/O	PC.11 / EBI_ADR5 / UART0_RXD / I2C0_SDA / UART6_RXD / SPI3_MOSI / I2C4_SDA / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	I/O	PC.10 / EBI_ADR6 / UART6_nRTS / SPI3_CLK / UART3_TXD / CAN1_TXD / I2C4_SMBAL / ECAP1_IC0 / EPWM1_CH2
8	I/O	PC.9 / EBI_ADR7 / UART6_nCTS / SPI3_SS / UART3_RXD / CAN1_RXD / I2C4_SMBSUS / EPWM1_CH3
9	I/O	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1 / KPI_ROW2
10	I/O	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1 / KPI_ROW3
11	P	V_{SS}
12	P	V_{DD}
13	I/O	PA.11 / ACMP0_P0 / EBI_nRD / SPI2_SS / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / KPI_ROW4
14	I/O	PA.10 / ACMP1_P0 / EBI_nWR / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / TM1_EXT / KPI_ROW5
15	I/O	PA.9 / EBI_MCLK / SPI2_MISO / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / TM2_EXT
16	I/O	PA.8 / EBI_ALE / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / TM3_EXT / INT4
17	I/O	PC.13 / EBI_ADR10 / SPI2_I2SMCLK / CAN1_TXD / USCI0_CTL0 / UART2_TXD / BPWM0_CH4 / CLKO / EADC0_ST
18	I/O	PD.12 / EBI_nCS0 / CAN1_RXD / UART2_RXD / BPWM0_CH5 / EQEI0_INDEX / CLKO / EADC0_ST / INT5
19	I/O	PD.11 / EBI_nCS1 / UART1_TXD / CAN0_TXD / EQEI0_A / INT6
20	I/O	PD.10 / EBI_nCS2 / UART1_RXD / CAN0_RXD / EQEI0_B / INT7
21	I/O	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / I2C3_SMBAL / TM0
22	I/O	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / I2C3_SMBSUS / TM1
23	I/O	PG.4 / EBI_ADR13 / SPI2_MISO / TM2
24	I/O	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / TAMPER5 / TM3
25	I/O	PF.10 / EBI_ADR15 / SC0_nCD / SPI0_I2SMCLK / UART5_RXD / TAMPER4
26	I/O	PF.9 / EBI_ADR16 / SC0_PWR / SPI0_SS / UART5_nRTS / CAN1_TXD / TAMPER3
27	I/O	PF.8 / EBI_ADR17 / SC0_RST / SPI0_CLK / UART5_nCTS / CAN1_RXD / TAMPER2
28	I/O	PF.7 / EBI_ADR18 / SC0_DAT / SPI0_MISO / UART4_TXD / TAMPER1
29	I/O	PF.6 / EBI_ADR19 / SC0_CLK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / SPI3_I2SMCLK / TAMPER0 / TRACE_SWO
30	P	V_{BAT}
31	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL
32	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT /

Pin	Type	M463KGCAE Pin Function
		I2C4_SDA
33	I/O	PH.4 / EBI_ADR3 / SPI1_MISO / UART7_nRTS / UART6_TXD
34	I/O	PH.5 / EBI_ADR2 / SPI1_MOSI / UART7_nCTS / UART6_RXD
35	I/O	PH.6 / EBI_ADR1 / SPI1_CLK / UART7_TXD
36	I/O	PH.7 / EBI_ADR0 / SPI1_SS / UART7_RXD
37	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL
38	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS
39	P	V_{SS}
40	P	V_{DD}
41	I/O	PE.8 / EBI_ADR10 / SPI2_CLK / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / TRACE_DATA3
42	I/O	PE.9 / EBI_ADR11 / SPI2_MISO / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / TRACE_DATA2
43	I/O	PE.10 / EBI_ADR12 / SPI2_MOSI / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / TRACE_DATA1
44	I/O	PE.11 / EBI_ADR13 / SPI2_SS / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0 / KPI_COL7
45	I/O	PE.12 / EBI_ADR14 / SPI2_I2SMCLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK / KPI_COL6
46	I/O	PE.13 / EBI_ADR15 / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0 / TRACE_SWO / KPI_COL5
47	I/O	PC.8 / EBI_ADR16 / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4 / KPI_COL4
48	I/O	PC.7 / EBI_AD9 / SPI1_MISO / UART4_TXD / UART0_nCTS / I2C1_SMBAL / UART6_TXD / EPWM1_CH2 / BPWM1_CH0 / TM0 / INT3 / KPI_COL3
49	I/O	PC.6 / EBI_AD8 / SPI1_MOSI / UART4_RXD / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / EPWM1_CH3 / BPWM1_CH1 / TM1 / INT2 / KPI_COL2
50	I/O	PA.7 / EBI_AD7 / SPI1_CLK / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
51	I/O	PA.6 / EBI_AD6 / SPI1_SS / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
52	P	V_{SS}
53	P	V_{DD}
54	A	LDO_CAP
55	I/O	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
56	I/O	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
57	I/O	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
58	I/O	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / PSIO0_CH5

Pin	Type	M463KGCAE Pin Function
59	I/O	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / PSIO0_CH6
60	I/O	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / PSIO0_CH7
61	P	V_{DDIO}
62	I/O	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / UART6_TXD / PSIO0_CH0
63	I/O	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / UART6_RXD / PSIO0_CH1
64	I	nRESET
65	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / UART2_TXD / I2C0_SCL / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST / QSPI1_MISO0
66	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / UART2_RXD / I2C0_SDA / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / QSPI1_MOSI0
67	I/O	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / UART7_TXD / PSIO0_CH2
68	I/O	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / UART7_RXD / PSIO0_CH3
69	I/O	PC.5 / EBI_AD5 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / QSPI1_SS / I2C3_SMBAL / PSIO0_CH0 / KPI_ROW0
70	I/O	PC.4 / EBI_AD4 / QSPI0_MOSI1 / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / QSPI1_CLK / I2C3_SMBSUS / PSIO0_CH1 / KPI_ROW1
71	I/O	PC.3 / EBI_AD3 / QSPI0_SS / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / CAN1_TXD / UART3_TXD / EPWM1_CH2 / QSPI1_MISO0 / I2C3_SCL / PSIO0_CH2 / KPI_ROW2
72	I/O	PC.2 / EBI_AD2 / QSPI0_CLK / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / QSPI1_MOSI0 / I2C3_SDA / PSIO0_CH3 / KPI_ROW3
73	I/O	PC.1 / EBI_AD1 / QSPI0_MISO0 / SPI1_CLK / UART2_TXD / I2C0_SCL / EPWM1_CH4 / ACMP0_O / EADC0_ST / QSPI1_SS / KPI_ROW4
74	I/O	PC.0 / EBI_AD0 / QSPI0_MOSI0 / SPI1_SS / UART2_RXD / I2C0_SDA / EPWM1_CH5 / ACMP1_O / QSPI1_CLK / KPI_ROW5
75	P	V_{SS}
76	P	V_{DD}
77	I/O	PG.9 / EBI_AD0 / QSPI1_MISO1 / I2C4_SCL / BPWM0_CH5
78	I/O	PG.10 / EBI_AD1 / QSPI1_MOSI1 / I2C4_SDA / BPWM0_CH4
79	I/O	PG.11 / EBI_AD2 / QSPI1_SS / UART7_TXD / I2C4_SMBAL / BPWM0_CH3
80	I/O	PG.12 / EBI_AD3 / QSPI1_CLK / UART7_RXD / I2C4_SMBSUS / BPWM0_CH2
81	I/O	PG.13 / EBI_AD4 / QSPI1_MISO0 / UART6_TXD / BPWM0_CH1
82	I/O	PG.14 / EBI_AD5 / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0
83	I/O	PG.15 / CLKO / EADC0_ST / QSPI1_MISO1
84	I/O	PD.13 / EBI_AD10 / SD0_nCD / SPI0_I2SMCLK / SPI1_I2SMCLK / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0 / CLKO / EADC0_ST / QSPI1_MOSI1
85	I/O	PA.12 / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SPI0_SS / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / PSIO0_CH4
86	I/O	PA.13 / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SPI0_CLK / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / PSIO0_CH5
87	I/O	PA.14 / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SPI0_MISO / BPWM1_CH4 / EQEI1_B /

Pin	Type	M463KGCAE Pin Function
		I2C0_SCL / PSIO0_CH6
88	I/O	PA.15 / UART0_RXD / SPI2_MOSI / I2C2_SDA / SPI0_MOSI / BPWM1_CH5 / EPWM0_SYNC_IN / I2C0_SDA / PSIO0_CH7
89	A	HSUSB_VRES
90	P	HSUSB_VDD33
91	I/O	HSUSB_VBUS
92	A	HSUSB_D-
93	P	HSUSB_VSS
94	A	HSUSB_D+
95	A	HSUSB_VDD12_CAP
96	I	HSUSB_ID
97	I/O	PE.7 / SD0_CMD / UART5_TXD / CAN1_TXD / EQEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / PSIO0_CH0
98	I/O	PE.6 / SD0_CLK / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / CAN1_RXD / EQEI1_A / EPWM0_CH1 / BPWM0_CH4 / PSIO0_CH1
99	I/O	PE.5 / EBI_nRD / SD0_DAT3 / SPI3_SS / SC0_PWR / USCI0_CTL1 / UART6_TXD / UART7_nRTS / EQEI1_B / EPWM0_CH2 / BPWM0_CH3 / PSIO0_CH2
100	I/O	PE.4 / EBI_nWR / SD0_DAT2 / SPI3_CLK / SC0_RST / USCI0_DAT1 / UART6_RXD / UART7_nCTS / EQEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / PSIO0_CH3
101	I/O	PE.3 / EBI_MCLK / SD0_DAT1 / SPI3_MISO / SC0_DAT / USCI0_DAT0 / UART6_nRTS / UART7_TXD / EQEI0_A / EPWM0_CH4 / BPWM0_CH1
102	I/O	PE.2 / EBI_ALE / SD0_DAT0 / SPI3_MOSI / SC0_CLK / USCI0_CLK / UART6_nCTS / UART7_RXD / EQEI0_B / EPWM0_CH5 / BPWM0_CH0
103	P	V_{SS}
104	P	V_{DD}
105	I/O	PE.1 / EBI_AD10 / QSPI0_MISO0 / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS
106	I/O	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS
107	I/O	PH.8 / EBI_AD12 / QSPI0_CLK / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD
108	I/O	PH.9 / EBI_AD13 / QSPI0_SS / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD
109	I/O	PH.10 / EBI_AD14 / QSPI0_MISO1 / SPI1_I2SMCLK / UART4_TXD / UART0_TXD
110	I/O	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5
111	I/O	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SPI0_I2SMCLK / EPWM0_CH4
112	P	V_{SS}
113	A	LDO_CAP
114	P	V_{DD}
115	I/O	PC.14 / EBI_AD11 / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / TM1
116	I/O	PB.15 / EADC0_CH15 / EBI_AD12 / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM0_BRAKE1 / EPWM1_CH0 / TM0_EXT / PSIO0_CH0 / KPI_COL0

Pin	Type	M463KGCAE Pin Function
117	I/O	PB.14 / EADC0_CH14 / EBI_AD13 / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM1_CH1 / TM1_EXT / CLKO / PSIO0_CH1 / KPI_COL1
118	I/O	PB.13 / EADC0_CH13 / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / PSIO0_CH2 / KPI_COL2
119	I/O	PB.12 / EADC0_CH12 / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / PSIO0_CH3 / KPI_COL3
120	P	AV_{DD}
121	A	V_{REF}
122	P	AV_{SS}
123	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / HSUSB_VBUS_ST
124	I/O	PB.10 / EADC0_CH10 / EBI_ADR17 / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / HSUSB_VBUS_EN
125	I/O	PB.9 / EADC0_CH9 / EBI_ADR18 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / INT7
126	I/O	PB.8 / EADC0_CH8 / EBI_ADR19 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / INT6
127	I/O	PB.7 / EADC0_CH7 / EBI_nWRL / CAN1_TXD / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / ACMP0_O / KPI_COL4 / SPI1_CLK
128	I/O	PB.6 / EADC0_CH6 / EBI_nWRH / CAN1_RXD / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / ACMP1_O / KPI_COL5 / SPI1_SS

Table 4.1-4 M463KGCAE Multi-function Pin Table

4.1.3.5 M463 Series LQFP48 EPAD Pin Multi-function Pin Diagram

Corresponding Part Number: M463L2GCAC

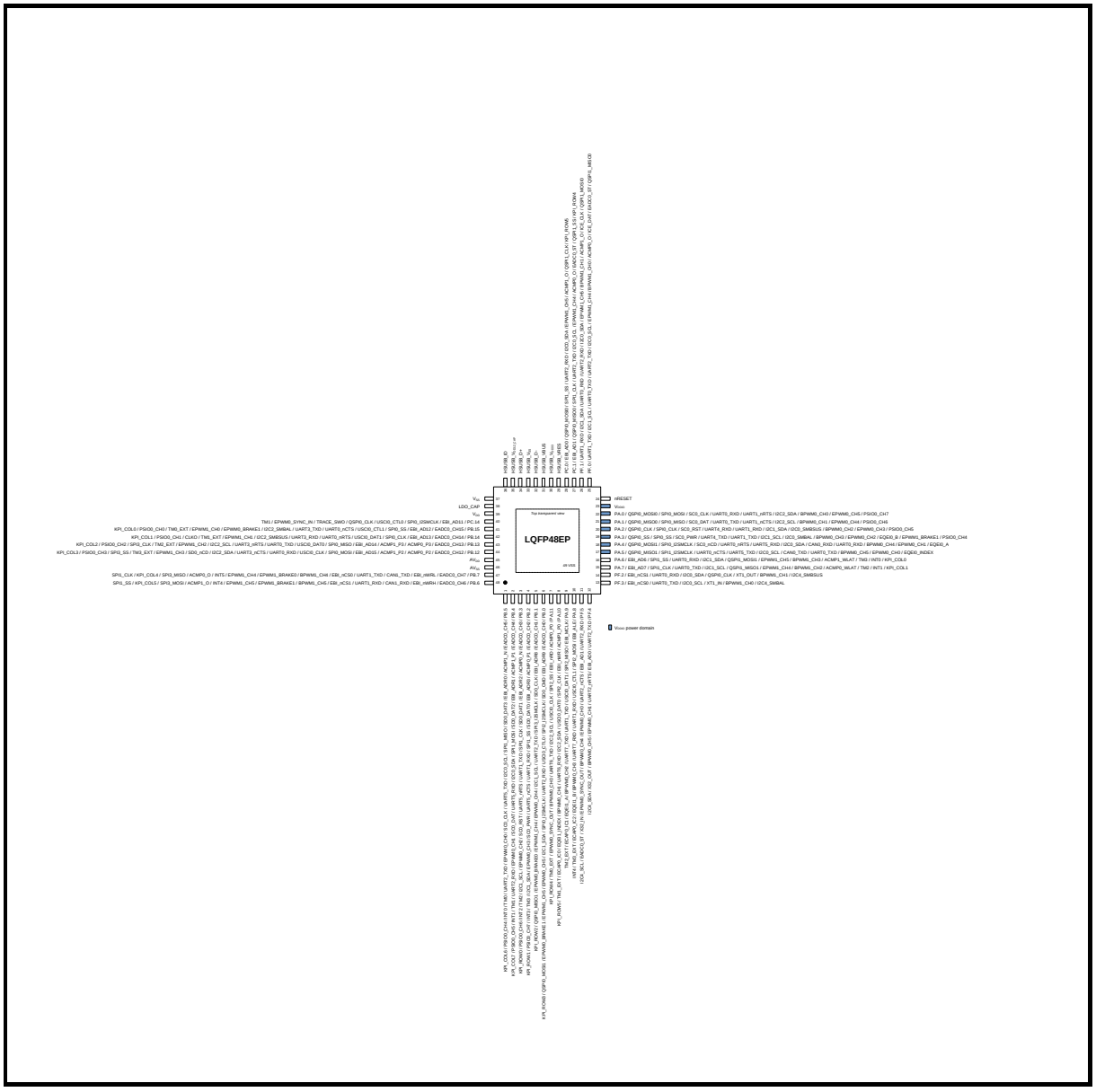


Figure 4.1-20 M463L2GCAC Multi-function Pin Diagram

Pin	Type	M463L2GCAC Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1 / KPI_ROW2
6	I/O	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1 / KPI_ROW3

Pin	Type	M463L2GCAC Pin Function
7	I/O	PA.11 / ACMP0_P0 / EBI_nRD / SPI2_SS / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / KPI_ROW4
8	I/O	PA.10 / ACMP1_P0 / EBI_nWR / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / TM1_EXT / KPI_ROW5
9	I/O	PA.9 / EBI_MCLK / SPI2_MISO / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / TM2_EXT
10	I/O	PA.8 / EBI_ALE / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / TM3_EXT / INT4
11	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL
12	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / I2C4_SDA
13	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL
14	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS
15	I/O	PA.7 / EBI_AD7 / SPI1_CLK / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
16	I/O	PA.6 / EBI_AD6 / SPI1_SS / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
17	I/O	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
18	I/O	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
19	I/O	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
20	I/O	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / PSIO0_CH5
21	I/O	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / PSIO0_CH6
22	I/O	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / PSIO0_CH7
23	P	V_{DDIO}
24	I	nRESET
25	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / UART2_TXD / I2C0_SCL / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST / QSPI1_MISO0
26	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / UART2_RXD / I2C0_SDA / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / QSPI1_MOSI0
27	I/O	PC.1 / EBI_AD1 / QSPI0_MISO0 / SPI1_CLK / UART2_TXD / I2C0_SCL / EPWM1_CH4 / ACMP0_O / EADC0_ST / QSPI1_SS / KPI_ROW4
28	I/O	PC.0 / EBI_AD0 / QSPI0_MOSI0 / SPI1_SS / UART2_RXD / I2C0_SDA / EPWM1_CH5 / ACMP1_O / QSPI1_CLK / KPI_ROW5
29	A	HSUSB_VRES
30	P	HSUSB_VDD33
31	I/O	HSUSB_VBUS
32	A	HSUSB_D-
33	P	HSUSB_VSS
34	A	HSUSB_D+
35	A	HSUSB_VDD12_CAP
36	I	HSUSB_ID
37	P	V_{SS}
38	A	LDO_CAP
39	P	V_{DD}
40	I/O	PC.14 / EBI_AD11 / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / TM1
41	I/O	PB.15 / EADC0_CH15 / EBI_AD12 / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM0_BRAKE1 / EPWM1_CH0 / TM0_EXT / PSIO0_CH0 / KPI_COL0
42	I/O	PB.14 / EADC0_CH14 / EBI_AD13 / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM1_CH1 / TM1_EXT / CLKO / PSIO0_CH1 / KPI_COL1
43	I/O	PB.13 / EADC0_CH13 / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / SPI3_CLK / PSIO0_CH2 / KPI_COL2
44	I/O	PB.12 / EADC0_CH12 / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / SPI3_SS / PSIO0_CH3 / KPI_COL3
45	P	AV_{DD}
46	P	AV_{SS}
47	I/O	PB.7 / EADC0_CH7 / EBI_nWRL / CAN1_TXD / UART1_TXD / EBI_nCS0 / BPWM1_CH4 /

Pin	Type	M463L2GCAC Pin Function
		EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / ACMP0_O / SPI3_MISO / KPI_COL4 / SPI1_CLK
48	I/O	PB.6 / EADC0_CH6 / EBI_nWRH / CAN1_RXD / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / ACMP1_O / SPI3_MOSI / KPI_COL5 / SPI1_SS
49	P	V_{SS}

Table 4.1-5 M463L2GCAC Multi-function Pin Table

4.1.3.6 M463 Series LQFP64 EPAD Pin Multi-function Pin Diagram

Corresponding Part Number: M463S2GCAC

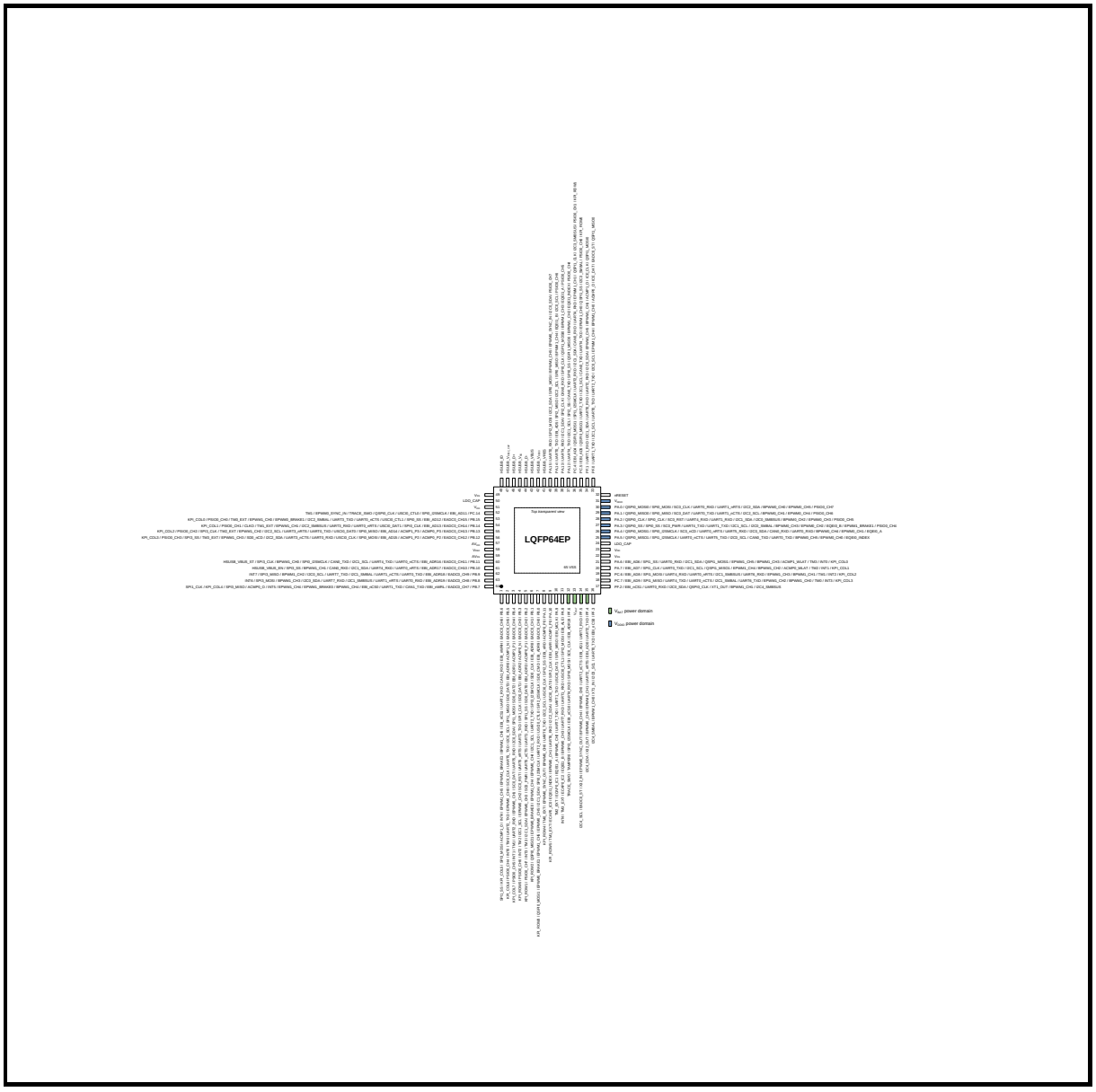


Figure 4.1-21 M463S2GCAC Multi-function Pin Diagram

Pin	Type	M463S2GCAC Pin Function
1	I/O	PB.6 / EADC0_CH6 / EBI_nWRH / CAN1_RXD / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / ACMP1_O / SPI3_MOSI / KPI_COL5 / SPI1_SS
2	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
3	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
4	I/O	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
5	I/O	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
6	I/O	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1 / KPI_ROW2
7	I/O	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD /

Pin	Type	M463S2GCAC Pin Function
		SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1 / KPI_ROW3
8	I/O	PA.11 / ACMP0_P0 / EBI_nRD / SPI2_SS / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / KPI_ROW4
9	I/O	PA.10 / ACMP1_P0 / EBI_nWR / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / TM1_EXT / KPI_ROW5
10	I/O	PA.9 / EBI_MCLK / SPI2_MISO / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / TM2_EXT
11	I/O	PA.8 / EBI_ALE / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / TM3_EXT / INT4
12	I/O	PF.6 / EBI_ADR19 / SC0_CLK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / SPI3_I2SMCLK / TAMPER0 / TRACE_SWO
13	P	V_{BAT}
14	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL
15	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / I2C4_SDA
16	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL
17	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS
18	I/O	PC.7 / EBI_AD9 / SPI1_MISO / UART4_TXD / UART0_nCTS / I2C1_SMBAL / UART6_TXD / EPWM1_CH2 / BPWM1_CH0 / TM0 / INT3 / KPI_COL3
19	I/O	PC.6 / EBI_AD8 / SPI1_MOSI / UART4_RXD / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / EPWM1_CH3 / BPWM1_CH1 / TM1 / INT2 / KPI_COL2
20	I/O	PA.7 / EBI_AD7 / SPI1_CLK / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
21	I/O	PA.6 / EBI_AD6 / SPI1_SS / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
22	P	V_{SS}
23	P	V_{DD}
24	A	LDO_CAP
25	I/O	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
26	I/O	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
27	I/O	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
28	I/O	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / PSIO0_CH5
29	I/O	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / PSIO0_CH6
30	I/O	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / PSIO0_CH7
31	P	V_{DDIO}
32	I	nRESET
33	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / UART2_TXD / I2C0_SCL / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST / QSPI1_MISO0
34	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / UART2_RXD / I2C0_SDA / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / QSPI1_MOSI0
35	I/O	PC.5 / EBI_AD5 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / QSPI1_SS / I2C3_SMBAL / PSIO0_CH0 / KPI_ROW0
36	I/O	PC.4 / EBI_AD4 / QSPI0_MOSI1 / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / QSPI1_CLK / I2C3_SMBSUS / PSIO0_CH1 / KPI_ROW1
37	I/O	PA.12 / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SPI0_SS / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / PSIO0_CH4
38	I/O	PA.13 / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SPI0_CLK / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / PSIO0_CH5
39	I/O	PA.14 / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SPI0_MISO / BPWM1_CH4 / EQEI1_B / I2C0_SCL / PSIO0_CH6
40	I/O	PA.15 / UART0_RXD / SPI2_MOSI / I2C2_SDA / SPI0_MOSI / BPWM1_CH5 / EPWM0_SYNC_IN / I2C0_SDA / PSIO0_CH7
41	A	HSUSB_VRES
42	P	HSUSB_VDD33
43	I/O	HSUSB_VBUS
44	A	HSUSB_D-
45	P	HSUSB_VSS
46	A	HSUSB_D+

Pin	Type	M463S2GCAC Pin Function
47	A	HSUSB_VDD12_CAP
48	I	HSUSB_ID
49	P	V_{SS}
50	A	LDO_CAP
51	P	V_{DD}
52	I/O	PC.14 / EBI_AD11 / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / TM1
53	I/O	PB.15 / EADC0_CH15 / EBI_AD12 / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM0_BRAKE1 / EPWM1_CH0 / TM0_EXT / PSIO0_CH0 / KPI_COL0
54	I/O	PB.14 / EADC0_CH14 / EBI_AD13 / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM1_CH1 / TM1_EXT / CLKO / PSIO0_CH1 / KPI_COL1
55	I/O	PB.13 / EADC0_CH13 / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / SPI3_CLK / PSIO0_CH2 / KPI_COL2
56	I/O	PB.12 / EADC0_CH12 / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / SPI3_SS / PSIO0_CH3 / KPI_COL3
57	P	AV_{DD}
58	A	V_{REF}
59	P	AV_{SS}
60	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / HSUSB_VBUS_ST
61	I/O	PB.10 / EADC0_CH10 / EBI_ADR17 / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / HSUSB_VBUS_EN
62	I/O	PB.9 / EADC0_CH9 / EBI_ADR18 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / INT7
63	I/O	PB.8 / EADC0_CH8 / EBI_ADR19 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / INT6
64	I/O	PB.7 / EADC0_CH7 / EBI_nWRL / CAN1_TXD / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / ACMP0_O / SPI3_MISO / KPI_COL4 / SPI1_CLK
65	P	V_{SS}

Table 4.1-6 M463S2GCAC Multi-function Pin Table

4.1.3.7 M463 Series LQFP128 EPAD Pin Multi-function Pin Diagram

Corresponding Part Number: M463K2GCAC

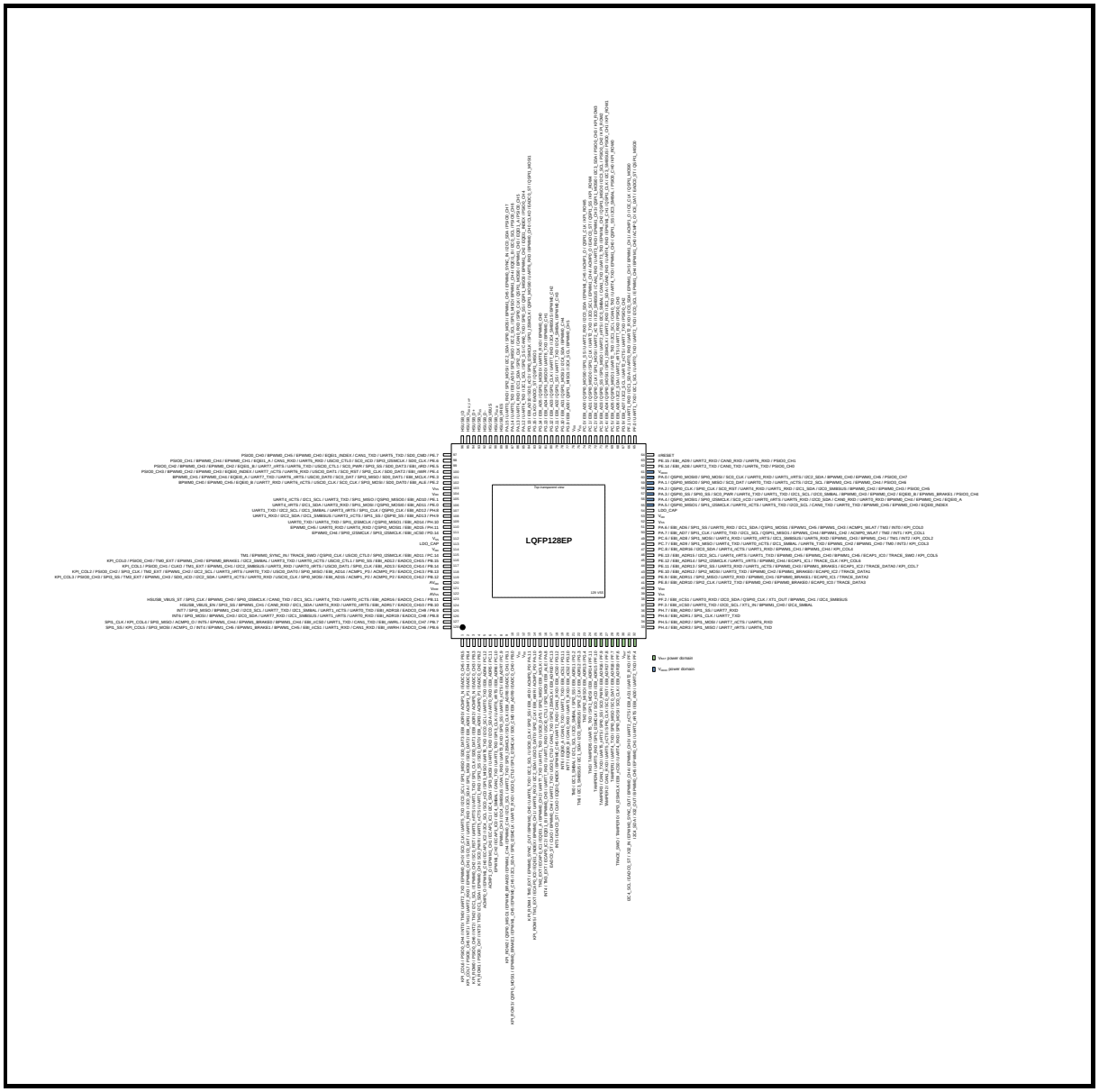


Figure 4.1-22 M463K2GCAC Multi-function Pin Diagram

Pin	Type	M463K2GCAC Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PC.12 / EBI_ADR4 / UART0_TXD / I2C0_SCL / UART6_TXD / SPI3_MISO / SC0_nCD / I2C4_SCL / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	I/O	PC.11 / EBI_ADR5 / UART0_RXD / I2C0_SDA / UART6_RXD / SPI3_MOSI / I2C4_SDA / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	I/O	PC.10 / EBI_ADR6 / UART6_nRTS / SPI3_CLK / UART3_TXD / CAN1_TXD / I2C4_SMBAL /

Pin	Type	M463K2GCAC Pin Function
		ECAP1_IC0 / EPWM1_CH2
8	I/O	PC.9 / EBI_ADR7 / UART6_nCTS / SPI3_SS / UART3_RXD / CAN1_RXD / I2C4_SMBSUS / EPWM1_CH3
9	I/O	PB.1 / EADC0_CH1 / EBI_ADR8 / SD0_CLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / QSPI0_MISO1 / KPI_ROW2
10	I/O	PB.0 / EADC0_CH0 / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / QSPI0_MOSI1 / KPI_ROW3
11	P	V_{SS}
12	P	V_{DD}
13	I/O	PA.11 / ACMP0_P0 / EBI_nRD / SPI2_SS / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / TM0_EXT / KPI_ROW4
14	I/O	PA.10 / ACMP1_P0 / EBI_nWR / SPI2_CLK / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / TM1_EXT / KPI_ROW5
15	I/O	PA.9 / EBI_MCLK / SPI2_MISO / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / TM2_EXT
16	I/O	PA.8 / EBI_ALE / SPI2_MOSI / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / TM3_EXT / INT4
17	I/O	PC.13 / EBI_ADR10 / SPI2_I2SMCLK / CAN1_TXD / USCI0_CTL0 / UART2_TXD / BPWM0_CH4 / CLKO / EADC0_ST
18	I/O	PD.12 / EBI_nCS0 / CAN1_RXD / UART2_RXD / BPWM0_CH5 / EQEI0_INDEX / CLKO / EADC0_ST / INT5
19	I/O	PD.11 / EBI_nCS1 / UART1_TXD / CAN0_TXD / EQEI0_A / INT6
20	I/O	PD.10 / EBI_nCS2 / UART1_RXD / CAN0_RXD / EQEI0_B / INT7
21	I/O	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / I2C3_SMBAL / TM0
22	I/O	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / I2C3_SMBSUS / TM1
23	I/O	PG.4 / EBI_ADR13 / SPI2_MISO / TM2
24	I/O	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / TAMPER5 / TM3
25	I/O	PF.10 / EBI_ADR15 / SC0_nCD / SPI0_I2SMCLK / UART5_RXD / TAMPER4
26	I/O	PF.9 / EBI_ADR16 / SC0_PWR / SPI0_SS / UART5_nRTS / CAN1_TXD / TAMPER3
27	I/O	PF.8 / EBI_ADR17 / SC0_RST / SPI0_CLK / UART5_nCTS / CAN1_RXD / TAMPER2
28	I/O	PF.7 / EBI_ADR18 / SC0_DAT / SPI0_MISO / UART4_TXD / TAMPER1
29	I/O	PF.6 / EBI_ADR19 / SC0_CLK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / SPI3_I2SMCLK / TAMPER0 / TRACE_SWO
30	P	V_{BAT}
31	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL
32	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / I2C4_SDA
33	I/O	PH.4 / EBI_ADR3 / SPI1_MISO / UART7_nRTS / UART6_TXD
34	I/O	PH.5 / EBI_ADR2 / SPI1_MOSI / UART7_nCTS / UART6_RXD
35	I/O	PH.6 / EBI_ADR1 / SPI1_CLK / UART7_TXD
36	I/O	PH.7 / EBI_ADR0 / SPI1_SS / UART7_RXD
37	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL
38	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS
39	P	V_{SS}
40	P	V_{DD}
41	I/O	PE.8 / EBI_ADR10 / SPI2_CLK / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / TRACE_DATA3
42	I/O	PE.9 / EBI_ADR11 / SPI2_MISO / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / TRACE_DATA2
43	I/O	PE.10 / EBI_ADR12 / SPI2_MOSI / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / TRACE_DATA1
44	I/O	PE.11 / EBI_ADR13 / SPI2_SS / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0 / KPI_COL7
45	I/O	PE.12 / EBI_ADR14 / SPI2_I2SMCLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK / KPI_COL6
46	I/O	PE.13 / EBI_ADR15 / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0 / TRACE_SWO / KPI_COL5
47	I/O	PC.8 / EBI_ADR16 / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4 / KPI_COL4
48	I/O	PC.7 / EBI_AD9 / SPI1_MISO / UART4_TXD / UART0_nCTS / I2C1_SMBAL / UART6_TXD / EPWM1_CH2 / BPWM1_CH0 / TM0 / INT3 / KPI_COL3
49	I/O	PC.6 / EBI_AD8 / SPI1_MOSI / UART4_RXD / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / EPWM1_CH3 / BPWM1_CH1 / TM1 / INT2 / KPI_COL2
50	I/O	PA.7 / EBI_AD7 / SPI1_CLK / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 /

Pin	Type	M463K2GCAC Pin Function
		BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
51	I/O	PA.6 / EBI_AD6 / SPI1_SS / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
52	P	V_{SS}
53	P	V_{DD}
54	A	LDO_CAP
55	I/O	PA.5 / QSPI0_MISO1 / SPI1_I2SMCLK / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
56	I/O	PA.4 / QSPI0_MOSI1 / SPI0_I2SMCLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
57	I/O	PA.3 / QSPI0_SS / SPI0_SS / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
58	I/O	PA.2 / QSPI0_CLK / SPI0_CLK / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / PSIO0_CH5
59	I/O	PA.1 / QSPI0_MISO0 / SPI0_MISO / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / BPWM0_CH1 / EPWM0_CH4 / PSIO0_CH6
60	I/O	PA.0 / QSPI0_MOSI0 / SPI0_MOSI / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / BPWM0_CH0 / EPWM0_CH5 / PSIO0_CH7
61	P	V_{DDIO}
62	I/O	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / UART6_TXD / PSIO0_CH0
63	I/O	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / UART6_RXD / PSIO0_CH1
64	I	nRESET
65	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / UART2_TXD / I2C0_SCL / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST / QSPI1_MISO0
66	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / UART2_RXD / I2C0_SDA / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / QSPI1_MOSI0
67	I/O	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / UART7_TXD / PSIO0_CH2
68	I/O	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / UART7_RXD / PSIO0_CH3
69	I/O	PC.5 / EBI_AD5 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / QSPI1_SS / I2C3_SMBAL / PSIO0_CH0 / KPI_ROW0
70	I/O	PC.4 / EBI_AD4 / QSPI0_MOSI1 / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / QSPI1_CLK / I2C3_SMBSUS / PSIO0_CH1 / KPI_ROW1
71	I/O	PC.3 / EBI_AD3 / QSPI0_SS / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / CAN1_TXD / UART3_TXD / EPWM1_CH2 / QSPI1_MISO0 / I2C3_SCL / PSIO0_CH2 / KPI_ROW2
72	I/O	PC.2 / EBI_AD2 / QSPI0_CLK / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / QSPI1_MOSI0 / I2C3_SDA / PSIO0_CH3 / KPI_ROW3
73	I/O	PC.1 / EBI_AD1 / QSPI0_MISO0 / SPI1_CLK / UART2_TXD / I2C0_SCL / EPWM1_CH4 / ACMP0_O / EADC0_ST / QSPI1_SS / KPI_ROW4
74	I/O	PC.0 / EBI_AD0 / QSPI0_MOSI0 / SPI1_SS / UART2_RXD / I2C0_SDA / EPWM1_CH5 / ACMP1_O / QSPI1_CLK / KPI_ROW5
75	P	V_{SS}
76	P	V_{DD}
77	I/O	PG.9 / EBI_AD0 / QSPI1_MISO1 / I2C4_SCL / BPWM0_CH5
78	I/O	PG.10 / EBI_AD1 / QSPI1_MOSI1 / I2C4_SDA / BPWM0_CH4
79	I/O	PG.11 / EBI_AD2 / QSPI1_SS / UART7_TXD / I2C4_SMBAL / BPWM0_CH3
80	I/O	PG.12 / EBI_AD3 / QSPI1_CLK / UART7_RXD / I2C4_SMBSUS / BPWM0_CH2
81	I/O	PG.13 / EBI_AD4 / QSPI1_MISO0 / UART6_TXD / BPWM0_CH1
82	I/O	PG.14 / EBI_AD5 / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0
83	I/O	PG.15 / CLKO / EADC0_ST / QSPI1_MISO1
84	I/O	PD.13 / EBI_AD10 / SD0_nCD / SPI0_I2SMCLK / SPI1_I2SMCLK / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0 / CLKO / EADC0_ST / QSPI1_MOSI1
85	I/O	PA.12 / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SPI0_SS / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / PSIO0_CH4
86	I/O	PA.13 / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SPI0_CLK / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / PSIO0_CH5
87	I/O	PA.14 / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SPI0_MISO / BPWM1_CH4 / EQEI1_B / I2C0_SCL / PSIO0_CH6
88	I/O	PA.15 / UART0_RXD / SPI2_MOSI / I2C2_SDA / SPI0_MOSI / BPWM1_CH5 / EPWM0_SYNC_IN / I2C0_SDA / PSIO0_CH7
89	A	HSUSB_VRES
90	P	HSUSB_VDD33
91	I/O	HSUSB_VBUS
92	A	HSUSB_D-
93	P	HSUSB_VSS
94	A	HSUSB_D+
95	A	HSUSB_VDD12_CAP
96	I	HSUSB_ID

Pin	Type	M463K2GCAC Pin Function
97	I/O	PE.7 / SD0_CMD / UART5_TXD / CAN1_TXD / EQEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / PSIO0_CH0
98	I/O	PE.6 / SD0_CLK / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / CAN1_RXD / EQEI1_A / EPWM0_CH1 / BPWM0_CH4 / PSIO0_CH1
99	I/O	PE.5 / EBI_nRD / SD0_DAT3 / SPI3_SS / SC0_PWR / USCI0_CTL1 / UART6_TXD / UART7_nRTS / EQEI1_B / EPWM0_CH2 / BPWM0_CH3 / PSIO0_CH2
100	I/O	PE.4 / EBI_nWR / SD0_DAT2 / SPI3_CLK / SC0_RST / USCI0_DAT1 / UART6_RXD / UART7_nCTS / EQEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / PSIO0_CH3
101	I/O	PE.3 / EBI_MCLK / SD0_DAT1 / SPI3_MISO / SC0_DAT / USCI0_DAT0 / UART6_nRTS / UART7_TXD / EQEI0_A / EPWM0_CH4 / BPWM0_CH1
102	I/O	PE.2 / EBI_ALE / SD0_DAT0 / SPI3_MOSI / SC0_CLK / USCI0_CLK / UART6_nCTS / UART7_RXD / EQEI0_B / EPWM0_CH5 / BPWM0_CH0
103	P	V_{SS}
104	P	V_{DD}
105	I/O	PE.1 / EBI_AD10 / QSPI0_MISO0 / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS
106	I/O	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS
107	I/O	PH.8 / EBI_AD12 / QSPI0_CLK / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD
108	I/O	PH.9 / EBI_AD13 / QSPI0_SS / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD
109	I/O	PH.10 / EBI_AD14 / QSPI0_MISO1 / SPI1_I2SMCLK / UART4_TXD / UART0_TXD
110	I/O	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5
111	I/O	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SPI0_I2SMCLK / EPWM0_CH4
112	P	V_{SS}
113	A	LDO_CAP
114	P	V_{DD}
115	I/O	PC.14 / EBI_AD11 / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / TM1
116	I/O	PB.15 / EADC0_CH15 / EBI_AD12 / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / EPWM0_BRAKE1 / EPWM1_CH0 / TM0_EXT / PSIO0_CH0 / KPI_COL0
117	I/O	PB.14 / EADC0_CH14 / EBI_AD13 / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / EPWM1_CH1 / TM1_EXT / CLKO / PSIO0_CH1 / KPI_COL1
118	I/O	PB.13 / EADC0_CH13 / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / EPWM1_CH2 / TM2_EXT / SPI3_CLK / PSIO0_CH2 / KPI_COL2
119	I/O	PB.12 / EADC0_CH12 / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / EPWM1_CH3 / TM3_EXT / SPI3_SS / PSIO0_CH3 / KPI_COL3
120	P	AV_{DD}
121	A	V_{REF}
122	P	AV_{SS}
123	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / HSUSB_VBUS_ST
124	I/O	PB.10 / EADC0_CH10 / EBI_ADR17 / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / HSUSB_VBUS_EN
125	I/O	PB.9 / EADC0_CH9 / EBI_ADR18 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / INT7
126	I/O	PB.8 / EADC0_CH8 / EBI_ADR19 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / INT6
127	I/O	PB.7 / EADC0_CH7 / EBI_nWRL / CAN1_TXD / UART1_TXD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / ACMP0_O / SPI3_MISO / KPI_COL4 / SPI1_CLK
128	I/O	PB.6 / EADC0_CH6 / EBI_nWRH / CAN1_RXD / UART1_RXD / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / ACMP1_O / SPI3_MOSI / KPI_COL5 / SPI1_SS
129	P	V_{SS}

Table 4.1-7 M463K2GCAC Multi-function Pin Table

4.1.4 M467 Series Multi-function Pin Diagram

4.1.4.1 M467 Series LQFP64-Pin Multi-function Pin Diagram

Corresponding Part Number: M467SJHAN

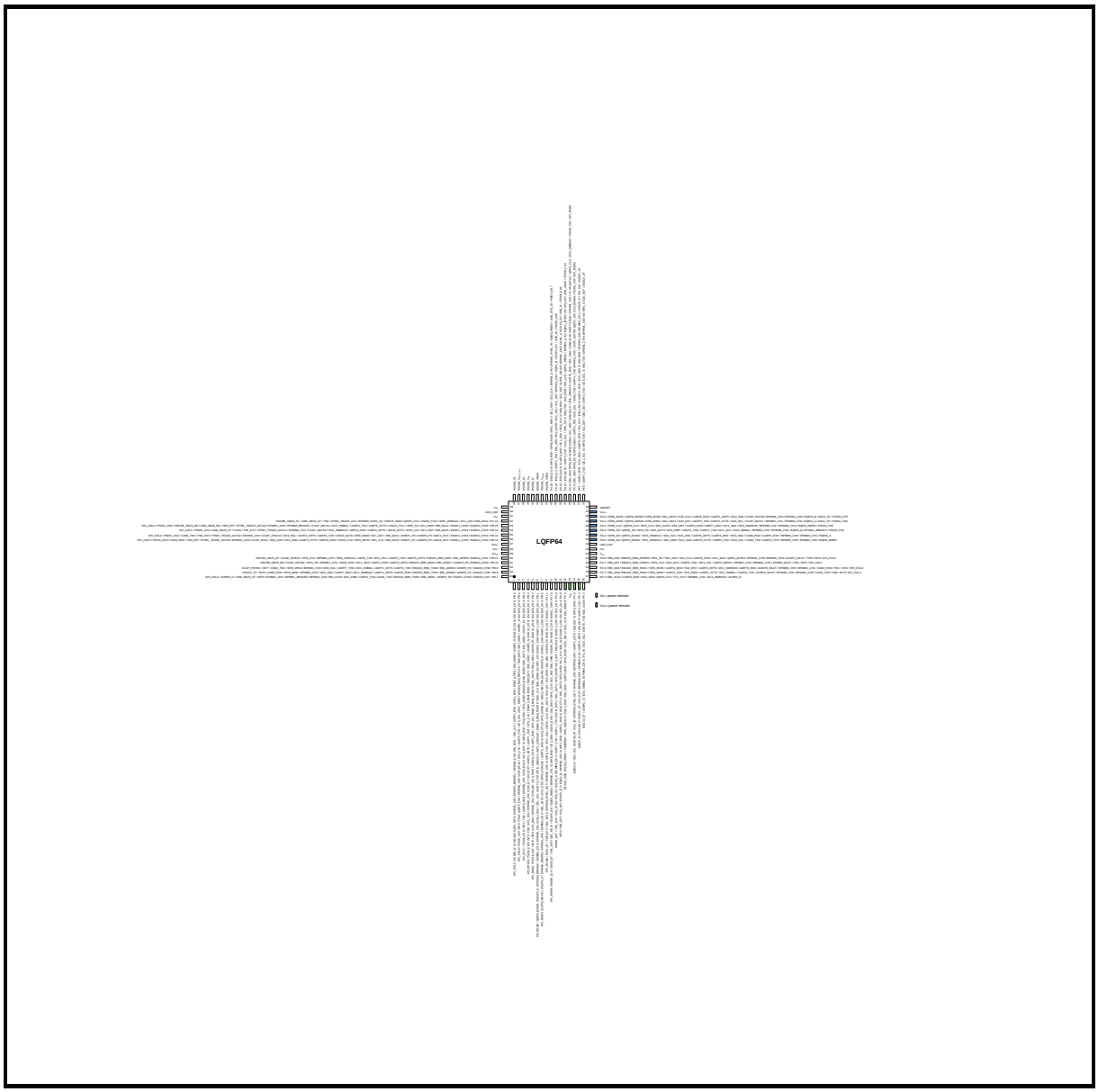


Figure 4.1-23 M467SJHAN Multi-function Pin Diagram

Pin	Type	M467SJHAN Pin Function
1	I/O	PB.6 / EADC0_CH6 / EADC2_CH14 / ACMP2_N / EBI_nWRH / EMAC0_PPS / CAN1_RXD / UART1_RXD / SD1_CLK / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O / KPI_COL5
2	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / EMAC0_RMII_REFCLK / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
3	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / EMAC0_RMII_RXD0 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
4	I/O	PB.3 / EADC0_CH3 / EADC1_CH11 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / EMAC0_RMII_RXD1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM2 /

Pin	Type	M467SJHAN Pin Function
		INT2 / PSIO0_CH6 / KPI_ROW0
5	I/O	PB.2 / EADC0_CH2 / EADC1_CH10 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / EMAC0_RMII_CRSDV / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
6	I/O	PB.1 / EADC0_CH1 / EADC1_CH9 / EADC2_CH9 / ACMP3_P0 / EBI_ADR8 / SD0_CLK / EMAC0_RMII_RXERR / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / ACMP2_O / QSPI0_MISO1 / KPI_ROW2
7	I/O	PB.0 / EADC0_CH0 / EADC1_CH8 / EADC2_CH8 / ACMP3_N / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / I2S1_LRCK / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / ACMP3_O / QSPI0_MOSI1 / KPI_ROW3
8	I/O	PA.11 / EADC1_CH7 / EADC2_CH7 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / SD1_DAT3 / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / I2S1_BCLK / TM0_EXT / DAC1_ST / KPI_ROW4
9	I/O	PA.10 / EADC1_CH6 / EADC2_CH6 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / SD1_DAT2 / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / I2S1_MCLK / TM1_EXT / DAC0_ST / SWDH_CLK / KPI_ROW5
10	I/O	PA.9 / EADC1_CH5 / EADC2_CH5 / EBI_MCLK / SC2_DAT / SPI2_MISO / SD1_DAT1 / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / I2S1_DI / TM2_EXT / SWDH_DAT
11	I/O	PA.8 / EADC1_CH4 / EADC2_CH4 / EBI_ALE / SC2_CLK / SPI2_MOSI / SD1_DAT0 / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / I2S1_DO / TM3_EXT / INT4
12	I/O	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / CAN2_RXD / SPI3_I2SMCLK / TAMPER0 / EQEI2_INDEX / TRACE_SWO
13	P	V_{BAT}
14	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / EQEI2_A
15	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / EADC1_ST / I2C4_SDA / EQEI2_B
16	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / ACMP2_O / EADC2_ST
17	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / ACMP3_O
18	I/O	PC.7 / EBI_AD9 / EMAC0_RMII_RXD0 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / UART6_TXD / ACMP2_WLAT / EPWM1_CH2 / BPWM1_CH0 / CAN3_TXD / TM0 / INT3 / KPI_COL3
19	I/O	PC.6 / EBI_AD8 / EMAC0_RMII_RXD1 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / ACMP3_WLAT / EPWM1_CH3 / BPWM1_CH1 / CAN3_RXD / TM1 / INT2 / KPI_COL2
20	I/O	PA.7 / EBI_AD7 / EMAC0_RMII_CRSDV / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
21	I/O	PA.6 / EBI_AD6 / EMAC0_RMII_RXERR / SPI1_SS / SD1_nCD / SC2_CLK / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
22	P	V_{SS}
23	P	V_{DD}
24	A	LDO_CAP
25	I/O	PA.5 / SPIM_D2 / QSPI0_MISO1 / SPI1_I2SMCLK / SD1_CMD / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
26	I/O	PA.4 / SPIM_D3 / QSPI0_MOSI1 / SPI0_I2SMCLK / SD1_CLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
27	I/O	PA.3 / SPIM_SS / QSPI0_SS / SPI0_SS / SD1_DAT3 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
28	I/O	PA.2 / SPIM_CLK / QSPI0_CLK / SPI0_CLK / SD1_DAT2 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / EQEI3_INDEX / PSIO0_CH5
29	I/O	PA.1 / SPIM_MISO / QSPI0_MISO0 / SPI0_MISO / SD1_DAT1 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / CCAP_DATA7 / BPWM0_CH1 / EPWM0_CH4 / EQEI3_A / DAC1_ST / PSIO0_CH6
30	I/O	PA.0 / SPIM_MOSI / QSPI0_MOSI0 / SPI0_MOSI / SD1_DAT0 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / CCAP_DATA6 / BPWM0_CH0 / EPWM0_CH5 / EQEI3_B / DAC0_ST / PSIO0_CH7
31	P	V_{DDIO}
32	I	nRESET
33	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / SC1_DAT / I2S0_DO / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST
34	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / SC1_CLK / I2S0_LRCK / UART2_RXD / I2C0_SDA

Pin	Type	M467SJHAN Pin Function
		/ CAN2_RXD / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / EADC1_ST
35	I/O	PC.5 / EBI_AD5 / SPIM_D2 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / CCAP_DATA5 / QSPI1_SS / I2C3_SMBAL / PSIO0_CH0 / KPI_ROW0
36	I/O	PC.4 / EBI_AD4 / SPIM_D3 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / CCAP_DATA4 / QSPI1_CLK / I2C3_SMBSUS / PSIO0_CH1 / KPI_ROW1
37	I/O	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / SD1_nCD / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / ECAP3_IC0 / USB_VBUS / PSIO0_CH4
38	I/O	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / ECAP3_IC1 / USB_D- / PSIO0_CH5
39	I/O	PA.14 / I2S0_DI / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / EQEI1_B / ECAP3_IC2 / USB_D+ / PSIO0_CH6
40	I/O	PA.15 / I2S0_DO / UART0_RXD / SPIM_MOSI / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / EQEI3_INDEX / USB_OTG_ID / PSIO0_CH7
41	A	HSUSB_VRES
42	P	HSUSB_VDD33
43	I/O	HSUSB_VBUS
44	A	HSUSB_D-
45	P	HSUSB_VSS
46	A	HSUSB_D+
47	A	HSUSB_VDD12_CAP
48	I	HSUSB_ID
49	P	V_{SS}
50	A	LDO_CAP
51	P	V_{DD}
52	I/O	PC.14 / EBI_AD11 / SC1_nCD / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / ETMC_TRACE_CLK / TM1 / USB_VBUS_ST / HSUSB_VBUS_ST
53	I/O	PB.15 / EADC0_CH15 / EADC1_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / CCAP_DATA1 / EPWM0_BRAKE1 / EPWM1_CH0 / ETMC_TRACE_DATA0 / TM0_EXT / USB_VBUS_EN / HSUSB_VBUS_EN / PSIO0_CH0 / KPI_COL0
54	I/O	PB.14 / EADC0_CH14 / EADC1_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / CCAP_DATA0 / EPWM1_CH1 / ETMC_TRACE_DATA1 / TM1_EXT / CLKO / USB_VBUS_ST / PSIO0_CH1 / KPI_COL1
55	I/O	PB.13 / EADC0_CH13 / EADC1_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / CCAP_PIXCLK / EPWM1_CH2 / ETMC_TRACE_DATA2 / TM2_EXT / CAN3_TXD / PSIO0_CH2 / KPI_COL2
56	I/O	PB.12 / EADC0_CH12 / EADC1_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / CCAP_SCLK / EPWM1_CH3 / ETMC_TRACE_DATA3 / TM3_EXT / CAN3_RXD / PSIO0_CH3 / KPI_COL3
57	P	AV_{DD}
58	A	V_{REF}
59	P	AV_{SS}
60	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / EMAC0_RMII_MDC / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / CCAP_SFIELD / HSUSB_VBUS_ST
61	I/O	PB.10 / EADC0_CH10 / ACMP2_P3 / EBI_ADR17 / EMAC0_RMII_MDIO / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / CCAP_VSYNC / HSUSB_VBUS_EN
62	I/O	PB.9 / EADC0_CH9 / ACMP2_P2 / EBI_ADR18 / EMAC0_RMII_TXD0 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / CAN2_TXD / INT7 / CCAP_HSYNC
63	I/O	PB.8 / EADC0_CH8 / ACMP2_P1 / EBI_ADR19 / EMAC0_RMII_TXD1 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / CAN2_RXD / INT6 / EADC2_ST
64	I/O	PB.7 / EADC0_CH7 / EADC2_CH15 / ACMP2_P0 / EBI_nWRL / EMAC0_RMII_TXEN / CAN1_TXD / UART1_TXD / SD1_CMD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O / KPI_COL4

Table 4.1-8 M467SJHAN Multi-function Pin Table

4.1.4.2 M467 Series LQFP128-Pin Multi-function Pin Diagram

Corresponding Part Number: M467KJHAN

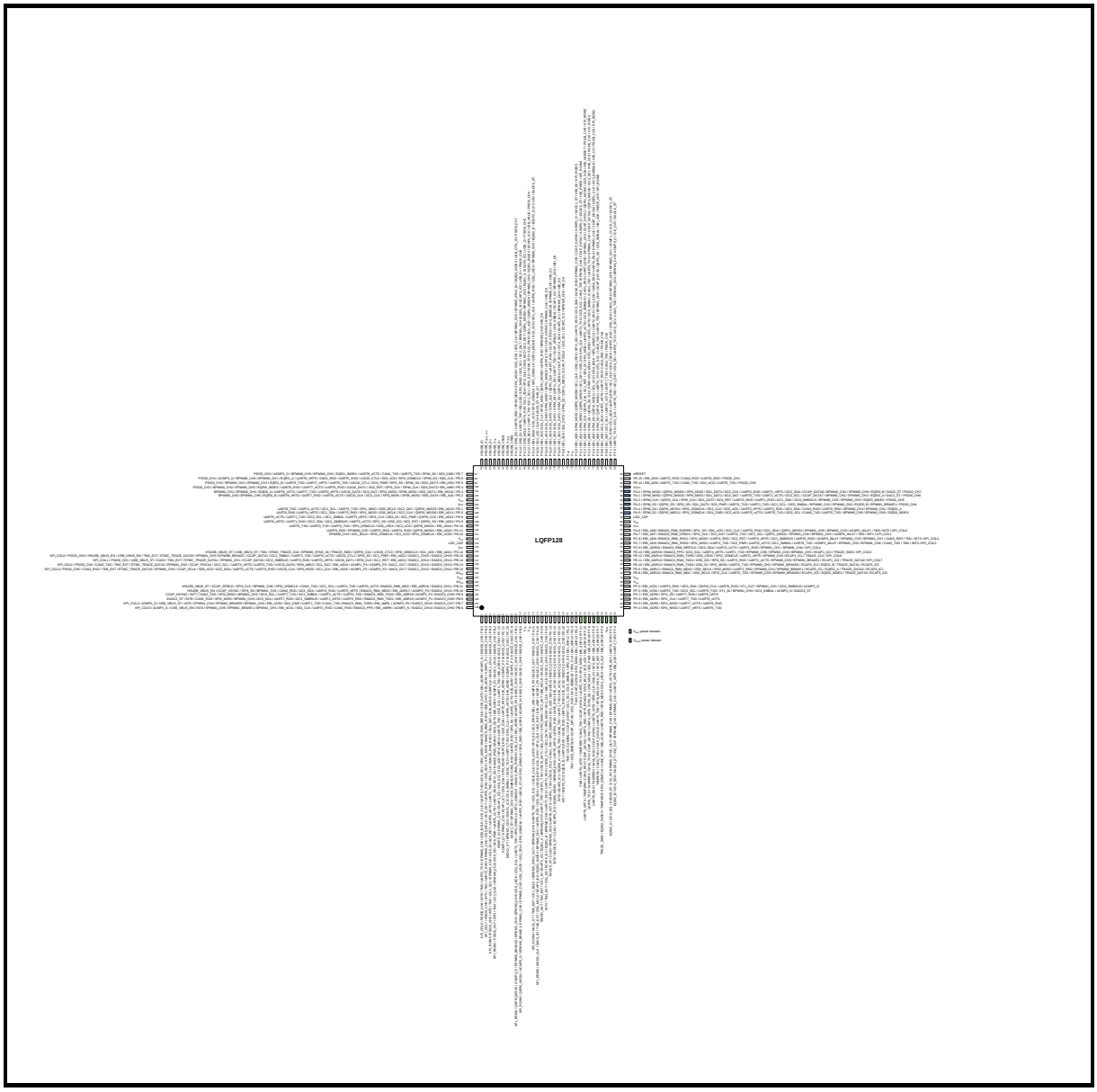


Figure 4.1-24 M467KJHAN Multi-function Pin Diagram

Pin	Type	M467KJHAN Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / EMAC0_RMII_REFCLK / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / EMAC0_RMII_RXD0 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / EADC1_CH11 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / EMAC0_RMII_RXD1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / EADC1_CH10 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / EMAC0_RMII_CRSDV / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PC.12 / EADC2_CH13 / EBI_ADR4 / UART0_TXD / I2C0_SCL / UART6_TXD / SPI3_MISO /

Pin	Type	M467KJHAN Pin Function
		SC0_nCD / I2C4_SCL / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	I/O	PC.11 / EADC2_CH12 / ACMP3_P3 / EBI_ADR5 / UART0_RXD / I2C0_SDA / UART6_RXD / SPI3_MOSI / I2C4_SDA / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	I/O	PC.10 / EADC2_CH11 / ACMP3_P2 / EBI_ADR6 / UART6_nRTS / SPI3_CLK / UART3_TXD / CAN1_TXD / I2C4_SMBAL / ECAP1_IC0 / EPWM1_CH2 / EADC1_ST
8	I/O	PC.9 / EADC2_CH10 / ACMP3_P1 / EBI_ADR7 / UART6_nCTS / SPI3_SS / UART3_RXD / CAN1_RXD / I2C4_SMBSUS / EPWM1_CH3 / EADC1_ST
9	I/O	PB.1 / EADC0_CH1 / EADC1_CH9 / EADC2_CH9 / ACMP3_P0 / EBI_ADR8 / SD0_CLK / EMAC0_RMII_RXERR / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / ACMP2_O / QSPI0_MISO1 / KPI_ROW2
10	I/O	PB.0 / EADC0_CH0 / EADC1_CH8 / EADC2_CH8 / ACMP3_N / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / I2S1_LRCK / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / ACMP3_O / QSPI0_MOSI1 / KPI_ROW3
11	P	V_{SS}
12	P	V_{DD}
13	I/O	PA.11 / EADC1_CH7 / EADC2_CH7 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / SD1_DAT3 / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / I2S1_BCLK / TM0_EXT / DAC1_ST / KPI_ROW4
14	I/O	PA.10 / EADC1_CH6 / EADC2_CH6 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / SD1_DAT2 / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / I2S1_MCLK / TM1_EXT / DAC0_ST / SWDH_CLK / KPI_ROW5
15	I/O	PA.9 / EADC1_CH5 / EADC2_CH5 / EBI_MCLK / SC2_DAT / SPI2_MISO / SD1_DAT1 / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / I2S1_DI / TM2_EXT / SWDH_DAT
16	I/O	PA.8 / EADC1_CH4 / EADC2_CH4 / EBI_ALE / SC2_CLK / SPI2_MOSI / SD1_DAT0 / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / I2S1_DO / TM3_EXT / INT4
17	I/O	PC.13 / EADC1_CH3 / EADC2_CH3 / EBI_ADR10 / SC2_nCD / SPI2_I2SMCLK / CAN1_TXD / USCI0_CTL0 / UART2_TXD / UART8_nCTS / BPWM0_CH4 / CLKO / EADC0_ST
18	I/O	PD.12 / EADC1_CH2 / EADC2_CH2 / EBI_nCS0 / CAN1_RXD / UART2_RXD / UART8_nRTS / BPWM0_CH5 / EQEI0_INDEX / ECAP3_IC0 / CLKO / EADC0_ST / INT5
19	I/O	PD.11 / EADC1_CH1 / EADC2_CH1 / EBI_nCS1 / UART1_TXD / CAN0_TXD / UART8_TXD / EQEI0_A / ECAP3_IC1 / INT6
20	I/O	PD.10 / EADC1_CH0 / EADC2_CH0 / EBI_nCS2 / UART1_RXD / CAN0_RXD / UART8_RXD / EQEI0_B / ECAP3_IC2 / INT7
21	I/O	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / CCAP_DATA7 / I2C3_SMBAL / TM0
22	I/O	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / CCAP_DATA6 / I2C3_SMBSUS / TM1
23	I/O	PG.4 / EBI_ADR13 / SPI2_MISO / CCAP_DATA5 / TM2
24	I/O	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / CCAP_DATA4 / CAN3_TXD / TAMPER5 / UART9_nCTS / TM3
25	I/O	PF.10 / EBI_ADR15 / SC0_nCD / I2S0_BCLK / SPI0_I2SMCLK / UART5_RXD / CCAP_DATA3 / CAN3_RXD / TAMPER4 / UART9_nRTS
26	I/O	PF.9 / EBI_ADR16 / SC0_PWR / I2S0_MCLK / SPI0_SS / UART5_nRTS / CCAP_DATA2 / CAN1_TXD / TAMPER3 / UART9_TXD
27	I/O	PF.8 / EBI_ADR17 / SC0_RST / I2S0_DI / SPI0_CLK / UART5_nCTS / CCAP_DATA1 / CAN1_RXD / TAMPER2 / UART9_RXD
28	I/O	PF.7 / EBI_ADR18 / SC0_DAT / I2S0_DO / SPI0_MISO / UART4_TXD / CCAP_DATA0 / CAN2_TXD / TAMPER1
29	I/O	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / CAN2_RXD / SPI3_I2SMCLK / TAMPER0 / EQEI2_INDEX / TRACE_SWO
30	P	V_{BAT}
31	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / EQEI2_A
32	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / EADC1_ST / I2C4_SDA / EQEI2_B
33	I/O	PH.4 / EBI_ADR3 / SPI1_MISO / UART7_nRTS / UART6_TXD
34	I/O	PH.5 / EBI_ADR2 / SPI1_MOSI / UART7_nCTS / UART6_RXD
35	I/O	PH.6 / EBI_ADR1 / SPI1_CLK / UART7_TXD / UART9_nCTS
36	I/O	PH.7 / EBI_ADR0 / SPI1_SS / UART7_RXD / UART9_nRTS
37	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / UART9_TXD / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / ACMP2_O / EADC2_ST
38	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / UART9_RXD / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / ACMP3_O
39	P	V_{SS}
40	P	V_{DD}
41	I/O	PE.8 / EBI_ADR10 / EMAC0_RMII_MDC / I2S0_BCLK / SPI2_CLK / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / EQEI2_INDEX / TRACE_DATA3 / ECAP3_IC0

Pin	Type	M467KJHAN Pin Function
42	I/O	PE.9 / EBI_ADR11 / EMAC0_RMII_MDIO / I2S0_MCLK / SPI2_MISO / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / EQEI2_A / TRACE_DATA2 / ECAP3_IC1
43	I/O	PE.10 / EBI_ADR12 / EMAC0_RMII_TXD0 / I2S0_DI / SPI2_MOSI / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / EQEI2_B / TRACE_DATA1 / ECAP3_IC2
44	I/O	PE.11 / EBI_ADR13 / EMAC0_RMII_TXD1 / I2S0_DO / SPI2_SS / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0 / KPI_COL7
45	I/O	PE.12 / EBI_ADR14 / EMAC0_RMII_TXEN / I2S0_LRCK / SPI2_I2SMCLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK / KPI_COL6
46	I/O	PE.13 / EBI_ADR15 / EMAC0_PPS / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0 / TRACE_SWO / KPI_COL5
47	I/O	PC.8 / EBI_ADR16 / EMAC0_RMII_REFCLK / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4 / KPI_COL4
48	I/O	PC.7 / EBI_AD9 / EMAC0_RMII_RXD0 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / UART6_TXD / ACMP2_WLAT / EPWM1_CH2 / BPWM1_CH0 / CAN3_TXD / TM0 / INT3 / KPI_COL3
49	I/O	PC.6 / EBI_AD8 / EMAC0_RMII_RXD1 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / ACMP3_WLAT / EPWM1_CH3 / BPWM1_CH1 / CAN3_RXD / TM1 / INT2 / KPI_COL2
50	I/O	PA.7 / EBI_AD7 / EMAC0_RMII_CRSDV / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
51	I/O	PA.6 / EBI_AD6 / EMAC0_RMII_RXERR / SPI1_SS / SD1_nCD / SC2_CLK / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
52	P	V_{SS}
53	P	V_{DD}
54	A	LDO_CAP
55	I/O	PA.5 / SPIM_D2 / QSPI0_MISO1 / SPI1_I2SMCLK / SD1_CMD / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
56	I/O	PA.4 / SPIM_D3 / QSPI0_MOSI1 / SPI0_I2SMCLK / SD1_CLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
57	I/O	PA.3 / SPIM_SS / QSPI0_SS / SPI0_SS / SD1_DAT3 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
58	I/O	PA.2 / SPIM_CLK / QSPI0_CLK / SPI0_CLK / SD1_DAT2 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / EQEI3_INDEX / PSIO0_CH5
59	I/O	PA.1 / SPIM_MISO / QSPI0_MISO0 / SPI0_MISO / SD1_DAT1 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / CCAP_DATA7 / BPWM0_CH1 / EPWM0_CH4 / EQEI3_A / DAC1_ST / PSIO0_CH6
60	I/O	PA.0 / SPIM_MOSI / QSPI0_MOSI0 / SPI0_MOSI / SD1_DAT0 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / CCAP_DATA6 / BPWM0_CH0 / EPWM0_CH5 / EQEI3_B / DAC0_ST / PSIO0_CH7
61	P	V_{DDIO}
62	I/O	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / SD1_nCD / UART6_TXD / PSIO0_CH0
63	I/O	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / UART6_RXD / PSIO0_CH1
64	I	nRESET
65	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / SC1_DAT / I2S0_DO / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST
66	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / SC1_CLK / I2S0_LRCK / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / EADC1_ST
67	I/O	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / UART7_TXD / CAN2_TXD / PSIO0_CH2
68	I/O	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / UART7_RXD / CAN2_RXD / PSIO0_CH3
69	I/O	PC.5 / EBI_AD5 / SPIM_D2 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / CCAP_DATA5 / QSPI1_SS / I2C3_SMBAL / HBI_nCK / PSIO0_CH0 / KPI_ROW0
70	I/O	PC.4 / EBI_AD4 / SPIM_D3 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / CCAP_DATA4 / QSPI1_CLK / I2C3_SMBSUS / HBI_CK / PSIO0_CH1 / KPI_ROW1
71	I/O	PC.3 / EBI_AD3 / SPIM_SS / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / CAN1_TXD / UART3_TXD / EPWM1_CH2 / CCAP_DATA3 / QSPI1_MISO0 / I2C3_SCL / HBI_nCS / PSIO0_CH2 / KPI_ROW2
72	I/O	PC.2 / EBI_AD2 / SPIM_CLK / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / CCAP_DATA2 / QSPI1_MOSI0 / I2C3_SDA / HBI_nRESET / PSIO0_CH3 / KPI_ROW3
73	I/O	PC.1 / EBI_AD1 / SPIM_MISO / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / CCAP_DATA1 / ACMP0_O / EADC0_ST / HBI_RWDS / KPI_ROW4
74	I/O	PC.0 / EBI_AD0 / SPIM_MOSI / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / CCAP_DATA0 / ACMP1_O / EADC1_ST / HBI_D2 /

Pin	Type	M467KJHAN Pin Function
		KPI_ROW5
75	P	V_{SS}
76	P	V_{DD}
77	I/O	PG.9 / EBI_AD0 / SD1_DAT3 / SPIM_D2 / QSPI1_MISO1 / CCAP_PIXCLK / I2C4_SCL / ECAP2_IC0 / BPWM0_CH5 / HBI_D4
78	I/O	PG.10 / EBI_AD1 / SD1_DAT2 / SPIM_D3 / QSPI1_MOSI1 / CCAP_SCLK / I2C4_SDA / ECAP2_IC1 / BPWM0_CH4 / HBI_D3
79	I/O	PG.11 / EBI_AD2 / SD1_DAT1 / SPIM_SS / QSPI1_SS / UART7_TXD / CCAP_SFIELD / I2C4_SMBAL / ECAP2_IC2 / BPWM0_CH3 / HBI_D0
80	I/O	PG.12 / EBI_AD3 / SD1_DAT0 / SPIM_CLK / QSPI1_CLK / UART7_RXD / CCAP_VSYNC / I2C4_SMBSUS / BPWM0_CH2 / HBI_D1
81	I/O	PG.13 / EBI_AD4 / SD1_CMD / SPIM_MISO / QSPI1_MISO0 / UART6_TXD / CCAP_HSYNC / BPWM0_CH1 / HBI_D5
82	I/O	PG.14 / EBI_AD5 / SD1_CLK / SPIM_MOSI / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0 / HBI_D6
83	I/O	PG.15 / SD1_nCD / CLKO / EADC0_ST / HBI_D7
84	I/O	PD.13 / EBI_AD10 / SD0_nCD / SPI0_I2SMCLK / SPI1_I2SMCLK / QSPI1_MOSI0 / SC2_nCD / SD1_CLK / UART6_RXD / I2S1_LRCK / BPWM0_CH0 / EQEI2_B / ECAP2_IC2 / CLKO / EADC0_ST
85	I/O	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / SD1_nCD / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / ECAP3_IC0 / USB_VBUS / PSIO0_CH4
86	I/O	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / ECAP3_IC1 / USB_D- / PSIO0_CH5
87	I/O	PA.14 / I2S0_DI / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / EQEI1_B / ECAP3_IC2 / USB_D+ / PSIO0_CH6
88	I/O	PA.15 / I2S0_DO / UART0_RXD / SPIM_MOSI / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / EQEI3_INDEX / USB_OTG_ID / PSIO0_CH7
89	A	HSUSB_VRES
90	P	HSUSB_VDD33
91	I/O	HSUSB_VBUS
92	A	HSUSB_D-
93	P	HSUSB_VSS
94	A	HSUSB_D+
95	A	HSUSB_VDD12_CAP
96	I	HSUSB_ID
97	I/O	PE.7 / SD0_CMD / SPIM_D2 / UART5_TXD / CAN1_TXD / UART9_nCTS / EQEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / ACMP2_O / PSIO0_CH0
98	I/O	PE.6 / SD0_CLK / SPIM_D3 / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / CAN1_RXD / UART9_nRTS / EQEI1_A / EPWM0_CH1 / BPWM0_CH4 / ACMP3_O / PSIO0_CH1
99	I/O	PE.5 / EBI_nRD / SD0_DAT3 / SPIM_SS / SPI3_SS / SC0_PWR / USCI0_CTL1 / UART6_TXD / UART7_nRTS / UART9_TXD / EQEI1_B / EPWM0_CH2 / BPWM0_CH3 / PSIO0_CH2
100	I/O	PE.4 / EBI_nWR / SD0_DAT2 / SPIM_CLK / SPI3_CLK / SC0_RST / USCI0_DAT1 / UART6_RXD / UART7_nCTS / UART9_RXD / EQEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / PSIO0_CH3
101	I/O	PE.3 / EBI_MCLK / SD0_DAT1 / SPIM_MISO / SPI3_MISO / SC0_DAT / USCI0_DAT0 / UART6_nRTS / UART7_TXD / UART8_nCTS / EQEI0_A / EPWM0_CH4 / BPWM0_CH1
102	I/O	PE.2 / EBI_ALE / SD0_DAT0 / SPIM_MOSI / SPI3_MOSI / SC0_CLK / USCI0_CLK / UART6_nCTS / UART7_RXD / UART8_nRTS / EQEI0_B / EPWM0_CH5 / BPWM0_CH0
103	P	V_{SS}
104	P	V_{DD}
105	I/O	PE.1 / EBI_AD10 / QSPI0_MISO0 / SC2_DAT / I2S0_BCLK / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS / UART8_TXD
106	I/O	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SC2_CLK / I2S0_MCLK / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS / UART8_RXD
107	I/O	PH.8 / EBI_AD12 / QSPI0_CLK / SC2_PWR / I2S0_DI / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD / UART9_nCTS
108	I/O	PH.9 / EBI_AD13 / QSPI0_SS / SC2_RST / I2S0_DO / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD / UART9_nRTS
109	I/O	PH.10 / EBI_AD14 / QSPI0_MISO1 / SC2_nCD / I2S0_LRCK / SPI1_I2SMCLK / UART4_TXD / UART0_TXD / UART9_TXD
110	I/O	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5 / UART9_RXD
111	I/O	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SC1_nCD / SPI0_I2SMCLK / I2S1_BCLK / EPWM0_CH4
112	P	V_{SS}
113	A	LDO_CAP
114	P	V_{DD}
115	I/O	PC.14 / EBI_AD11 / SC1_nCD / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / ETMC_TRACE_CLK / TM1 / USB_VBUS_ST / HSUSB_VBUS_ST
116	I/O	PB.15 / EADC0_CH15 / EADC1_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 /

Pin	Type	M467KJHAN Pin Function
		UART0_nCTS / UART3_TXD / I2C2_SMBAL / CCAP_DATA1 / EPWM0_BRAKE1 / EPWM1_CH0 / ETMC_TRACE_DATA0 / TM0_EXT / USB_VBUS_EN / HSUSB_VBUS_EN / PSIO0_CH0 / KPI_COL0
117	I/O	PB.14 / EADC0_CH14 / EADC1_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / CCAP_DATA0 / EPWM1_CH1 / ETMC_TRACE_DATA1 / TM1_EXT / CLKO / USB_VBUS_ST / PSIO0_CH1 / KPI_COL1
118	I/O	PB.13 / EADC0_CH13 / EADC1_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / CCAP_PIXCLK / EPWM1_CH2 / ETMC_TRACE_DATA2 / TM2_EXT / CAN3_TXD / PSIO0_CH2 / KPI_COL2
119	I/O	PB.12 / EADC0_CH12 / EADC1_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / CCAP_SCLK / EPWM1_CH3 / ETMC_TRACE_DATA3 / TM3_EXT / CAN3_RXD / PSIO0_CH3 / KPI_COL3
120	P	AV_{DD}
121	A	V_{REF}
122	P	AV_{SS}
123	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / EMAC0_RMII_MDC / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / CCAP_SFIELD / HSUSB_VBUS_ST
124	I/O	PB.10 / EADC0_CH10 / ACMP2_P3 / EBI_ADR17 / EMAC0_RMII_MDIO / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / CCAP_VSYNC / HSUSB_VBUS_EN
125	I/O	PB.9 / EADC0_CH9 / ACMP2_P2 / EBI_ADR18 / EMAC0_RMII_TXD0 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / CAN2_TXD / INT7 / CCAP_HSYNC
126	I/O	PB.8 / EADC0_CH8 / ACMP2_P1 / EBI_ADR19 / EMAC0_RMII_TXD1 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / CAN2_RXD / INT6 / EADC2_ST
127	I/O	PB.7 / EADC0_CH7 / EADC2_CH15 / ACMP2_P0 / EBI_nWRL / EMAC0_RMII_TXEN / CAN1_TXD / UART1_TXD / SD1_CMD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O / KPI_COL4
128	I/O	PB.6 / EADC0_CH6 / EADC2_CH14 / ACMP2_N / EBI_nWRH / EMAC0_PPS / CAN1_RXD / UART1_RXD / SD1_CLK / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O / KPI_COL5

Table 4.1-9 M467KJHAN Multi-function Pin Table

4.1.4.3 M467 Series LQFP144-Pin Multi-function Pin Diagram

Corresponding Part Number: M467JJHAN

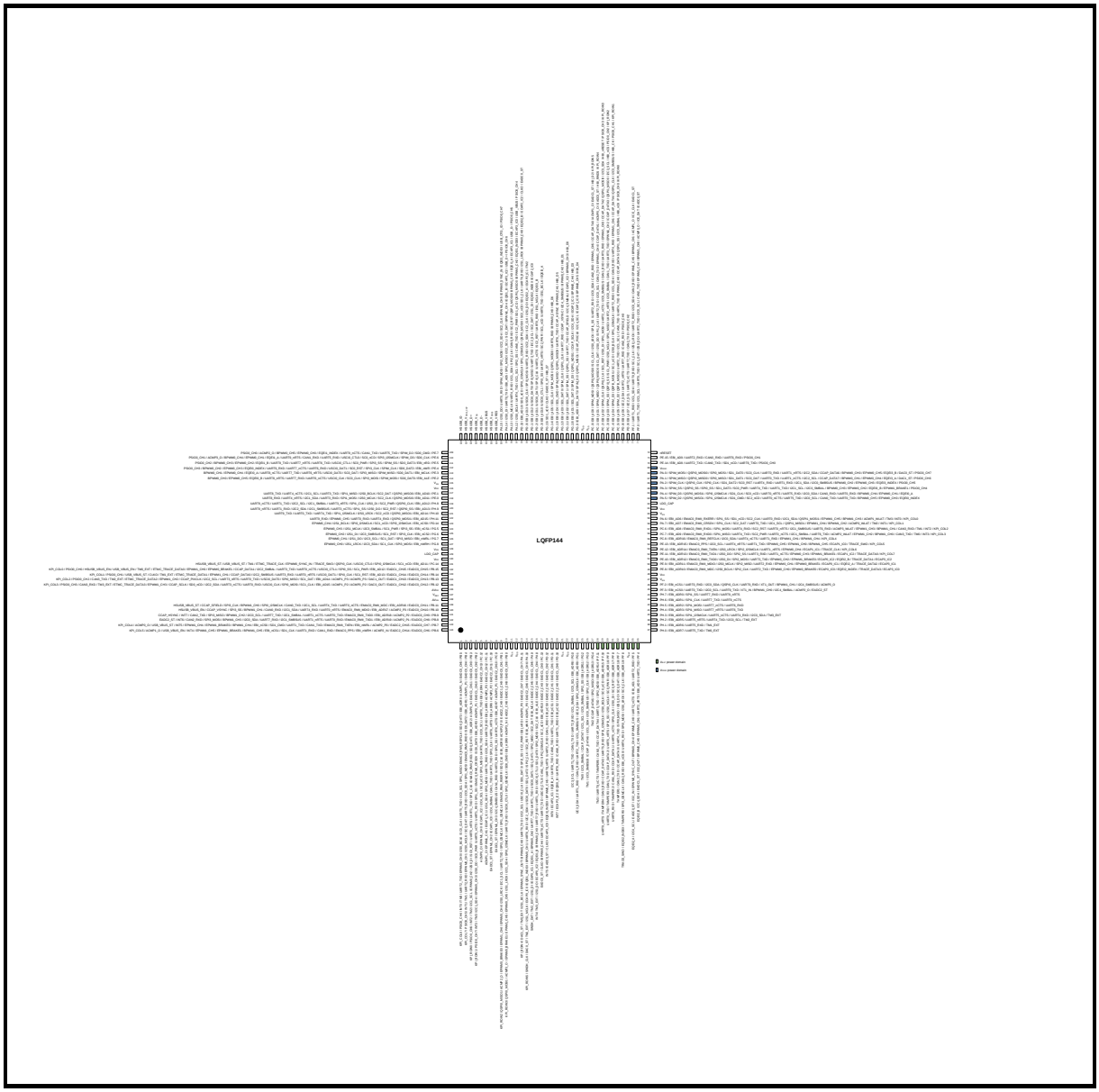


Figure 4.1-25 M467JJHAN Multi-function Pin Diagram

Pin	Type	M467JJHAN Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / EMAC0_RMII_REFCLK / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / EMAC0_RMII_RXD0 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / EADC1_CH11 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / EMAC0_RMII_RXD1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / EADC1_CH10 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / EMAC0_RMII_CRSDV / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PC.12 / EADC2_CH13 / EBI_ADR4 / UART0_TXD / I2C0_SCL / UART6_TXD / SPI3_MISO /

Pin	Type	M467JJHAN Pin Function
		SC0_nCD / I2C4_SCL / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	I/O	PC.11 / EADC2_CH12 / ACMP3_P3 / EBI_ADR5 / UART0_RXD / I2C0_SDA / UART6_RXD / SPI3_MOSI / I2C4_SDA / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	I/O	PC.10 / EADC2_CH11 / ACMP3_P2 / EBI_ADR6 / UART6_nRTS / SPI3_CLK / UART3_TXD / CAN1_TXD / I2C4_SMBAL / ECAP1_IC0 / EPWM1_CH2 / EADC1_ST
8	I/O	PC.9 / EADC2_CH10 / ACMP3_P1 / EBI_ADR7 / UART6_nCTS / SPI3_SS / UART3_RXD / CAN1_RXD / I2C4_SMBSUS / EPWM1_CH3 / EADC1_ST
9	I/O	PB.1 / EADC0_CH1 / EADC1_CH9 / EADC2_CH9 / ACMP3_P0 / EBI_ADR8 / SD0_CLK / EMAC0_RMII_RXERR / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / ACMP2_O / QSPI0_MISO1 / KPI_ROW2
10	I/O	PB.0 / EADC0_CH0 / EADC1_CH8 / EADC2_CH8 / ACMP3_N / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / I2S1_LRCK / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / ACMP3_O / QSPI0_MOSI1 / KPI_ROW3
11	P	V_{SS}
12	P	V_{DD}
13	I/O	PA.11 / EADC1_CH7 / EADC2_CH7 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / SD1_DAT3 / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / I2S1_BCLK / TM0_EXT / DAC1_ST / KPI_ROW4
14	I/O	PA.10 / EADC1_CH6 / EADC2_CH6 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / SD1_DAT2 / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / I2S1_MCLK / TM1_EXT / DAC0_ST / SWDH_CLK / KPI_ROW5
15	I/O	PA.9 / EADC1_CH5 / EADC2_CH5 / EBI_MCLK / SC2_DAT / SPI2_MISO / SD1_DAT1 / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / I2S1_DI / TM2_EXT / SWDH_DAT
16	I/O	PA.8 / EADC1_CH4 / EADC2_CH4 / EBI_ALE / SC2_CLK / SPI2_MOSI / SD1_DAT0 / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / I2S1_DO / TM3_EXT / INT4
17	I/O	PC.13 / EADC1_CH3 / EADC2_CH3 / EBI_ADR10 / SC2_nCD / SPI2_I2SMCLK / CAN1_TXD / USCI0_CTL0 / UART2_TXD / UART8_nCTS / BPWM0_CH4 / CLKO / EADC0_ST
18	I/O	PD.12 / EADC1_CH2 / EADC2_CH2 / EBI_nCS0 / CAN1_RXD / UART2_RXD / UART8_nRTS / BPWM0_CH5 / EQEI0_INDEX / ECAP3_IC0 / CLKO / EADC0_ST / INT5
19	I/O	PD.11 / EADC1_CH1 / EADC2_CH1 / EBI_nCS1 / UART1_TXD / CAN0_TXD / UART8_TXD / EQEI0_A / ECAP3_IC1 / INT6
20	I/O	PD.10 / EADC1_CH0 / EADC2_CH0 / EBI_nCS2 / UART1_RXD / CAN0_RXD / UART8_RXD / EQEI0_B / ECAP3_IC2 / INT7
21	P	V_{SS}
22	P	V_{DD}
23	I/O	PG.0 / EBI_ADR8 / I2C0_SCL / I2C1_SMBAL / UART2_RXD / CAN1_TXD / UART1_TXD / I2C3_SCL
24	I/O	PG.1 / EBI_ADR9 / SPI2_I2SMCLK / I2C0_SDA / I2C1_SMBSUS / UART2_TXD / CAN1_RXD / UART1_RXD / I2C3_SDA
25	I/O	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / CCAP_DATA7 / I2C3_SMBAL / TM0
26	I/O	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / CCAP_DATA6 / I2C3_SMBSUS / TM1
27	I/O	PG.4 / EBI_ADR13 / SPI2_MISO / CCAP_DATA5 / TM2
28	I/O	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / CCAP_DATA4 / CAN3_TXD / TAMPER5 / UART9_nCTS / TM3
29	I/O	PF.10 / EBI_ADR15 / SC0_nCD / I2S0_BCLK / SPI0_I2SMCLK / UART5_RXD / CCAP_DATA3 / CAN3_RXD / TAMPER4 / UART9_nRTS
30	I/O	PF.9 / EBI_ADR16 / SC0_PWR / I2S0_MCLK / SPI0_SS / UART5_nRTS / CCAP_DATA2 / CAN1_TXD / TAMPER3 / UART9_TXD
31	I/O	PF.8 / EBI_ADR17 / SC0_RST / I2S0_DI / SPI0_CLK / UART5_nCTS / CCAP_DATA1 / CAN1_RXD / TAMPER2 / UART9_RXD
32	I/O	PF.7 / EBI_ADR18 / SC0_DAT / I2S0_DO / SPI0_MISO / UART4_TXD / CCAP_DATA0 / CAN2_TXD / TAMPER1
33	I/O	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / CAN2_RXD / SPI3_I2SMCLK / TAMPER0 / EQEI2_INDEX / TRACE_SWO
34	P	V_{BAT}
35	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / EQEI2_A
36	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / EADC1_ST / I2C4_SDA / EQEI2_B
37	I/O	PH.0 / EBI_ADR7 / UART5_TXD / TM0_EXT
38	I/O	PH.1 / EBI_ADR6 / UART5_RXD / TM1_EXT
39	I/O	PH.2 / EBI_ADR5 / UART5_nRTS / UART4_TXD / I2C0_SCL / TM2_EXT
40	I/O	PH.3 / EBI_ADR4 / SPI1_I2SMCLK / UART5_nCTS / UART4_RXD / I2C0_SDA / TM3_EXT
41	I/O	PH.4 / EBI_ADR3 / SPI1_MISO / UART7_nRTS / UART6_TXD

Pin	Type	M467JJHAN Pin Function
42	I/O	PH.5 / EBI_ADR2 / SPI1_MOSI / UART7_nCTS / UART6_RXD
43	I/O	PH.6 / EBI_ADR1 / SPI1_CLK / UART7_TXD / UART9_nCTS
44	I/O	PH.7 / EBI_ADR0 / SPI1_SS / UART7_RXD / UART9_nRTS
45	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / UART9_TXD / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / ACMP2_O / EADC2_ST
46	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / UART9_RXD / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / ACMP3_O
47	P	V_{SS}
48	P	V_{DD}
49	I/O	PE.8 / EBI_ADR10 / EMAC0_RMII_MDC / I2S0_BCLK / SPI2_CLK / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / EQEI2_INDEX / TRACE_DATA3 / ECAP3_IC0
50	I/O	PE.9 / EBI_ADR11 / EMAC0_RMII_MDIO / I2S0_MCLK / SPI2_MISO / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / EQEI2_A / TRACE_DATA2 / ECAP3_IC1
51	I/O	PE.10 / EBI_ADR12 / EMAC0_RMII_TXD0 / I2S0_DI / SPI2_MOSI / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / EQEI2_B / TRACE_DATA1 / ECAP3_IC2
52	I/O	PE.11 / EBI_ADR13 / EMAC0_RMII_TXD1 / I2S0_DO / SPI2_SS / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0 / KPI_COL7
53	I/O	PE.12 / EBI_ADR14 / EMAC0_RMII_TXEN / I2S0_LRCK / SPI2_I2SMCLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK / KPI_COL6
54	I/O	PE.13 / EBI_ADR15 / EMAC0_PPS / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0 / TRACE_SWO / KPI_COL5
55	I/O	PC.8 / EBI_ADR16 / EMAC0_RMII_REFCLK / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4 / KPI_COL4
56	I/O	PC.7 / EBI_AD9 / EMAC0_RMII_RXD0 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / UART6_TXD / ACMP2_WLAT / EPWM1_CH2 / BPWM1_CH0 / CAN3_TXD / TM0 / INT3 / KPI_COL3
57	I/O	PC.6 / EBI_AD8 / EMAC0_RMII_RXD1 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / ACMP3_WLAT / EPWM1_CH3 / BPWM1_CH1 / CAN3_RXD / TM1 / INT2 / KPI_COL2
58	I/O	PA.7 / EBI_AD7 / EMAC0_RMII_CRSDV / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
59	I/O	PA.6 / EBI_AD6 / EMAC0_RMII_RXERR / SPI1_SS / SD1_nCD / SC2_CLK / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
60	P	V_{SS}
61	P	V_{DD}
62	A	LDO_CAP
63	I/O	PA.5 / SPIM_D2 / QSPI0_MISO1 / SPI1_I2SMCLK / SD1_CMD / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
64	I/O	PA.4 / SPIM_D3 / QSPI0_MOSI1 / SPI0_I2SMCLK / SD1_CLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
65	I/O	PA.3 / SPIM_SS / QSPI0_SS / SPI0_SS / SD1_DAT3 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
66	I/O	PA.2 / SPIM_CLK / QSPI0_CLK / SPI0_CLK / SD1_DAT2 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / EQEI3_INDEX / PSIO0_CH5
67	I/O	PA.1 / SPIM_MISO / QSPI0_MISO0 / SPI0_MISO / SD1_DAT1 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / CCAP_DATA7 / BPWM0_CH1 / EPWM0_CH4 / EQEI3_A / DAC1_ST / PSIO0_CH6
68	I/O	PA.0 / SPIM_MOSI / QSPI0_MOSI0 / SPI0_MOSI / SD1_DAT0 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / CCAP_DATA6 / BPWM0_CH0 / EPWM0_CH5 / EQEI3_B / DAC0_ST / PSIO0_CH7
69	P	V_{DDIO}
70	I/O	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / SD1_nCD / UART6_TXD / PSIO0_CH0
71	I/O	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / UART6_RXD / PSIO0_CH1
72	I	nRESET
73	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / SC1_DAT / I2S0_DO / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST
74	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / SC1_CLK / I2S0_LRCK / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / EADC1_ST
75	I/O	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / UART7_TXD / CAN2_TXD / PSIO0_CH2
76	I/O	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / UART7_RXD / CAN2_RXD / PSIO0_CH3
77	I/O	PC.5 / EBI_AD5 / SPIM_D2 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / CCAP_DATA5 / QSPI1_SS / I2C3_SMBAL / HBI_nCK / PSIO0_CH0 / KPI_ROW0
78	I/O	PC.4 / EBI_AD4 / SPIM_D3 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / CCAP_DATA4 / QSPI1_CLK / I2C3_SMBSUS / HBI_CK / PSIO0_CH1 / KPI_ROW1

Pin	Type	M467JJHAN Pin Function
79	I/O	PC.3 / EBI_AD3 / SPIM_SS / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / CAN1_TXD / UART3_TXD / EPWM1_CH2 / CCAP_DATA3 / QSPI1_MISO0 / I2C3_SCL / HBI_nCS / PSIO0_CH2 / KPI_ROW2
80	I/O	PC.2 / EBI_AD2 / SPIM_CLK / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / CCAP_DATA2 / QSPI1_MOSI0 / I2C3_SDA / HBI_nRESET / PSIO0_CH3 / KPI_ROW3
81	I/O	PC.1 / EBI_AD1 / SPIM_MISO / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / CCAP_DATA1 / ACMP0_O / EADC0_ST / HBI_RWDS / KPI_ROW4
82	I/O	PC.0 / EBI_AD0 / SPIM_MOSI / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / CCAP_DATA0 / ACMP1_O / EADC1_ST / HBI_D2 / KPI_ROW5
83	P	V_{SS}
84	P	V_{DD}
85	I/O	PG.9 / EBI_AD0 / SD1_DAT3 / SPIM_D2 / QSPI1_MISO1 / CCAP_PIXCLK / I2C4_SCL / ECAP2_IC0 / BPWM0_CH5 / HBI_D4
86	I/O	PG.10 / EBI_AD1 / SD1_DAT2 / SPIM_D3 / QSPI1_MOSI1 / CCAP_SCLK / I2C4_SDA / ECAP2_IC1 / BPWM0_CH4 / HBI_D3
87	I/O	PG.11 / EBI_AD2 / SD1_DAT1 / SPIM_SS / QSPI1_SS / UART7_TXD / CCAP_SFIELD / I2C4_SMBAL / ECAP2_IC2 / BPWM0_CH3 / HBI_D0
88	I/O	PG.12 / EBI_AD3 / SD1_DAT0 / SPIM_CLK / QSPI1_CLK / UART7_RXD / CCAP_VSYNC / I2C4_SMBSUS / BPWM0_CH2 / HBI_D1
89	I/O	PG.13 / EBI_AD4 / SD1_CMD / SPIM_MISO / QSPI1_MISO0 / UART6_TXD / CCAP_HSYNC / BPWM0_CH1 / HBI_D5
90	I/O	PG.14 / EBI_AD5 / SD1_CLK / SPIM_MOSI / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0 / HBI_D6
91	I/O	PG.15 / SD1_nCD / CLKO / EADC0_ST / HBI_D7
92	I/O	PD.3 / EBI_AD10 / USCI0_CTL1 / SPI0_SS / UART3_nRTS / SC2_PWR / SC1_nCD / UART0_TXD / I2S1_BCLK / EQEI3_A
93	I/O	PD.2 / EBI_AD11 / USCI0_DAT1 / SPI0_CLK / UART3_nCTS / SC2_RST / UART0_RXD / I2S1_MCLK / EQEI3_B
94	I/O	PD.1 / EBI_AD12 / USCI0_DAT0 / SPI0_MISO / UART3_TXD / I2C2_SCL / SC2_DAT / I2S1_DI / EQEI2_INDEX / ECAP2_IC0
95	I/O	PD.0 / EBI_AD13 / USCI0_CLK / SPI0_MOSI / UART3_RXD / I2C2_SDA / SC2_CLK / I2S1_DO / EQEI2_A / ECAP2_IC1 / TM2
96	I/O	PD.13 / EBI_AD10 / SD0_nCD / SPI0_I2SMCLK / SPI1_I2SMCLK / QSPI1_MOSI0 / SC2_nCD / SD1_CLK / UART6_RXD / I2S1_LRCK / BPWM0_CH0 / EQEI2_B / ECAP2_IC2 / CLKO / EADC0_ST
97	I/O	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / SD1_nCD / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / ECAP3_IC0 / USB_VBUS / PSIO0_CH4
98	I/O	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / ECAP3_IC1 / USB_D- / PSIO0_CH5
99	I/O	PA.14 / I2S0_DI / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / EQEI1_B / ECAP3_IC2 / USB_D+ / PSIO0_CH6
100	I/O	PA.15 / I2S0_DO / UART0_RXD / SPIM_MOSI / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / EQEI3_INDEX / USB_OTG_ID / PSIO0_CH7
101	A	HSUSB_VRES
102	P	HSUSB_VDD33
103	I/O	HSUSB_VBUS
104	A	HSUSB_D-
105	P	HSUSB_VSS
106	A	HSUSB_D+
107	A	HSUSB_VDD12_CAP
108	I	HSUSB_ID
109	I/O	PE.7 / SD0_CMD / SPIM_D2 / UART5_TXD / CAN1_TXD / UART9_nCTS / EQEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / ACMP2_O / PSIO0_CH0
110	I/O	PE.6 / SD0_CLK / SPIM_D3 / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / CAN1_RXD / UART9_nRTS / EQEI1_A / EPWM0_CH1 / BPWM0_CH4 / ACMP3_O / PSIO0_CH1
111	I/O	PE.5 / EBI_nRD / SD0_DAT3 / SPIM_SS / SPI3_SS / SC0_PWR / USCI0_CTL1 / UART6_TXD / UART7_nRTS / UART9_TXD / EQEI1_B / EPWM0_CH2 / BPWM0_CH3 / PSIO0_CH2
112	I/O	PE.4 / EBI_nWR / SD0_DAT2 / SPIM_CLK / SPI3_CLK / SC0_RST / USCI0_DAT1 / UART6_RXD / UART7_nCTS / UART9_RXD / EQEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / PSIO0_CH3
113	I/O	PE.3 / EBI_MCLK / SD0_DAT1 / SPIM_MISO / SPI3_MISO / SC0_DAT / USCI0_DAT0 / UART6_nRTS / UART7_TXD / UART8_nCTS / EQEI0_A / EPWM0_CH4 / BPWM0_CH1
114	I/O	PE.2 / EBI_ALE / SD0_DAT0 / SPIM_MOSI / SPI3_MOSI / SC0_CLK / USCI0_CLK / UART6_nCTS / UART7_RXD / UART8_nRTS / EQEI0_B / EPWM0_CH5 / BPWM0_CH0
115	P	V_{SS}

Pin	Type	M467JJHAN Pin Function
116	P	V_{DD}
117	I/O	PE.1 / EBI_AD10 / QSPI0_MISO0 / SC2_DAT / I2S0_BCLK / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS / UART8_TXD
118	I/O	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SC2_CLK / I2S0_MCLK / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS / UART8_RXD
119	I/O	PH.8 / EBI_AD12 / QSPI0_CLK / SC2_PWR / I2S0_DI / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD / UART9_nCTS
120	I/O	PH.9 / EBI_AD13 / QSPI0_SS / SC2_RST / I2S0_DO / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD / UART9_nRTS
121	I/O	PH.10 / EBI_AD14 / QSPI0_MISO1 / SC2_nCD / I2S0_LRCK / SPI1_I2SMCLK / UART4_TXD / UART0_TXD / UART9_TXD
122	I/O	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5 / UART9_RXD
123	I/O	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SC1_nCD / SPI0_I2SMCLK / I2S1_BCLK / EPWM0_CH4
124	I/O	PG.5 / EBI_nCS1 / SPI3_SS / SC1_PWR / I2C3_SMBAL / I2S1_MCLK / EPWM0_CH3
125	I/O	PG.6 / EBI_nCS2 / SPI3_CLK / SC1_RST / I2C3_SMBSUS / I2S1_DI / EPWM0_CH2
126	I/O	PG.7 / EBI_nWRL / SPI3_MISO / SC1_DAT / I2C3_SCL / I2S1_DO / EPWM0_CH1
127	I/O	PG.8 / EBI_nWRH / SPI3_MOSI / SC1_CLK / I2C3_SDA / I2S1_LRCK / EPWM0_CH0
128	P	V_{SS}
129	A	LDO_CAP
130	P	V_{DD}
131	I/O	PC.14 / EBI_AD11 / SC1_nCD / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / ETMC_TRACE_CLK / TM1 / USB_VBUS_ST / HSUSB_VBUS_ST
132	I/O	PB.15 / EADC0_CH15 / EADC1_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / CCAP_DATA1 / EPWM0_BRAKE1 / EPWM1_CH0 / ETMC_TRACE_DATA0 / TM0_EXT / USB_VBUS_EN / HSUSB_VBUS_EN / PSIO0_CH0 / KPI_COL0
133	I/O	PB.14 / EADC0_CH14 / EADC1_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / CCAP_DATA0 / EPWM1_CH1 / ETMC_TRACE_DATA1 / TM1_EXT / CLKO / USB_VBUS_ST / PSIO0_CH1 / KPI_COL1
134	I/O	PB.13 / EADC0_CH13 / EADC1_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / CCAP_PIXCLK / EPWM1_CH2 / ETMC_TRACE_DATA2 / TM2_EXT / CAN3_TXD / PSIO0_CH2 / KPI_COL2
135	I/O	PB.12 / EADC0_CH12 / EADC1_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / CCAP_SCLK / EPWM1_CH3 / ETMC_TRACE_DATA3 / TM3_EXT / CAN3_RXD / PSIO0_CH3 / KPI_COL3
136	P	AV_{DD}
137	A	V_{REF}
138	P	AV_{SS}
139	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / EMAC0_RMII_MDC / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / CCAP_SFIELD / HSUSB_VBUS_ST
140	I/O	PB.10 / EADC0_CH10 / ACMP2_P3 / EBI_ADR17 / EMAC0_RMII_MDIO / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / CCAP_VSYNC / HSUSB_VBUS_EN
141	I/O	PB.9 / EADC0_CH9 / ACMP2_P2 / EBI_ADR18 / EMAC0_RMII_TXD0 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / CAN2_TXD / INT7 / CCAP_HSYNC
142	I/O	PB.8 / EADC0_CH8 / ACMP2_P1 / EBI_ADR19 / EMAC0_RMII_TXD1 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / CAN2_RXD / INT6 / EADC2_ST
143	I/O	PB.7 / EADC0_CH7 / EADC2_CH15 / ACMP2_P0 / EBI_nWRL / EMAC0_RMII_TXEN / CAN1_TXD / UART1_TXD / SD1_CMD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O / KPI_COL4
144	I/O	PB.6 / EADC0_CH6 / EADC2_CH14 / ACMP2_N / EBI_nWRH / EMAC0_PPS / CAN1_RXD / UART1_RXD / SD1_CLK / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O / KPI_COL5

Table 4.1-10 M467JJHAN Multi-function Pin Table

4.1.4.4 M467 Series LQFP176-Pin Multi-function Pin Diagram

Corresponding Part Number: M467HJHAN

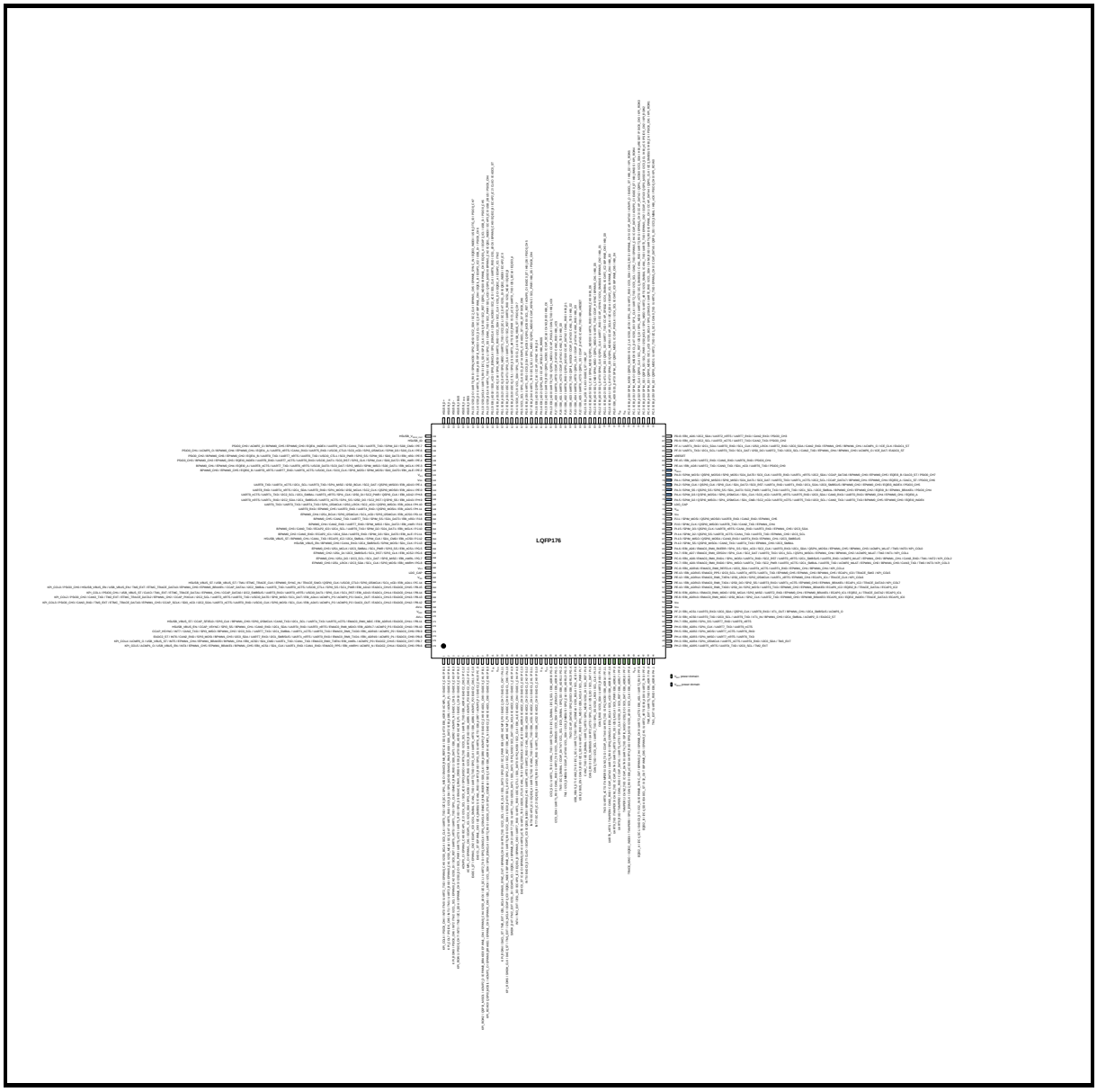


Figure 4.1-26 M467HJHAN Multi-function Pin Diagram

Pin	Type	M467HJHAN Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / EMAC0_RMII_REFCLK / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / EMAC0_RMII_RXD0 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / EADC1_CH11 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / EMAC0_RMII_RXD1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / EADC1_CH10 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / EMAC0_RMII_CRSDV / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PC.12 / EADC2_CH13 / EBI_ADR4 / UART0_TXD / I2C0_SCL / UART6_TXD / SPI3_MISO /

Pin	Type	M467HJHAN Pin Function
		SC0_nCD / I2C4_SCL / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	I/O	PC.11 / EADC2_CH12 / ACMP3_P3 / EBI_ADR5 / UART0_RXD / I2C0_SDA / UART6_RXD / SPI3_MOSI / I2C4_SDA / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	I/O	PC.10 / EADC2_CH11 / ACMP3_P2 / EBI_ADR6 / UART6_nRTS / SPI3_CLK / UART3_TXD / CAN1_TXD / I2C4_SMBAL / ECAP1_IC0 / EPWM1_CH2 / EADC1_ST
8	I/O	PC.9 / EADC2_CH10 / ACMP3_P1 / EBI_ADR7 / UART6_nCTS / SPI3_SS / UART3_RXD / CAN1_RXD / I2C4_SMBSUS / EPWM1_CH3 / EADC1_ST
9	I/O	PB.1 / EADC0_CH1 / EADC1_CH9 / EADC2_CH9 / ACMP3_P0 / EBI_ADR8 / SD0_CLK / EMAC0_RMII_RXERR / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / ACMP2_O / QSPI0_MISO1 / KPI_ROW2
10	I/O	PB.0 / EADC0_CH0 / EADC1_CH8 / EADC2_CH8 / ACMP3_N / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / I2S1_LRCK / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / ACMP3_O / QSPI0_MOSI1 / KPI_ROW3
11	P	V_{SS}
12	P	V_{DD}
13	I/O	PA.11 / EADC1_CH7 / EADC2_CH7 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / SD1_DAT3 / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / I2S1_BCLK / TM0_EXT / DAC1_ST / KPI_ROW4
14	I/O	PA.10 / EADC1_CH6 / EADC2_CH6 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / SD1_DAT2 / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / I2S1_MCLK / TM1_EXT / DAC0_ST / SWDH_CLK / KPI_ROW5
15	I/O	PA.9 / EADC1_CH5 / EADC2_CH5 / EBI_MCLK / SC2_DAT / SPI2_MISO / SD1_DAT1 / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / I2S1_DI / TM2_EXT / SWDH_DAT
16	I/O	PA.8 / EADC1_CH4 / EADC2_CH4 / EBI_ALE / SC2_CLK / SPI2_MOSI / SD1_DAT0 / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / I2S1_DO / TM3_EXT / INT4
17	I/O	PC.13 / EADC1_CH3 / EADC2_CH3 / EBI_ADR10 / SC2_nCD / SPI2_I2SMCLK / CAN1_TXD / USCI0_CTL0 / UART2_TXD / UART8_nCTS / BPWM0_CH4 / CLKO / EADC0_ST
18	I/O	PD.12 / EADC1_CH2 / EADC2_CH2 / EBI_nCS0 / CAN1_RXD / UART2_RXD / UART8_nRTS / BPWM0_CH5 / EQEI0_INDEX / ECAP3_IC0 / CLKO / EADC0_ST / INT5
19	I/O	PD.11 / EADC1_CH1 / EADC2_CH1 / EBI_nCS1 / UART1_TXD / CAN0_TXD / UART8_TXD / EQEI0_A / ECAP3_IC1 / INT6
20	I/O	PD.10 / EADC1_CH0 / EADC2_CH0 / EBI_nCS2 / UART1_RXD / CAN0_RXD / UART8_RXD / EQEI0_B / ECAP3_IC2 / INT7
21	P	V_{SS}
22	P	V_{DD}
23	I/O	PG.0 / EBI_ADR8 / I2C0_SCL / I2C1_SMBAL / UART2_RXD / CAN1_TXD / UART1_TXD / I2C3_SCL
24	I/O	PG.1 / EBI_ADR9 / SPI2_I2SMCLK / I2C0_SDA / I2C1_SMBSUS / UART2_TXD / CAN1_RXD / UART1_RXD / I2C3_SDA
25	I/O	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / CCAP_DATA7 / I2C3_SMBAL / TM0
26	I/O	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / CCAP_DATA6 / I2C3_SMBSUS / TM1
27	I/O	PG.4 / EBI_ADR13 / SPI2_MISO / CCAP_DATA5 / TM2
28	I/O	PI.6 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_TXD / I2C1_SCL / CAN3_TXD / USB_VBUS_ST
29	I/O	PI.7 / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_RXD / I2C1_SDA / CAN3_RXD / USB_VBUS_EN
30	I/O	PI.8 / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nRTS / I2C0_SMBAL / CAN2_TXD
31	I/O	PI.9 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_nCTS / I2C0_SMBSUS / CAN2_RXD
32	I/O	PI.10 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_TXD / I2C0_SCL / CAN3_TXD
33	I/O	PI.11 / UART2_RXD / I2C0_SDA / CAN3_RXD
34	I/O	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / CCAP_DATA4 / CAN3_TXD / TAMPER5 / UART9_nCTS / TM3
35	I/O	PF.10 / EBI_ADR15 / SC0_nCD / I2S0_BCLK / SPI0_I2SMCLK / UART5_RXD / CCAP_DATA3 / CAN3_RXD / TAMPER4 / UART9_nRTS
36	I/O	PF.9 / EBI_ADR16 / SC0_PWR / I2S0_MCLK / SPI0_SS / UART5_nRTS / CCAP_DATA2 / CAN1_TXD / TAMPER3 / UART9_TXD
37	I/O	PF.8 / EBI_ADR17 / SC0_RST / I2S0_DI / SPI0_CLK / UART5_nCTS / CCAP_DATA1 / CAN1_RXD / TAMPER2 / UART9_RXD
38	I/O	PF.7 / EBI_ADR18 / SC0_DAT / I2S0_DO / SPI0_MISO / UART4_TXD / CCAP_DATA0 / CAN2_TXD / TAMPER1
39	I/O	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / CAN2_RXD / SPI3_I2SMCLK / TAMPER0 / EQEI2_INDEX / TRACE_SWO
40	P	V_{BAT}
41	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / EQEI2_A

Pin	Type	M467HJHAN Pin Function
42	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / EADC1_ST / I2C4_SDA / EQEI2_B
43	I/O	PH.0 / EBI_ADR7 / UART5_TXD / TM0_EXT
44	I/O	PH.1 / EBI_ADR6 / UART5_RXD / TM1_EXT
45	I/O	PH.2 / EBI_ADR5 / UART5_nRTS / UART4_TXD / I2C0_SCL / TM2_EXT
46	I/O	PH.3 / EBI_ADR4 / SPI1_I2SMCLK / UART5_nCTS / UART4_RXD / I2C0_SDA / TM3_EXT
47	I/O	PH.4 / EBI_ADR3 / SPI1_MISO / UART7_nRTS / UART6_TXD
48	I/O	PH.5 / EBI_ADR2 / SPI1_MOSI / UART7_nCTS / UART6_RXD
49	I/O	PH.6 / EBI_ADR1 / SPI1_CLK / UART7_TXD / UART9_nCTS
50	I/O	PH.7 / EBI_ADR0 / SPI1_SS / UART7_RXD / UART9_nRTS
51	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / UART9_TXD / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / ACMP2_O / EADC2_ST
52	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / UART9_RXD / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / ACMP3_O
53	P	V_{SS}
54	P	V_{DD}
55	I/O	PE.8 / EBI_ADR10 / EMAC0_RMII_MDC / I2S0_BCLK / SPI2_CLK / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / EQEI2_INDEX / TRACE_DATA3 / ECAP3_IC0
56	I/O	PE.9 / EBI_ADR11 / EMAC0_RMII_MDIO / I2S0_MCLK / SPI2_MISO / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / EQEI2_A / TRACE_DATA2 / ECAP3_IC1
57	I/O	PE.10 / EBI_ADR12 / EMAC0_RMII_TXD0 / I2S0_DI / SPI2_MOSI / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / EQEI2_B / TRACE_DATA1 / ECAP3_IC2
58	I/O	PE.11 / EBI_ADR13 / EMAC0_RMII_TXD1 / I2S0_DO / SPI2_SS / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0 / KPI_COL7
59	I/O	PE.12 / EBI_ADR14 / EMAC0_RMII_TXEN / I2S0_LRCK / SPI2_I2SMCLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK / KPI_COL6
60	I/O	PE.13 / EBI_ADR15 / EMAC0_PPS / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0 / TRACE_SWO / KPI_COL5
61	I/O	PC.8 / EBI_ADR16 / EMAC0_RMII_REFCLK / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4 / KPI_COL4
62	I/O	PC.7 / EBI_AD9 / EMAC0_RMII_RXD0 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / UART6_TXD / ACMP2_WLAT / EPWM1_CH2 / BPWM1_CH0 / CAN3_TXD / TM0 / INT3 / KPI_COL3
63	I/O	PC.6 / EBI_AD8 / EMAC0_RMII_RXD1 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / ACMP3_WLAT / EPWM1_CH3 / BPWM1_CH1 / CAN3_RXD / TM1 / INT2 / KPI_COL2
64	I/O	PA.7 / EBI_AD7 / EMAC0_RMII_CRSDV / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
65	I/O	PA.6 / EBI_AD6 / EMAC0_RMII_RXERR / SPI1_SS / SD1_nCD / SC2_CLK / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
66	I/O	PI.12 / SPIM_SS / QSPI0_MISO1 / CAN0_TXD / UART4_TXD / EPWM1_CH0 / I2C3_SMBAL
67	I/O	PI.13 / SPIM_MISO / QSPI0_MOSI1 / CAN0_RXD / UART4_RXD / EPWM1_CH1 / I2C3_SMBSUS
68	I/O	PI.14 / SPIM_D2 / QSPI0_SS / UART8_nCTS / CAN1_TXD / UART3_TXD / EPWM1_CH2 / I2C3_SCL
69	I/O	PI.15 / SPIM_D3 / QSPI0_CLK / UART8_nRTS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / I2C3_SDA
70	I/O	PJ.0 / SPIM_CLK / QSPI0_MISO0 / UART8_TXD / CAN2_TXD / EPWM1_CH4
71	I/O	PJ.1 / SPIM_MOSI / QSPI0_MOSI0 / UART8_RXD / CAN2_RXD / EPWM1_CH5
72	P	V_{SS}
73	P	V_{DD}
74	A	LDO_CAP
75	I/O	PA.5 / SPIM_D2 / QSPI0_MISO1 / SPI1_I2SMCLK / SD1_CMD / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
76	I/O	PA.4 / SPIM_D3 / QSPI0_MOSI1 / SPI0_I2SMCLK / SD1_CLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
77	I/O	PA.3 / SPIM_SS / QSPI0_SS / SPI0_SS / SD1_DAT3 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
78	I/O	PA.2 / SPIM_CLK / QSPI0_CLK / SPI0_CLK / SD1_DAT2 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / EQEI3_INDEX / PSIO0_CH5
79	I/O	PA.1 / SPIM_MISO / QSPI0_MISO0 / SPI0_MISO / SD1_DAT1 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / CCAP_DATA7 / BPWM0_CH1 / EPWM0_CH4 / EQEI3_A / DAC1_ST / PSIO0_CH6
80	I/O	PA.0 / SPIM_MOSI / QSPI0_MOSI0 / SPI0_MOSI / SD1_DAT0 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / CCAP_DATA6 / BPWM0_CH0 / EPWM0_CH5 / EQEI3_B / DAC0_ST /

Pin	Type	M467HJHAN Pin Function
		PSIO0_CH7
81	P	V_{DDIO}
82	I/O	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / SD1_nCD / UART6_TXD / PSIO0_CH0
83	I/O	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / UART6_RXD / PSIO0_CH1
84	I	nRESET
85	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / SC1_DAT / I2S0_DO / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST
86	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / SC1_CLK / I2S0_LRCK / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / EADC1_ST
87	I/O	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / UART7_TXD / CAN2_TXD / PSIO0_CH2
88	I/O	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / UART7_RXD / CAN2_RXD / PSIO0_CH3
89	I/O	PC.5 / EBI_AD5 / SPIM_D2 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / CCAP_DATA5 / QSPI1_SS / I2C3_SMBAL / HBI_nCK / PSIO0_CH0 / KPI_ROW0
90	I/O	PC.4 / EBI_AD4 / SPIM_D3 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / CCAP_DATA4 / QSPI1_CLK / I2C3_SMBSUS / HBI_CK / PSIO0_CH1 / KPI_ROW1
91	I/O	PC.3 / EBI_AD3 / SPIM_SS / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / CAN1_TXD / UART3_TXD / EPWM1_CH2 / CCAP_DATA3 / QSPI1_MISO0 / I2C3_SCL / HBI_nCS / PSIO0_CH2 / KPI_ROW2
92	I/O	PC.2 / EBI_AD2 / SPIM_CLK / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / CCAP_DATA2 / QSPI1_MOSI0 / I2C3_SDA / HBI_nRESET / PSIO0_CH3 / KPI_ROW3
93	I/O	PC.1 / EBI_AD1 / SPIM_MISO / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / CCAP_DATA1 / ACMP0_O / EADC0_ST / HBI_RWDS / KPI_ROW4
94	I/O	PC.0 / EBI_AD0 / SPIM_MOSI / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / CCAP_DATA0 / ACMP1_O / EADC1_ST / HBI_D2 / KPI_ROW5
95	P	V_{SS}
96	P	V_{DD}
97	I/O	PG.9 / EBI_AD0 / SD1_DAT3 / SPIM_D2 / QSPI1_MISO1 / CCAP_PIXCLK / I2C4_SCL / ECAP2_IC0 / BPWM0_CH5 / HBI_D4
98	I/O	PG.10 / EBI_AD1 / SD1_DAT2 / SPIM_D3 / QSPI1_MOSI1 / CCAP_SCLK / I2C4_SDA / ECAP2_IC1 / BPWM0_CH4 / HBI_D3
99	I/O	PG.11 / EBI_AD2 / SD1_DAT1 / SPIM_SS / QSPI1_SS / UART7_TXD / CCAP_SFIELD / I2C4_SMBAL / ECAP2_IC2 / BPWM0_CH3 / HBI_D0
100	I/O	PG.12 / EBI_AD3 / SD1_DAT0 / SPIM_CLK / QSPI1_CLK / UART7_RXD / CCAP_VSYNC / I2C4_SMBSUS / BPWM0_CH2 / HBI_D1
101	I/O	PG.13 / EBI_AD4 / SD1_CMD / SPIM_MISO / QSPI1_MISO0 / UART6_TXD / CCAP_HSYNC / BPWM0_CH1 / HBI_D5
102	I/O	PG.14 / EBI_AD5 / SD1_CLK / SPIM_MOSI / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0 / HBI_D6
103	I/O	PG.15 / SD1_nCD / CLKO / EADC0_ST / HBI_D7
104	I/O	PJ.2 / EBI_AD5 / UART8_nCTS / QSPI1_SS / CCAP_DATA5 / CAN0_TXD / HBI_nRESET
105	I/O	PJ.3 / EBI_AD4 / UART8_nRTS / QSPI1_CLK / CCAP_DATA4 / CAN0_RXD / HBI_D3
106	I/O	PJ.4 / EBI_AD3 / UART8_TXD / QSPI1_MISO0 / CCAP_DATA3 / CAN1_TXD / HBI_D2
107	I/O	PJ.5 / EBI_AD2 / UART8_RXD / QSPI1_MOSI0 / CCAP_DATA2 / CAN1_RXD / HBI_D1
108	I/O	PJ.6 / EBI_AD1 / UART9_nCTS / CCAP_DATA1 / CAN2_TXD / HBI_D0
109	I/O	PJ.7 / EBI_AD0 / UART9_nRTS / CCAP_DATA0 / CAN2_RXD / HBI_nCS
110	I/O	PH.12 / EBI_AD0 / UART9_TXD / QSPI1_MISO1 / CCAP_PIXCLK / CAN3_TXD / HBI_nCK
111	I/O	PH.13 / EBI_AD1 / UART9_RXD / QSPI1_MOSI1 / CCAP_SCLK / CAN3_RXD / HBI_CK
112	I/O	PH.14 / EBI_AD2 / QSPI1_SS / CCAP_SFIELD / HBI_RWDS
113	I/O	PH.15 / EBI_AD3 / QSPI1_CLK / CCAP_VSYNC / HBI_D4
114	I/O	PD.7 / EBI_AD4 / UART1_TXD / I2C0_SCL / SPI1_MISO / QSPI1_MISO0 / CCAP_HSYNC / SC1_PWR / HBI_D5 / PSIO0_CH4
115	I/O	PD.6 / EBI_AD5 / UART1_RXD / I2C0_SDA / SPI1_MOSI / QSPI1_MOSI0 / SC1_RST / ACMP0_O / EADC0_ST / HBI_D6 / PSIO0_CH5
116	I/O	PD.5 / I2C1_SCL / SPI1_CLK / SC1_DAT / ACMP1_O / EADC1_ST / HBI_D7 / PSIO0_CH6
117	I/O	PD.4 / USCI0_CTL0 / I2C1_SDA / SPI1_SS / SC1_CLK / USB_VBUS_ST / PSIO0_CH7
118	I/O	PD.3 / EBI_AD10 / USCI0_CTL1 / SPI0_SS / UART3_nRTS / SC2_PWR / SC1_nCD / UART0_TXD / I2S1_BCLK / EQEI3_A
119	I/O	PD.2 / EBI_AD11 / USCI0_DAT1 / SPI0_CLK / UART3_nCTS / SC2_RST / UART0_RXD / I2S1_MCLK / EQEI3_B
120	I/O	PD.1 / EBI_AD12 / USCI0_DAT0 / SPI0_MISO / UART3_TXD / I2C2_SCL / SC2_DAT / I2S1_DI / EQEI2_INDEX / ECAP2_IC0
121	I/O	PD.0 / EBI_AD13 / USCI0_CLK / SPI0_MOSI / UART3_RXD / I2C2_SDA / SC2_CLK / I2S1_DO / EQEI2_A / ECAP2_IC1 / TM2

Pin	Type	M467HJHAN Pin Function
122	I/O	PD.13 / EBI_AD10 / SD0_nCD / SPI0_I2SMCLK / SPI1_I2SMCLK / QSPI1_MOSI0 / SC2_nCD / SD1_CLK / UART6_RXD / I2S1_LRCK / BPWM0_CH0 / EQEI2_B / ECAP2_IC2 / CLKO / EADC0_ST
123	I/O	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / SD1_nCD / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / ECAP3_IC0 / USB_VBUS / PSIO0_CH4
124	I/O	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / ECAP3_IC1 / USB_D- / PSIO0_CH5
125	I/O	PA.14 / I2S0_DI / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / EQEI1_B / ECAP3_IC2 / USB_D+ / PSIO0_CH6
126	I/O	PA.15 / I2S0_DO / UART0_RXD / SPIM_MOSI / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / EQEI3_INDEX / USB_OTG_ID / PSIO0_CH7
127	A	HSUSB_VRES
128	P	HSUSB_VDD33
129	I/O	HSUSB_VBUS
130	A	HSUSB_D-
131	P	HSUSB_VSS
132	A	HSUSB_D+
133	A	HSUSB_VDD12_CAP
134	I	HSUSB_ID
135	I/O	PE.7 / SD0_CMD / SPIM_D2 / UART5_TXD / CAN1_TXD / UART9_nCTS / EQEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / ACMP2_O / PSIO0_CH0
136	I/O	PE.6 / SD0_CLK / SPIM_D3 / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / CAN1_RXD / UART9_nRTS / EQEI1_A / EPWM0_CH1 / BPWM0_CH4 / ACMP3_O / PSIO0_CH1
137	I/O	PE.5 / EBI_nRD / SD0_DAT3 / SPIM_SS / SPI3_SS / SC0_PWR / USCI0_CTL1 / UART6_TXD / UART7_nRTS / UART9_TXD / EQEI1_B / EPWM0_CH2 / BPWM0_CH3 / PSIO0_CH2
138	I/O	PE.4 / EBI_nWR / SD0_DAT2 / SPIM_CLK / SPI3_CLK / SC0_RST / USCI0_DAT1 / UART6_RXD / UART7_nCTS / UART9_RXD / EQEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / PSIO0_CH3
139	I/O	PE.3 / EBI_MCLK / SD0_DAT1 / SPIM_MISO / SPI3_MISO / SC0_DAT / USCI0_DAT0 / UART6_nRTS / UART7_TXD / UART8_nCTS / EQEI0_A / EPWM0_CH4 / BPWM0_CH1
140	I/O	PE.2 / EBI_ALE / SD0_DAT0 / SPIM_MOSI / SPI3_MOSI / SC0_CLK / USCI0_CLK / UART6_nCTS / UART7_RXD / UART8_nRTS / EQEI0_B / EPWM0_CH5 / BPWM0_CH0
141	P	V_{SS}
142	P	V_{DD}
143	I/O	PE.1 / EBI_AD10 / QSPI0_MISO0 / SC2_DAT / I2S0_BCLK / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS / UART8_TXD
144	I/O	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SC2_CLK / I2S0_MCLK / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS / UART8_RXD
145	I/O	PH.8 / EBI_AD12 / QSPI0_CLK / SC2_PWR / I2S0_DI / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD / UART9_nCTS
146	I/O	PH.9 / EBI_AD13 / QSPI0_SS / SC2_RST / I2S0_DO / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD / UART9_nRTS
147	I/O	PH.10 / EBI_AD14 / QSPI0_MISO1 / SC2_nCD / I2S0_LRCK / SPI1_I2SMCLK / UART4_TXD / UART0_TXD / UART9_TXD
148	I/O	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5 / UART9_RXD
149	I/O	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SC1_nCD / SPI0_I2SMCLK / I2S1_BCLK / EPWM0_CH4
150	I/O	PJ.8 / EBI_nRD / SD1_DAT3 / SPIM_SS / UART7_TXD / CAN2_TXD / BPWM0_CH5
151	I/O	PJ.9 / EBI_nWR / SD1_DAT2 / SPIM_MISO / UART7_RXD / CAN2_RXD / BPWM0_CH4
152	I/O	PJ.10 / EBI_MCLK / SD1_DAT1 / SPIM_D2 / UART6_TXD / I2C4_SCL / ECAP2_IC0 / CAN0_TXD / BPWM0_CH3
153	I/O	PJ.11 / EBI_ALE / SD1_DAT0 / SPIM_D3 / UART6_RXD / I2C4_SDA / ECAP2_IC1 / CAN0_RXD / BPWM0_CH2
154	I/O	PJ.12 / EBI_nCS0 / SD1_CMD / SPIM_CLK / I2C4_SMBAL / ECAP2_IC2 / CAN1_TXD / BPWM0_CH1 / HSUSB_VBUS_ST
155	I/O	PJ.13 / SD1_CLK / SPIM_MOSI / I2C4_SMBSUS / CAN1_RXD / BPWM0_CH0 / HSUSB_VBUS_EN
156	I/O	PG.5 / EBI_nCS1 / SPI3_SS / SC1_PWR / I2C3_SMBAL / I2S1_MCLK / EPWM0_CH3
157	I/O	PG.6 / EBI_nCS2 / SPI3_CLK / SC1_RST / I2C3_SMBSUS / I2S1_DI / EPWM0_CH2
158	I/O	PG.7 / EBI_nWRL / SPI3_MISO / SC1_DAT / I2C3_SCL / I2S1_DO / EPWM0_CH1
159	I/O	PG.8 / EBI_nWRH / SPI3_MOSI / SC1_CLK / I2C3_SDA / I2S1_LRCK / EPWM0_CH0
160	P	V_{SS}
161	A	LDO_CAP
162	P	V_{DD}
163	I/O	PC.14 / EBI_AD11 / SC1_nCD / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / ETMC_TRACE_CLK / TM1 / USB_VBUS_ST / HSUSB_VBUS_ST
164	I/O	PB.15 / EADC0_CH15 / EADC1_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / CCAP_DATA1 / EPWM0_BRAKE1 / EPWM1_CH0 / ETMC_TRACE_DATA0 / TM0_EXT / USB_VBUS_EN / HSUSB_VBUS_EN / PSIO0_CH0 / KPI_COL0

Pin	Type	M467HJHAN Pin Function
165	I/O	PB.14 / EADC0_CH14 / EADC1_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / CCAP_DATA0 / EPWM1_CH1 / ETMC_TRACE_DATA1 / TM1_EXT / CLKO / USB_VBUS_ST / PSIO0_CH1 / KPI_COL1
166	I/O	PB.13 / EADC0_CH13 / EADC1_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / CCAP_PIXCLK / EPWM1_CH2 / ETMC_TRACE_DATA2 / TM2_EXT / CAN3_TXD / PSIO0_CH2 / KPI_COL2
167	I/O	PB.12 / EADC0_CH12 / EADC1_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / CCAP_SCLK / EPWM1_CH3 / ETMC_TRACE_DATA3 / TM3_EXT / CAN3_RXD / PSIO0_CH3 / KPI_COL3
168	P	AV_{DD}
169	A	V_{REF}
170	P	AV_{SS}
171	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / EMAC0_RMII_MDC / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / CCAP_SFIELD / HSUSB_VBUS_ST
172	I/O	PB.10 / EADC0_CH10 / ACMP2_P3 / EBI_ADR17 / EMAC0_RMII_MDIO / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / CCAP_VSYNC / HSUSB_VBUS_EN
173	I/O	PB.9 / EADC0_CH9 / ACMP2_P2 / EBI_ADR18 / EMAC0_RMII_TXD0 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / CAN2_TXD / INT7 / CCAP_HSYNC
174	I/O	PB.8 / EADC0_CH8 / ACMP2_P1 / EBI_ADR19 / EMAC0_RMII_TXD1 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / CAN2_RXD / INT6 / EADC2_ST
175	I/O	PB.7 / EADC0_CH7 / EADC2_CH15 / ACMP2_P0 / EBI_nWRL / EMAC0_RMII_TXEN / CAN1_TXD / UART1_TXD / SD1_CMD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O / KPI_COL4
176	I/O	PB.6 / EADC0_CH6 / EADC2_CH14 / ACMP2_N / EBI_nWRH / EMAC0_PPS / CAN1_RXD / UART1_RXD / SD1_CLK / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O / KPI_COL5

Table 4.1-11 M467HJHAN Multi-function Pin Table

4.1.4.5 M467 Series LQFP64 EPAD Pin Multi-function Pin Diagram

Corresponding Part Number: M467S2JHAE

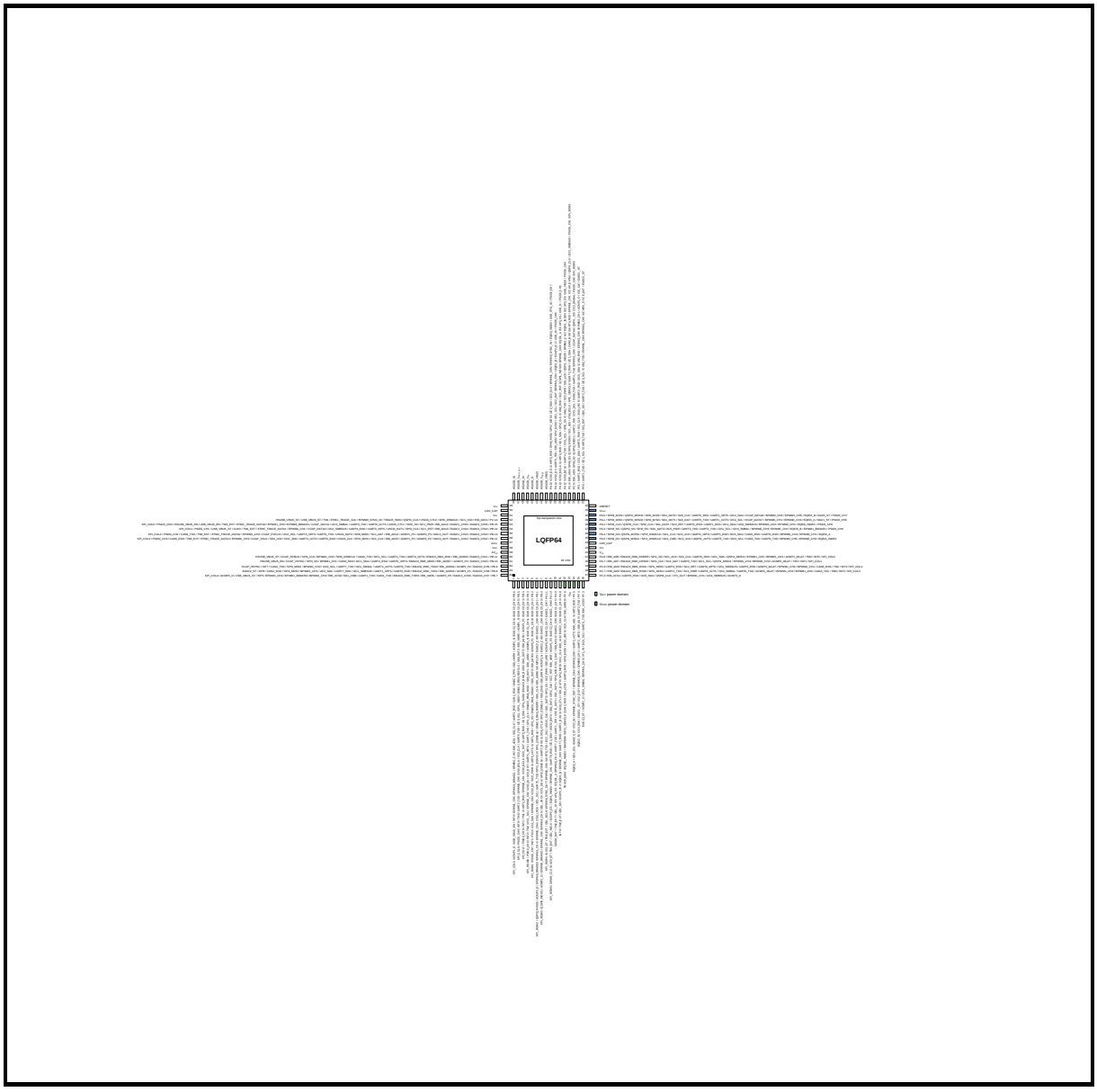


Figure 4.1-27 M467S2JHAE Multi-function Pin Diagram

Pin	Type	M467S2JHAE Pin Function
1	I/O	PB.6 / EADC0_CH6 / EADC2_CH14 / ACMP2_N / EBI_nWRH / EMAC0_PPS / CAN1_RXD / UART1_RXD / SD1_CLK / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O / KPI_COL5
2	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / EMAC0_RMII_REFCLK / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
3	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / EMAC0_RMII_RXD0 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
4	I/O	PB.3 / EADC0_CH3 / EADC1_CH11 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / EMAC0_RMII_RXD1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
5	I/O	PB.2 / EADC0_CH2 / EADC1_CH10 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / EMAC0_RMII_CRSDV

Pin	Type	M467S2JHAE Pin Function
		/ SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
6	I/O	PB.1 / EADC0_CH1 / EADC1_CH9 / EADC2_CH9 / ACMP3_P0 / EBI_ADR8 / SD0_CLK / EMAC0_RMII_RXERR / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / ACMP2_O / QSPI0_MISO1 / KPI_ROW2
7	I/O	PB.0 / EADC0_CH0 / EADC1_CH8 / EADC2_CH8 / ACMP3_N / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / I2S1_LRCK / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / ACMP3_O / QSPI0_MOSI1 / KPI_ROW3
8	I/O	PA.11 / EADC1_CH7 / EADC2_CH7 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / SD1_DAT3 / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / I2S1_BCLK / TM0_EXT / DAC1_ST / KPI_ROW4
9	I/O	PA.10 / EADC1_CH6 / EADC2_CH6 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / SD1_DAT2 / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / I2S1_MCLK / TM1_EXT / DAC0_ST / SWDH_CLK / KPI_ROW5
10	I/O	PA.9 / EADC1_CH5 / EADC2_CH5 / EBI_MCLK / SC2_DAT / SPI2_MISO / SD1_DAT1 / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / I2S1_DI / TM2_EXT / SWDH_DAT
11	I/O	PA.8 / EADC1_CH4 / EADC2_CH4 / EBI_ALE / SC2_CLK / SPI2_MOSI / SD1_DAT0 / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / I2S1_DO / TM3_EXT / INT4
12	I/O	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / CAN2_RXD / SPI3_I2SMCLK / TAMPER0 / EQEI2_INDEX / TRACE_SWO
13	P	V_{BAT}
14	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / EQEI2_A
15	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / EADC1_ST / I2C4_SDA / EQEI2_B
16	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / ACMP2_O / EADC2_ST
17	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / ACMP3_O
18	I/O	PC.7 / EBI_AD9 / EMAC0_RMII_RXD0 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / UART6_TXD / ACMP2_WLAT / EPWM1_CH2 / BPWM1_CH0 / CAN3_TXD / TM0 / INT3 / KPI_COL3
19	I/O	PC.6 / EBI_AD8 / EMAC0_RMII_RXD1 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / ACMP3_WLAT / EPWM1_CH3 / BPWM1_CH1 / CAN3_RXD / TM1 / INT2 / KPI_COL2
20	I/O	PA.7 / EBI_AD7 / EMAC0_RMII_CRSDV / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
21	I/O	PA.6 / EBI_AD6 / EMAC0_RMII_RXERR / SPI1_SS / SD1_nCD / SC2_CLK / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
22	P	V_{SS}
23	P	V_{DD}
24	A	LDO_CAP
25	I/O	PA.5 / SPIM_D2 / QSPI0_MISO1 / SPI1_I2SMCLK / SD1_CMD / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
26	I/O	PA.4 / SPIM_D3 / QSPI0_MOSI1 / SPI0_I2SMCLK / SD1_CLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
27	I/O	PA.3 / SPIM_SS / QSPI0_SS / SPI0_SS / SD1_DAT3 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
28	I/O	PA.2 / SPIM_CLK / QSPI0_CLK / SPI0_CLK / SD1_DAT2 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / EQEI3_INDEX / PSIO0_CH5
29	I/O	PA.1 / SPIM_MISO / QSPI0_MISO0 / SPI0_MISO / SD1_DAT1 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / CCAP_DATA7 / BPWM0_CH1 / EPWM0_CH4 / EQEI3_A / DAC1_ST / PSIO0_CH6
30	I/O	PA.0 / SPIM_MOSI / QSPI0_MOSI0 / SPI0_MOSI / SD1_DAT0 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / CCAP_DATA6 / BPWM0_CH0 / EPWM0_CH5 / EQEI3_B / DAC0_ST / PSIO0_CH7
31	P	V_{DDIO}
32	I	nRESET
33	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / SC1_DAT / I2S0_DO / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST
34	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / SC1_CLK / I2S0_LRCK / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / EADC1_ST
35	I/O	PC.5 / EBI_AD5 / SPIM_D2 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD

Pin	Type	M467S2JHAE Pin Function
		/ EPWM1_CH0 / CCAP_DATA5 / QSPI1_SS / I2C3_SMBAL / PSIO0_CH0 / KPI_ROW0
36	I/O	PC.4 / EBI_AD4 / SPIM_D3 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / CCAP_DATA4 / QSPI1_CLK / I2C3_SMBSUS / PSIO0_CH1 / KPI_ROW1
37	I/O	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / SD1_nCD / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / ECAP3_IC0 / USB_VBUS / PSIO0_CH4
38	I/O	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / ECAP3_IC1 / USB_D- / PSIO0_CH5
39	I/O	PA.14 / I2S0_DI / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / EQEI1_B / ECAP3_IC2 / USB_D+ / PSIO0_CH6
40	I/O	PA.15 / I2S0_DO / UART0_RXD / SPIM_MOSI / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / EQEI3_INDEX / USB_OTG_ID / PSIO0_CH7
41	A	HSUSB_VRES
42	P	HSUSB_VDD33
43	I/O	HSUSB_VBUS
44	A	HSUSB_D-
45	P	HSUSB_VSS
46	A	HSUSB_D+
47	A	HSUSB_VDD12_CAP
48	I	HSUSB_ID
49	P	V_{SS}
50	A	LDO_CAP
51	P	V_{DD}
52	I/O	PC.14 / EBI_AD11 / SC1_nCD / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / ETMC_TRACE_CLK / TM1 / USB_VBUS_ST / HSUSB_VBUS_ST
53	I/O	PB.15 / EADC0_CH15 / EADC1_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / CCAP_DATA1 / EPWM0_BRAKE1 / EPWM1_CH0 / ETMC_TRACE_DATA0 / TM0_EXT / USB_VBUS_EN / HSUSB_VBUS_EN / PSIO0_CH0 / KPI_COL0
54	I/O	PB.14 / EADC0_CH14 / EADC1_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / CCAP_DATA0 / EPWM1_CH1 / ETMC_TRACE_DATA1 / TM1_EXT / CLKO / USB_VBUS_ST / PSIO0_CH1 / KPI_COL1
55	I/O	PB.13 / EADC0_CH13 / EADC1_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / CCAP_PIXCLK / EPWM1_CH2 / ETMC_TRACE_DATA2 / TM2_EXT / CAN3_TXD / PSIO0_CH2 / KPI_COL2
56	I/O	PB.12 / EADC0_CH12 / EADC1_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / CCAP_SCLK / EPWM1_CH3 / ETMC_TRACE_DATA3 / TM3_EXT / CAN3_RXD / PSIO0_CH3 / KPI_COL3
57	P	AV_{DD}
58	A	V_{REF}
59	P	AV_{SS}
60	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / EMAC0_RMII_MDC / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / CCAP_SFIELD / HSUSB_VBUS_ST
61	I/O	PB.10 / EADC0_CH10 / ACMP2_P3 / EBI_ADR17 / EMAC0_RMII_MDIO / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / CCAP_VSYNC / HSUSB_VBUS_EN
62	I/O	PB.9 / EADC0_CH9 / ACMP2_P2 / EBI_ADR18 / EMAC0_RMII_TXD0 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / CAN2_TXD / INT7 / CCAP_HSYNC
63	I/O	PB.8 / EADC0_CH8 / ACMP2_P1 / EBI_ADR19 / EMAC0_RMII_TXD1 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / CAN2_RXD / INT6 / EADC2_ST
64	I/O	PB.7 / EADC0_CH7 / EADC2_CH15 / ACMP2_P0 / EBI_nWRL / EMAC0_RMII_TXEN / CAN1_TXD / UART1_TXD / SD1_CMD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O / KPI_COL4
65	P	V_{SS}

Table 4.1-12 M467S2JHAE Multi-function Pin Table

4.1.4.6 M467 Series LQFP128 EPAD Pin Multi-function Pin Diagram

Corresponding Part Number: M467K2JHAE

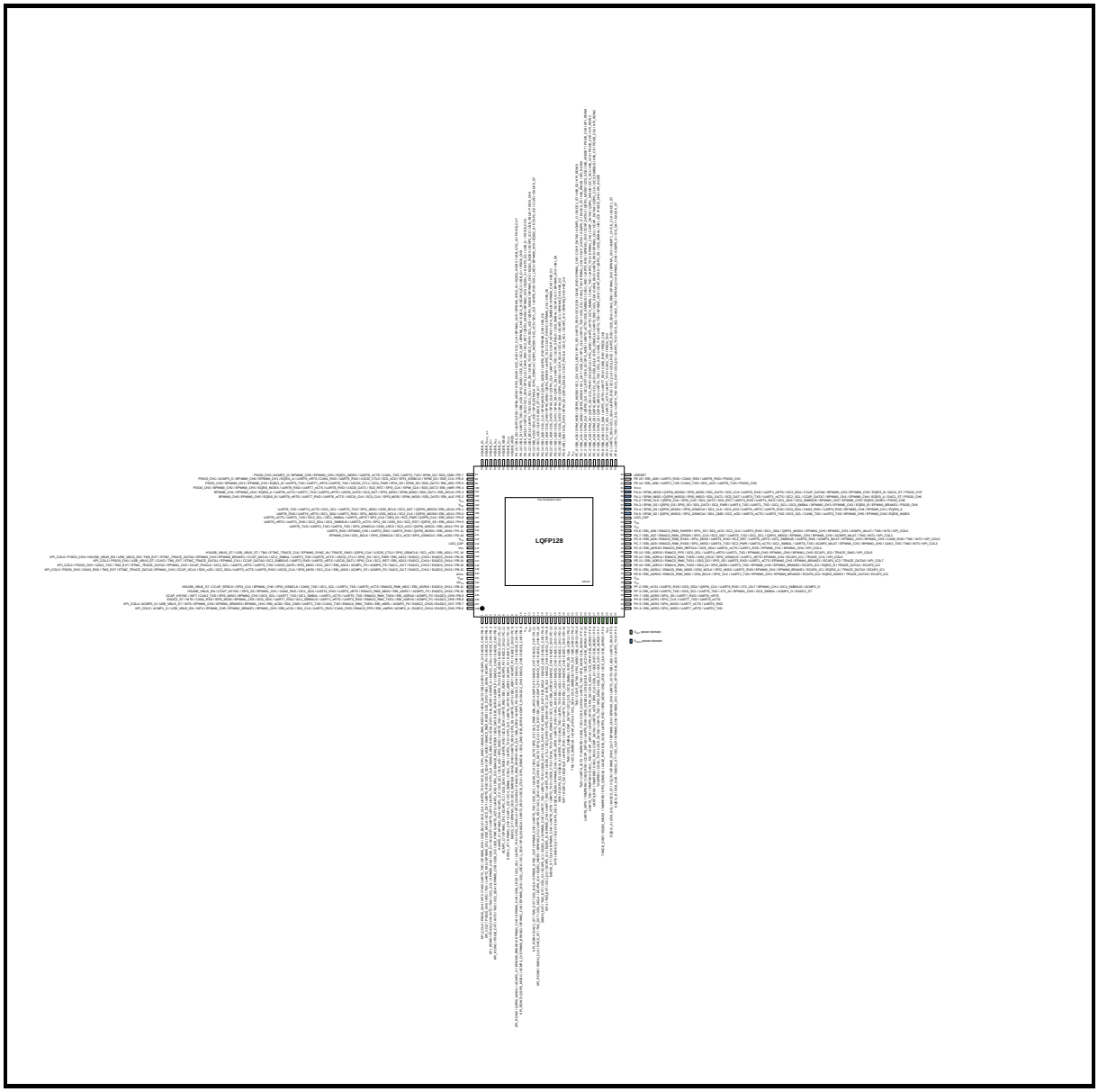


Figure 4.1-28 M467K2JHAE Multi-function Pin Diagram

Pin	Type	M467K2JHAE Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / EMAC0_RMII_REFCLK / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / EMAC0_RMII_RXD0 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / EADC1_CH11 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / EMAC0_RMII_RXD1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / EADC1_CH10 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / EMAC0_RMII_CRSDV / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PC.12 / EADC2_CH13 / EBI_ADR4 / UART0_TXD / I2C0_SCL / UART6_TXD / SPI3_MISO /

Pin	Type	M467K2JHAE Pin Function
		SC0_nCD / I2C4_SCL / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	I/O	PC.11 / EADC2_CH12 / ACMP3_P3 / EBI_ADR5 / UART0_RXD / I2C0_SDA / UART6_RXD / SPI3_MOSI / I2C4_SDA / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	I/O	PC.10 / EADC2_CH11 / ACMP3_P2 / EBI_ADR6 / UART6_nRTS / SPI3_CLK / UART3_TXD / CAN1_TXD / I2C4_SMBAL / ECAP1_IC0 / EPWM1_CH2 / EADC1_ST
8	I/O	PC.9 / EADC2_CH10 / ACMP3_P1 / EBI_ADR7 / UART6_nCTS / SPI3_SS / UART3_RXD / CAN1_RXD / I2C4_SMBSUS / EPWM1_CH3 / EADC1_ST
9	I/O	PB.1 / EADC0_CH1 / EADC1_CH9 / EADC2_CH9 / ACMP3_P0 / EBI_ADR8 / SD0_CLK / EMAC0_RMII_RXERR / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / ACMP2_O / QSPI0_MISO1 / KPI_ROW2
10	I/O	PB.0 / EADC0_CH0 / EADC1_CH8 / EADC2_CH8 / ACMP3_N / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / I2S1_LRCK / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / ACMP3_O / QSPI0_MOSI1 / KPI_ROW3
11	P	V_{SS}
12	P	V_{DD}
13	I/O	PA.11 / EADC1_CH7 / EADC2_CH7 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / SD1_DAT3 / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / I2S1_BCLK / TM0_EXT / DAC1_ST / KPI_ROW4
14	I/O	PA.10 / EADC1_CH6 / EADC2_CH6 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / SD1_DAT2 / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / I2S1_MCLK / TM1_EXT / DAC0_ST / SWDH_CLK / KPI_ROW5
15	I/O	PA.9 / EADC1_CH5 / EADC2_CH5 / EBI_MCLK / SC2_DAT / SPI2_MISO / SD1_DAT1 / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / I2S1_DI / TM2_EXT / SWDH_DAT
16	I/O	PA.8 / EADC1_CH4 / EADC2_CH4 / EBI_ALE / SC2_CLK / SPI2_MOSI / SD1_DAT0 / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / I2S1_DO / TM3_EXT / INT4
17	I/O	PC.13 / EADC1_CH3 / EADC2_CH3 / EBI_ADR10 / SC2_nCD / SPI2_I2SMCLK / CAN1_TXD / USCI0_CTL0 / UART2_TXD / UART8_nCTS / BPWM0_CH4 / CLKO / EADC0_ST
18	I/O	PD.12 / EADC1_CH2 / EADC2_CH2 / EBI_nCS0 / CAN1_RXD / UART2_RXD / UART8_nRTS / BPWM0_CH5 / EQEI0_INDEX / ECAP3_IC0 / CLKO / EADC0_ST / INT5
19	I/O	PD.11 / EADC1_CH1 / EADC2_CH1 / EBI_nCS1 / UART1_TXD / CAN0_TXD / UART8_TXD / EQEI0_A / ECAP3_IC1 / INT6
20	I/O	PD.10 / EADC1_CH0 / EADC2_CH0 / EBI_nCS2 / UART1_RXD / CAN0_RXD / UART8_RXD / EQEI0_B / ECAP3_IC2 / INT7
21	I/O	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / CCAP_DATA7 / I2C3_SMBAL / TM0
22	I/O	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / CCAP_DATA6 / I2C3_SMBSUS / TM1
23	I/O	PG.4 / EBI_ADR13 / SPI2_MISO / CCAP_DATA5 / TM2
24	I/O	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / CCAP_DATA4 / CAN3_TXD / TAMPER5 / UART9_nCTS / TM3
25	I/O	PF.10 / EBI_ADR15 / SC0_nCD / I2S0_BCLK / SPI0_I2SMCLK / UART5_RXD / CCAP_DATA3 / CAN3_RXD / TAMPER4 / UART9_nRTS
26	I/O	PF.9 / EBI_ADR16 / SC0_PWR / I2S0_MCLK / SPI0_SS / UART5_nRTS / CCAP_DATA2 / CAN1_TXD / TAMPER3 / UART9_TXD
27	I/O	PF.8 / EBI_ADR17 / SC0_RST / I2S0_DI / SPI0_CLK / UART5_nCTS / CCAP_DATA1 / CAN1_RXD / TAMPER2 / UART9_RXD
28	I/O	PF.7 / EBI_ADR18 / SC0_DAT / I2S0_DO / SPI0_MISO / UART4_TXD / CCAP_DATA0 / CAN2_TXD / TAMPER1
29	I/O	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / CAN2_RXD / SPI3_I2SMCLK / TAMPER0 / EQEI2_INDEX / TRACE_SWO
30	P	V_{BAT}
31	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / EQEI2_A
32	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / EADC1_ST / I2C4_SDA / EQEI2_B
33	I/O	PH.4 / EBI_ADR3 / SPI1_MISO / UART7_nRTS / UART6_TXD
34	I/O	PH.5 / EBI_ADR2 / SPI1_MOSI / UART7_nCTS / UART6_RXD
35	I/O	PH.6 / EBI_ADR1 / SPI1_CLK / UART7_TXD / UART9_nCTS
36	I/O	PH.7 / EBI_ADR0 / SPI1_SS / UART7_RXD / UART9_nRTS
37	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / UART9_TXD / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / ACMP2_O / EADC2_ST
38	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / UART9_RXD / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / ACMP3_O
39	P	V_{SS}
40	P	V_{DD}
41	I/O	PE.8 / EBI_ADR10 / EMAC0_RMII_MDC / I2S0_BCLK / SPI2_CLK / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / EQEI2_INDEX / TRACE_DATA3 / ECAP3_IC0

Pin	Type	M467K2JHAE Pin Function
42	I/O	PE.9 / EBI_ADR11 / EMAC0_RMII_MDIO / I2S0_MCLK / SPI2_MISO / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / EQEI2_A / TRACE_DATA2 / ECAP3_IC1
43	I/O	PE.10 / EBI_ADR12 / EMAC0_RMII_TXD0 / I2S0_DI / SPI2_MOSI / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / EQEI2_B / TRACE_DATA1 / ECAP3_IC2
44	I/O	PE.11 / EBI_ADR13 / EMAC0_RMII_TXD1 / I2S0_DO / SPI2_SS / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0 / KPI_COL7
45	I/O	PE.12 / EBI_ADR14 / EMAC0_RMII_TXEN / I2S0_LRCK / SPI2_I2SMCLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK / KPI_COL6
46	I/O	PE.13 / EBI_ADR15 / EMAC0_PPS / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0 / TRACE_SWO / KPI_COL5
47	I/O	PC.8 / EBI_ADR16 / EMAC0_RMII_REFCLK / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4 / KPI_COL4
48	I/O	PC.7 / EBI_AD9 / EMAC0_RMII_RXD0 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / UART6_TXD / ACMP2_WLAT / EPWM1_CH2 / BPWM1_CH0 / CAN3_TXD / TM0 / INT3 / KPI_COL3
49	I/O	PC.6 / EBI_AD8 / EMAC0_RMII_RXD1 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / ACMP3_WLAT / EPWM1_CH3 / BPWM1_CH1 / CAN3_RXD / TM1 / INT2 / KPI_COL2
50	I/O	PA.7 / EBI_AD7 / EMAC0_RMII_CRSDV / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
51	I/O	PA.6 / EBI_AD6 / EMAC0_RMII_RXERR / SPI1_SS / SD1_nCD / SC2_CLK / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
52	P	V_{SS}
53	P	V_{DD}
54	A	LDO_CAP
55	I/O	PA.5 / SPIM_D2 / QSPI0_MISO1 / SPI1_I2SMCLK / SD1_CMD / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
56	I/O	PA.4 / SPIM_D3 / QSPI0_MOSI1 / SPI0_I2SMCLK / SD1_CLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
57	I/O	PA.3 / SPIM_SS / QSPI0_SS / SPI0_SS / SD1_DAT3 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
58	I/O	PA.2 / SPIM_CLK / QSPI0_CLK / SPI0_CLK / SD1_DAT2 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / EQEI3_INDEX / PSIO0_CH5
59	I/O	PA.1 / SPIM_MISO / QSPI0_MISO0 / SPI0_MISO / SD1_DAT1 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / CCAP_DATA7 / BPWM0_CH1 / EPWM0_CH4 / EQEI3_A / DAC1_ST / PSIO0_CH6
60	I/O	PA.0 / SPIM_MOSI / QSPI0_MOSI0 / SPI0_MOSI / SD1_DAT0 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / CCAP_DATA6 / BPWM0_CH0 / EPWM0_CH5 / EQEI3_B / DAC0_ST / PSIO0_CH7
61	P	V_{DDIO}
62	I/O	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / SD1_nCD / UART6_TXD / PSIO0_CH0
63	I/O	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / UART6_RXD / PSIO0_CH1
64	I	nRESET
65	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / SC1_DAT / I2S0_DO / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST
66	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / SC1_CLK / I2S0_LRCK / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / EADC1_ST
67	I/O	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / UART7_TXD / CAN2_TXD / PSIO0_CH2
68	I/O	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / UART7_RXD / CAN2_RXD / PSIO0_CH3
69	I/O	PC.5 / EBI_AD5 / SPIM_D2 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / CCAP_DATA5 / QSPI1_SS / I2C3_SMBAL / HBI_nCK / PSIO0_CH0 / KPI_ROW0
70	I/O	PC.4 / EBI_AD4 / SPIM_D3 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / CCAP_DATA4 / QSPI1_CLK / I2C3_SMBSUS / HBI_CK / PSIO0_CH1 / KPI_ROW1
71	I/O	PC.3 / EBI_AD3 / SPIM_SS / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / CAN1_TXD / UART3_TXD / EPWM1_CH2 / CCAP_DATA3 / QSPI1_MISO0 / I2C3_SCL / HBI_nCS / PSIO0_CH2 / KPI_ROW2
72	I/O	PC.2 / EBI_AD2 / SPIM_CLK / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / CCAP_DATA2 / QSPI1_MOSI0 / I2C3_SDA / HBI_nRESET / PSIO0_CH3 / KPI_ROW3
73	I/O	PC.1 / EBI_AD1 / SPIM_MISO / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / CCAP_DATA1 / ACMP0_O / EADC0_ST / HBI_RWDS / KPI_ROW4
74	I/O	PC.0 / EBI_AD0 / SPIM_MOSI / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / CCAP_DATA0 / ACMP1_O / EADC1_ST / HBI_D2 /

Pin	Type	M467K2JHAE Pin Function
		KPI_ROW5
75	P	V_{SS}
76	P	V_{DD}
77	I/O	PG.9 / EBI_AD0 / SD1_DAT3 / SPIM_D2 / QSPI1_MISO1 / CCAP_PIXCLK / I2C4_SCL / ECAP2_IC0 / BPWM0_CH5 / HBI_D4
78	I/O	PG.10 / EBI_AD1 / SD1_DAT2 / SPIM_D3 / QSPI1_MOSI1 / CCAP_SCLK / I2C4_SDA / ECAP2_IC1 / BPWM0_CH4 / HBI_D3
79	I/O	PG.11 / EBI_AD2 / SD1_DAT1 / SPIM_SS / QSPI1_SS / UART7_TXD / CCAP_SFIELD / I2C4_SMBAL / ECAP2_IC2 / BPWM0_CH3 / HBI_D0
80	I/O	PG.12 / EBI_AD3 / SD1_DAT0 / SPIM_CLK / QSPI1_CLK / UART7_RXD / CCAP_VSYNC / I2C4_SMBSUS / BPWM0_CH2 / HBI_D1
81	I/O	PG.13 / EBI_AD4 / SD1_CMD / SPIM_MISO / QSPI1_MISO0 / UART6_TXD / CCAP_HSYNC / BPWM0_CH1 / HBI_D5
82	I/O	PG.14 / EBI_AD5 / SD1_CLK / SPIM_MOSI / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0 / HBI_D6
83	I/O	PG.15 / SD1_nCD / CLKO / EADC0_ST / HBI_D7
84	I/O	PD.13 / EBI_AD10 / SD0_nCD / SPI0_I2SMCLK / SPI1_I2SMCLK / QSPI1_MOSI0 / SC2_nCD / SD1_CLK / UART6_RXD / I2S1_LRCK / BPWM0_CH0 / EQEI2_B / ECAP2_IC2 / CLKO / EADC0_ST
85	I/O	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / SD1_nCD / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / ECAP3_IC0 / USB_VBUS / PSIO0_CH4
86	I/O	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / ECAP3_IC1 / USB_D- / PSIO0_CH5
87	I/O	PA.14 / I2S0_DI / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / EQEI1_B / ECAP3_IC2 / USB_D+ / PSIO0_CH6
88	I/O	PA.15 / I2S0_DO / UART0_RXD / SPIM_MOSI / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / EQEI3_INDEX / USB_OTG_ID / PSIO0_CH7
89	A	HSUSB_VRES
90	P	HSUSB_VDD33
91	I/O	HSUSB_VBUS
92	A	HSUSB_D-
93	P	HSUSB_VSS
94	A	HSUSB_D+
95	A	HSUSB_VDD12_CAP
96	I	HSUSB_ID
97	I/O	PE.7 / SD0_CMD / SPIM_D2 / UART5_TXD / CAN1_TXD / UART9_nCTS / EQEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / ACMP2_O / PSIO0_CH0
98	I/O	PE.6 / SD0_CLK / SPIM_D3 / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / CAN1_RXD / UART9_nRTS / EQEI1_A / EPWM0_CH1 / BPWM0_CH4 / ACMP3_O / PSIO0_CH1
99	I/O	PE.5 / EBI_nRD / SD0_DAT3 / SPIM_SS / SPI3_SS / SC0_PWR / USCI0_CTL1 / UART6_TXD / UART7_nRTS / UART9_TXD / EQEI1_B / EPWM0_CH2 / BPWM0_CH3 / PSIO0_CH2
100	I/O	PE.4 / EBI_nWR / SD0_DAT2 / SPIM_CLK / SPI3_CLK / SC0_RST / USCI0_DAT1 / UART6_RXD / UART7_nCTS / UART9_RXD / EQEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / PSIO0_CH3
101	I/O	PE.3 / EBI_MCLK / SD0_DAT1 / SPIM_MISO / SPI3_MISO / SC0_DAT / USCI0_DAT0 / UART6_nRTS / UART7_TXD / UART8_nCTS / EQEI0_A / EPWM0_CH4 / BPWM0_CH1
102	I/O	PE.2 / EBI_ALE / SD0_DAT0 / SPIM_MOSI / SPI3_MOSI / SC0_CLK / USCI0_CLK / UART6_nCTS / UART7_RXD / UART8_nRTS / EQEI0_B / EPWM0_CH5 / BPWM0_CH0
103	P	V_{SS}
104	P	V_{DD}
105	I/O	PE.1 / EBI_AD10 / QSPI0_MISO0 / SC2_DAT / I2S0_BCLK / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS / UART8_TXD
106	I/O	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SC2_CLK / I2S0_MCLK / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS / UART8_RXD
107	I/O	PH.8 / EBI_AD12 / QSPI0_CLK / SC2_PWR / I2S0_DI / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD / UART9_nCTS
108	I/O	PH.9 / EBI_AD13 / QSPI0_SS / SC2_RST / I2S0_DO / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD / UART9_nRTS
109	I/O	PH.10 / EBI_AD14 / QSPI0_MISO1 / SC2_nCD / I2S0_LRCK / SPI1_I2SMCLK / UART4_TXD / UART0_TXD / UART9_TXD
110	I/O	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5 / UART9_RXD
111	I/O	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SC1_nCD / SPI0_I2SMCLK / I2S1_BCLK / EPWM0_CH4
112	P	V_{SS}
113	A	LDO_CAP
114	P	V_{DD}
115	I/O	PC.14 / EBI_AD11 / SC1_nCD / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / ETMC_TRACE_CLK / TM1 / USB_VBUS_ST / HSUSB_VBUS_ST
116	I/O	PB.15 / EADC0_CH15 / EADC1_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 /

Pin	Type	M467K2JHAE Pin Function
		UART0_nCTS / UART3_TXD / I2C2_SMBAL / CCAP_DATA1 / EPWM0_BRAKE1 / EPWM1_CH0 / ETMC_TRACE_DATA0 / TM0_EXT / USB_VBUS_EN / HSUSB_VBUS_EN / PSIO0_CH0 / KPI_COL0
117	I/O	PB.14 / EADC0_CH14 / EADC1_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / CCAP_DATA0 / EPWM1_CH1 / ETMC_TRACE_DATA1 / TM1_EXT / CLKO / USB_VBUS_ST / PSIO0_CH1 / KPI_COL1
118	I/O	PB.13 / EADC0_CH13 / EADC1_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / CCAP_PIXCLK / EPWM1_CH2 / ETMC_TRACE_DATA2 / TM2_EXT / CAN3_TXD / PSIO0_CH2 / KPI_COL2
119	I/O	PB.12 / EADC0_CH12 / EADC1_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / CCAP_SCLK / EPWM1_CH3 / ETMC_TRACE_DATA3 / TM3_EXT / CAN3_RXD / PSIO0_CH3 / KPI_COL3
120	P	AV_{DD}
121	A	V_{REF}
122	P	AV_{SS}
123	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / EMAC0_RMII_MDC / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / CCAP_SFIELD / HSUSB_VBUS_ST
124	I/O	PB.10 / EADC0_CH10 / ACMP2_P3 / EBI_ADR17 / EMAC0_RMII_MDIO / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / CCAP_VSYNC / HSUSB_VBUS_EN
125	I/O	PB.9 / EADC0_CH9 / ACMP2_P2 / EBI_ADR18 / EMAC0_RMII_TXD0 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / CAN2_TXD / INT7 / CCAP_HSYNC
126	I/O	PB.8 / EADC0_CH8 / ACMP2_P1 / EBI_ADR19 / EMAC0_RMII_TXD1 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / CAN2_RXD / INT6 / EADC2_ST
127	I/O	PB.7 / EADC0_CH7 / EADC2_CH15 / ACMP2_P0 / EBI_nWRL / EMAC0_RMII_TXEN / CAN1_TXD / UART1_TXD / SD1_CMD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O / KPI_COL4
128	I/O	PB.6 / EADC0_CH6 / EADC2_CH14 / ACMP2_N / EBI_nWRH / EMAC0_PPS / CAN1_RXD / UART1_RXD / SD1_CLK / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O / KPI_COL5
129	-	NC

Table 4.1-13 M467K2JHAE Multi-function Pin Table

4.1.4.7 M467 Series LQFP144 EPAD Pin Multi-function Pin Diagram

Corresponding Part Number: M467J2JHAE

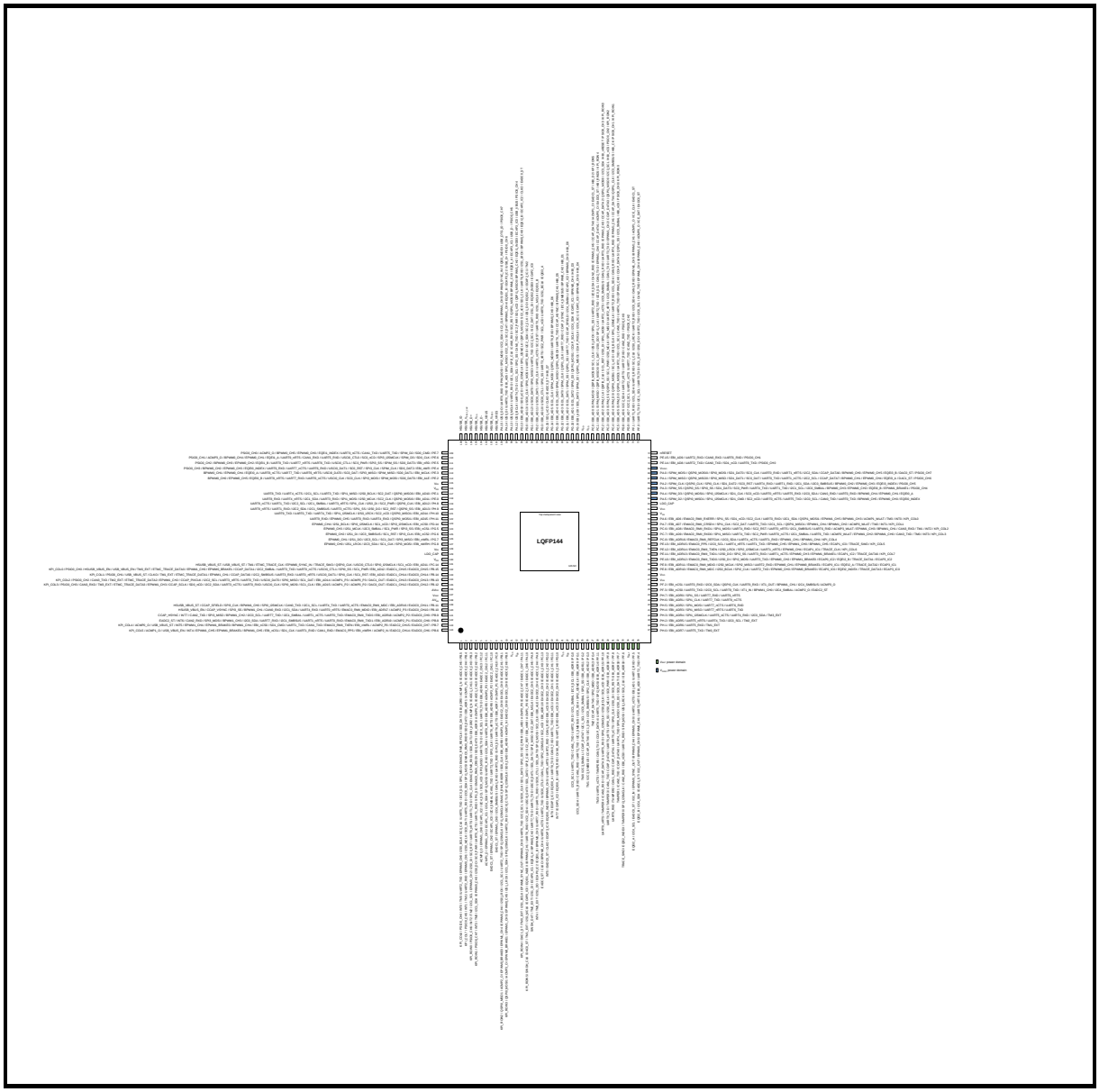


Figure 4.1-29 M467J2JHAE Multi-function Pin Diagram

Pin	Type	M467J2JHAE Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / EMAC0_RMII_REFCLK / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / EMAC0_RMII_RXD0 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / EADC1_CH11 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / EMAC0_RMII_RXD1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / EADC1_CH10 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / EMAC0_RMII_CRSDV / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PC.12 / EADC2_CH13 / EBI_ADR4 / UART0_TXD / I2C0_SCL / UART6_TXD / SPI3_MISO /

Pin	Type	M467J2JHAE Pin Function
		SC0_nCD / I2C4_SCL / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	I/O	PC.11 / EADC2_CH12 / ACMP3_P3 / EBI_ADR5 / UART0_RXD / I2C0_SDA / UART6_RXD / SPI3_MOSI / I2C4_SDA / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	I/O	PC.10 / EADC2_CH11 / ACMP3_P2 / EBI_ADR6 / UART6_nRTS / SPI3_CLK / UART3_TXD / CAN1_TXD / I2C4_SMBAL / ECAP1_IC0 / EPWM1_CH2 / EADC1_ST
8	I/O	PC.9 / EADC2_CH10 / ACMP3_P1 / EBI_ADR7 / UART6_nCTS / SPI3_SS / UART3_RXD / CAN1_RXD / I2C4_SMBSUS / EPWM1_CH3 / EADC1_ST
9	I/O	PB.1 / EADC0_CH1 / EADC1_CH9 / EADC2_CH9 / ACMP3_P0 / EBI_ADR8 / SD0_CLK / EMAC0_RMII_RXERR / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / ACMP2_O / QSPI0_MISO1 / KPI_ROW2
10	I/O	PB.0 / EADC0_CH0 / EADC1_CH8 / EADC2_CH8 / ACMP3_N / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / I2S1_LRCK / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / ACMP3_O / QSPI0_MOSI1 / KPI_ROW3
11	P	V_{SS}
12	P	V_{DD}
13	I/O	PA.11 / EADC1_CH7 / EADC2_CH7 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / SD1_DAT3 / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / I2S1_BCLK / TM0_EXT / DAC1_ST / KPI_ROW4
14	I/O	PA.10 / EADC1_CH6 / EADC2_CH6 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / SD1_DAT2 / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / I2S1_MCLK / TM1_EXT / DAC0_ST / SWDH_CLK / KPI_ROW5
15	I/O	PA.9 / EADC1_CH5 / EADC2_CH5 / EBI_MCLK / SC2_DAT / SPI2_MISO / SD1_DAT1 / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / I2S1_DI / TM2_EXT / SWDH_DAT
16	I/O	PA.8 / EADC1_CH4 / EADC2_CH4 / EBI_ALE / SC2_CLK / SPI2_MOSI / SD1_DAT0 / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / I2S1_DO / TM3_EXT / INT4
17	I/O	PC.13 / EADC1_CH3 / EADC2_CH3 / EBI_ADR10 / SC2_nCD / SPI2_I2SMCLK / CAN1_TXD / USCI0_CTL0 / UART2_TXD / UART8_nCTS / BPWM0_CH4 / CLKO / EADC0_ST
18	I/O	PD.12 / EADC1_CH2 / EADC2_CH2 / EBI_nCS0 / CAN1_RXD / UART2_RXD / UART8_nRTS / BPWM0_CH5 / EQEI0_INDEX / ECAP3_IC0 / CLKO / EADC0_ST / INT5
19	I/O	PD.11 / EADC1_CH1 / EADC2_CH1 / EBI_nCS1 / UART1_TXD / CAN0_TXD / UART8_TXD / EQEI0_A / ECAP3_IC1 / INT6
20	I/O	PD.10 / EADC1_CH0 / EADC2_CH0 / EBI_nCS2 / UART1_RXD / CAN0_RXD / UART8_RXD / EQEI0_B / ECAP3_IC2 / INT7
21	P	V_{SS}
22	P	V_{DD}
23	I/O	PG.0 / EBI_ADR8 / I2C0_SCL / I2C1_SMBAL / UART2_RXD / CAN1_TXD / UART1_TXD / I2C3_SCL
24	I/O	PG.1 / EBI_ADR9 / SPI2_I2SMCLK / I2C0_SDA / I2C1_SMBSUS / UART2_TXD / CAN1_RXD / UART1_RXD / I2C3_SDA
25	I/O	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / CCAP_DATA7 / I2C3_SMBAL / TM0
26	I/O	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / CCAP_DATA6 / I2C3_SMBSUS / TM1
27	I/O	PG.4 / EBI_ADR13 / SPI2_MISO / CCAP_DATA5 / TM2
28	I/O	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / CCAP_DATA4 / CAN3_TXD / TAMPER5 / UART9_nCTS / TM3
29	I/O	PF.10 / EBI_ADR15 / SC0_nCD / I2S0_BCLK / SPI0_I2SMCLK / UART5_RXD / CCAP_DATA3 / CAN3_RXD / TAMPER4 / UART9_nRTS
30	I/O	PF.9 / EBI_ADR16 / SC0_PWR / I2S0_MCLK / SPI0_SS / UART5_nRTS / CCAP_DATA2 / CAN1_TXD / TAMPER3 / UART9_TXD
31	I/O	PF.8 / EBI_ADR17 / SC0_RST / I2S0_DI / SPI0_CLK / UART5_nCTS / CCAP_DATA1 / CAN1_RXD / TAMPER2 / UART9_RXD
32	I/O	PF.7 / EBI_ADR18 / SC0_DAT / I2S0_DO / SPI0_MISO / UART4_TXD / CCAP_DATA0 / CAN2_TXD / TAMPER1
33	I/O	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / CAN2_RXD / SPI3_I2SMCLK / TAMPER0 / EQEI2_INDEX / TRACE_SWO
34	P	V_{BAT}
35	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / EQEI2_A
36	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / EADC1_ST / I2C4_SDA / EQEI2_B
37	I/O	PH.0 / EBI_ADR7 / UART5_TXD / TM0_EXT
38	I/O	PH.1 / EBI_ADR6 / UART5_RXD / TM1_EXT
39	I/O	PH.2 / EBI_ADR5 / UART5_nRTS / UART4_TXD / I2C0_SCL / TM2_EXT
40	I/O	PH.3 / EBI_ADR4 / SPI1_I2SMCLK / UART5_nCTS / UART4_RXD / I2C0_SDA / TM3_EXT
41	I/O	PH.4 / EBI_ADR3 / SPI1_MISO / UART7_nRTS / UART6_TXD

Pin	Type	M467J2JHAE Pin Function
42	I/O	PH.5 / EBI_ADR2 / SPI1_MOSI / UART7_nCTS / UART6_RXD
43	I/O	PH.6 / EBI_ADR1 / SPI1_CLK / UART7_TXD / UART9_nCTS
44	I/O	PH.7 / EBI_ADR0 / SPI1_SS / UART7_RXD / UART9_nRTS
45	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / UART9_TXD / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / ACMP2_O / EADC2_ST
46	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / UART9_RXD / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / ACMP3_O
47	P	V_{SS}
48	P	V_{DD}
49	I/O	PE.8 / EBI_ADR10 / EMAC0_RMII_MDC / I2S0_BCLK / SPI2_CLK / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / EQEI2_INDEX / TRACE_DATA3 / ECAP3_IC0
50	I/O	PE.9 / EBI_ADR11 / EMAC0_RMII_MDIO / I2S0_MCLK / SPI2_MISO / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / EQEI2_A / TRACE_DATA2 / ECAP3_IC1
51	I/O	PE.10 / EBI_ADR12 / EMAC0_RMII_TXD0 / I2S0_DI / SPI2_MOSI / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / EQEI2_B / TRACE_DATA1 / ECAP3_IC2
52	I/O	PE.11 / EBI_ADR13 / EMAC0_RMII_TXD1 / I2S0_DO / SPI2_SS / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0 / KPI_COL7
53	I/O	PE.12 / EBI_ADR14 / EMAC0_RMII_TXEN / I2S0_LRCK / SPI2_I2SMCLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK / KPI_COL6
54	I/O	PE.13 / EBI_ADR15 / EMAC0_PPS / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0 / TRACE_SWO / KPI_COL5
55	I/O	PC.8 / EBI_ADR16 / EMAC0_RMII_REFCLK / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4 / KPI_COL4
56	I/O	PC.7 / EBI_AD9 / EMAC0_RMII_RXD0 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / UART6_TXD / ACMP2_WLAT / EPWM1_CH2 / BPWM1_CH0 / CAN3_TXD / TM0 / INT3 / KPI_COL3
57	I/O	PC.6 / EBI_AD8 / EMAC0_RMII_RXD1 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / ACMP3_WLAT / EPWM1_CH3 / BPWM1_CH1 / CAN3_RXD / TM1 / INT2 / KPI_COL2
58	I/O	PA.7 / EBI_AD7 / EMAC0_RMII_CRSDV / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
59	I/O	PA.6 / EBI_AD6 / EMAC0_RMII_RXERR / SPI1_SS / SD1_nCD / SC2_CLK / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
60	P	V_{SS}
61	P	V_{DD}
62	A	LDO_CAP
63	I/O	PA.5 / SPIM_D2 / QSPI0_MISO1 / SPI1_I2SMCLK / SD1_CMD / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
64	I/O	PA.4 / SPIM_D3 / QSPI0_MOSI1 / SPI0_I2SMCLK / SD1_CLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
65	I/O	PA.3 / SPIM_SS / QSPI0_SS / SPI0_SS / SD1_DAT3 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
66	I/O	PA.2 / SPIM_CLK / QSPI0_CLK / SPI0_CLK / SD1_DAT2 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / EQEI3_INDEX / PSIO0_CH5
67	I/O	PA.1 / SPIM_MISO / QSPI0_MISO0 / SPI0_MISO / SD1_DAT1 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / CCAP_DATA7 / BPWM0_CH1 / EPWM0_CH4 / EQEI3_A / DAC1_ST / PSIO0_CH6
68	I/O	PA.0 / SPIM_MOSI / QSPI0_MOSI0 / SPI0_MOSI / SD1_DAT0 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / CCAP_DATA6 / BPWM0_CH0 / EPWM0_CH5 / EQEI3_B / DAC0_ST / PSIO0_CH7
69	P	V_{DDIO}
70	I/O	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / SD1_nCD / UART6_TXD / PSIO0_CH0
71	I/O	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / UART6_RXD / PSIO0_CH1
72	I	nRESET
73	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / SC1_DAT / I2S0_DO / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST
74	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / SC1_CLK / I2S0_LRCK / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / EADC1_ST
75	I/O	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / UART7_TXD / CAN2_TXD / PSIO0_CH2
76	I/O	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / UART7_RXD / CAN2_RXD / PSIO0_CH3
77	I/O	PC.5 / EBI_AD5 / SPIM_D2 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / CCAP_DATA5 / QSPI1_SS / I2C3_SMBAL / HBI_nCK / PSIO0_CH0 / KPI_ROW0
78	I/O	PC.4 / EBI_AD4 / SPIM_D3 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / CCAP_DATA4 / QSPI1_CLK / I2C3_SMBSUS / HBI_CK / PSIO0_CH1 / KPI_ROW1

Pin	Type	M467J2JHAE Pin Function
79	I/O	PC.3 / EBI_AD3 / SPIM_SS / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / CAN1_TXD / UART3_TXD / EPWM1_CH2 / CCAP_DATA3 / QSPI1_MISO0 / I2C3_SCL / HBI_nCS / PSIO0_CH2 / KPI_ROW2
80	I/O	PC.2 / EBI_AD2 / SPIM_CLK / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / CCAP_DATA2 / QSPI1_MOSI0 / I2C3_SDA / HBI_nRESET / PSIO0_CH3 / KPI_ROW3
81	I/O	PC.1 / EBI_AD1 / SPIM_MISO / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / CCAP_DATA1 / ACMP0_O / EADC0_ST / HBI_RWDS / KPI_ROW4
82	I/O	PC.0 / EBI_AD0 / SPIM_MOSI / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / CCAP_DATA0 / ACMP1_O / EADC1_ST / HBI_D2 / KPI_ROW5
83	P	V_{SS}
84	P	V_{DD}
85	I/O	PG.9 / EBI_AD0 / SD1_DAT3 / SPIM_D2 / QSPI1_MISO1 / CCAP_PIXCLK / I2C4_SCL / ECAP2_IC0 / BPWM0_CH5 / HBI_D4
86	I/O	PG.10 / EBI_AD1 / SD1_DAT2 / SPIM_D3 / QSPI1_MOSI1 / CCAP_SCLK / I2C4_SDA / ECAP2_IC1 / BPWM0_CH4 / HBI_D3
87	I/O	PG.11 / EBI_AD2 / SD1_DAT1 / SPIM_SS / QSPI1_SS / UART7_TXD / CCAP_SFIELD / I2C4_SMBAL / ECAP2_IC2 / BPWM0_CH3 / HBI_D0
88	I/O	PG.12 / EBI_AD3 / SD1_DAT0 / SPIM_CLK / QSPI1_CLK / UART7_RXD / CCAP_VSYNC / I2C4_SMBSUS / BPWM0_CH2 / HBI_D1
89	I/O	PG.13 / EBI_AD4 / SD1_CMD / SPIM_MISO / QSPI1_MISO0 / UART6_TXD / CCAP_HSYNC / BPWM0_CH1 / HBI_D5
90	I/O	PG.14 / EBI_AD5 / SD1_CLK / SPIM_MOSI / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0 / HBI_D6
91	I/O	PG.15 / SD1_nCD / CLKO / EADC0_ST / HBI_D7
92	I/O	PD.3 / EBI_AD10 / USCI0_CTL1 / SPI0_SS / UART3_nRTS / SC2_PWR / SC1_nCD / UART0_TXD / I2S1_BCLK / EQEI3_A
93	I/O	PD.2 / EBI_AD11 / USCI0_DAT1 / SPI0_CLK / UART3_nCTS / SC2_RST / UART0_RXD / I2S1_MCLK / EQEI3_B
94	I/O	PD.1 / EBI_AD12 / USCI0_DAT0 / SPI0_MISO / UART3_TXD / I2C2_SCL / SC2_DAT / I2S1_DI / EQEI2_INDEX / ECAP2_IC0
95	I/O	PD.0 / EBI_AD13 / USCI0_CLK / SPI0_MOSI / UART3_RXD / I2C2_SDA / SC2_CLK / I2S1_DO / EQEI2_A / ECAP2_IC1 / TM2
96	I/O	PD.13 / EBI_AD10 / SD0_nCD / SPI0_I2SMCLK / SPI1_I2SMCLK / QSPI1_MOSI0 / SC2_nCD / SD1_CLK / UART6_RXD / I2S1_LRCK / BPWM0_CH0 / EQEI2_B / ECAP2_IC2 / CLKO / EADC0_ST
97	I/O	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / SD1_nCD / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / ECAP3_IC0 / USB_VBUS / PSIO0_CH4
98	I/O	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / ECAP3_IC1 / USB_D- / PSIO0_CH5
99	I/O	PA.14 / I2S0_DI / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / EQEI1_B / ECAP3_IC2 / USB_D+ / PSIO0_CH6
100	I/O	PA.15 / I2S0_DO / UART0_RXD / SPIM_MOSI / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / EQEI3_INDEX / USB_OTG_ID / PSIO0_CH7
101	A	HSUSB_VRES
102	P	HSUSB_VDD33
103	I/O	HSUSB_VBUS
104	A	HSUSB_D-
105	P	HSUSB_VSS
106	A	HSUSB_D+
107	A	HSUSB_VDD12_CAP
108	I	HSUSB_ID
109	I/O	PE.7 / SD0_CMD / SPIM_D2 / UART5_TXD / CAN1_TXD / UART9_nCTS / EQEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / ACMP2_O / PSIO0_CH0
110	I/O	PE.6 / SD0_CLK / SPIM_D3 / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / CAN1_RXD / UART9_nRTS / EQEI1_A / EPWM0_CH1 / BPWM0_CH4 / ACMP3_O / PSIO0_CH1
111	I/O	PE.5 / EBI_nRD / SD0_DAT3 / SPIM_SS / SPI3_SS / SC0_PWR / USCI0_CTL1 / UART6_TXD / UART7_nRTS / UART9_TXD / EQEI1_B / EPWM0_CH2 / BPWM0_CH3 / PSIO0_CH2
112	I/O	PE.4 / EBI_nWR / SD0_DAT2 / SPIM_CLK / SPI3_CLK / SC0_RST / USCI0_DAT1 / UART6_RXD / UART7_nCTS / UART9_RXD / EQEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / PSIO0_CH3
113	I/O	PE.3 / EBI_MCLK / SD0_DAT1 / SPIM_MISO / SPI3_MISO / SC0_DAT / USCI0_DAT0 / UART6_nRTS / UART7_TXD / UART8_nCTS / EQEI0_A / EPWM0_CH4 / BPWM0_CH1
114	I/O	PE.2 / EBI_ALE / SD0_DAT0 / SPIM_MOSI / SPI3_MOSI / SC0_CLK / USCI0_CLK / UART6_nCTS / UART7_RXD / UART8_nRTS / EQEI0_B / EPWM0_CH5 / BPWM0_CH0
115	P	V_{SS}

Pin	Type	M467J2JHAE Pin Function
116	P	V_{DD}
117	I/O	PE.1 / EBI_AD10 / QSPI0_MISO0 / SC2_DAT / I2S0_BCLK / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS / UART8_TXD
118	I/O	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SC2_CLK / I2S0_MCLK / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS / UART8_RXD
119	I/O	PH.8 / EBI_AD12 / QSPI0_CLK / SC2_PWR / I2S0_DI / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD / UART9_nCTS
120	I/O	PH.9 / EBI_AD13 / QSPI0_SS / SC2_RST / I2S0_DO / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD / UART9_nRTS
121	I/O	PH.10 / EBI_AD14 / QSPI0_MISO1 / SC2_nCD / I2S0_LRCK / SPI1_I2SMCLK / UART4_TXD / UART0_TXD / UART9_TXD
122	I/O	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5 / UART9_RXD
123	I/O	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SC1_nCD / SPI0_I2SMCLK / I2S1_BCLK / EPWM0_CH4
124	I/O	PG.5 / EBI_nCS1 / SPI3_SS / SC1_PWR / I2C3_SMBAL / I2S1_MCLK / EPWM0_CH3
125	I/O	PG.6 / EBI_nCS2 / SPI3_CLK / SC1_RST / I2C3_SMBSUS / I2S1_DI / EPWM0_CH2
126	I/O	PG.7 / EBI_nWRL / SPI3_MISO / SC1_DAT / I2C3_SCL / I2S1_DO / EPWM0_CH1
127	I/O	PG.8 / EBI_nWRH / SPI3_MOSI / SC1_CLK / I2C3_SDA / I2S1_LRCK / EPWM0_CH0
128	P	V_{SS}
129	A	LDO_CAP
130	P	V_{DD}
131	I/O	PC.14 / EBI_AD11 / SC1_nCD / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / ETMC_TRACE_CLK / TM1 / USB_VBUS_ST / HSUSB_VBUS_ST
132	I/O	PB.15 / EADC0_CH15 / EADC1_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / CCAP_DATA1 / EPWM0_BRAKE1 / EPWM1_CH0 / ETMC_TRACE_DATA0 / TM0_EXT / USB_VBUS_EN / HSUSB_VBUS_EN / PSIO0_CH0 / KPI_COL0
133	I/O	PB.14 / EADC0_CH14 / EADC1_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / CCAP_DATA0 / EPWM1_CH1 / ETMC_TRACE_DATA1 / TM1_EXT / CLKO / USB_VBUS_ST / PSIO0_CH1 / KPI_COL1
134	I/O	PB.13 / EADC0_CH13 / EADC1_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / CCAP_PIXCLK / EPWM1_CH2 / ETMC_TRACE_DATA2 / TM2_EXT / CAN3_TXD / PSIO0_CH2 / KPI_COL2
135	I/O	PB.12 / EADC0_CH12 / EADC1_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / CCAP_SCLK / EPWM1_CH3 / ETMC_TRACE_DATA3 / TM3_EXT / CAN3_RXD / PSIO0_CH3 / KPI_COL3
136	P	AV_{DD}
137	A	V_{REF}
138	P	AV_{SS}
139	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / EMAC0_RMII_MDC / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / CCAP_SFIELD / HSUSB_VBUS_ST
140	I/O	PB.10 / EADC0_CH10 / ACMP2_P3 / EBI_ADR17 / EMAC0_RMII_MDIO / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / CCAP_VSYNC / HSUSB_VBUS_EN
141	I/O	PB.9 / EADC0_CH9 / ACMP2_P2 / EBI_ADR18 / EMAC0_RMII_TXD0 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / CAN2_TXD / INT7 / CCAP_HSYNC
142	I/O	PB.8 / EADC0_CH8 / ACMP2_P1 / EBI_ADR19 / EMAC0_RMII_TXD1 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / CAN2_RXD / INT6 / EADC2_ST
143	I/O	PB.7 / EADC0_CH7 / EADC2_CH15 / ACMP2_P0 / EBI_nWRL / EMAC0_RMII_TXEN / CAN1_TXD / UART1_TXD / SD1_CMD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O / KPI_COL4
144	I/O	PB.6 / EADC0_CH6 / EADC2_CH14 / ACMP2_N / EBI_nWRH / EMAC0_PPS / CAN1_RXD / UART1_RXD / SD1_CLK / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O / KPI_COL5
145	-	NC

Table 4.1-14 M467J2JHAE Multi-function Pin Table

4.1.4.8 M467 Series LQFP176 EPAD Pin Multi-function Pin Diagram

Corresponding Part Number: M467H2JHAE

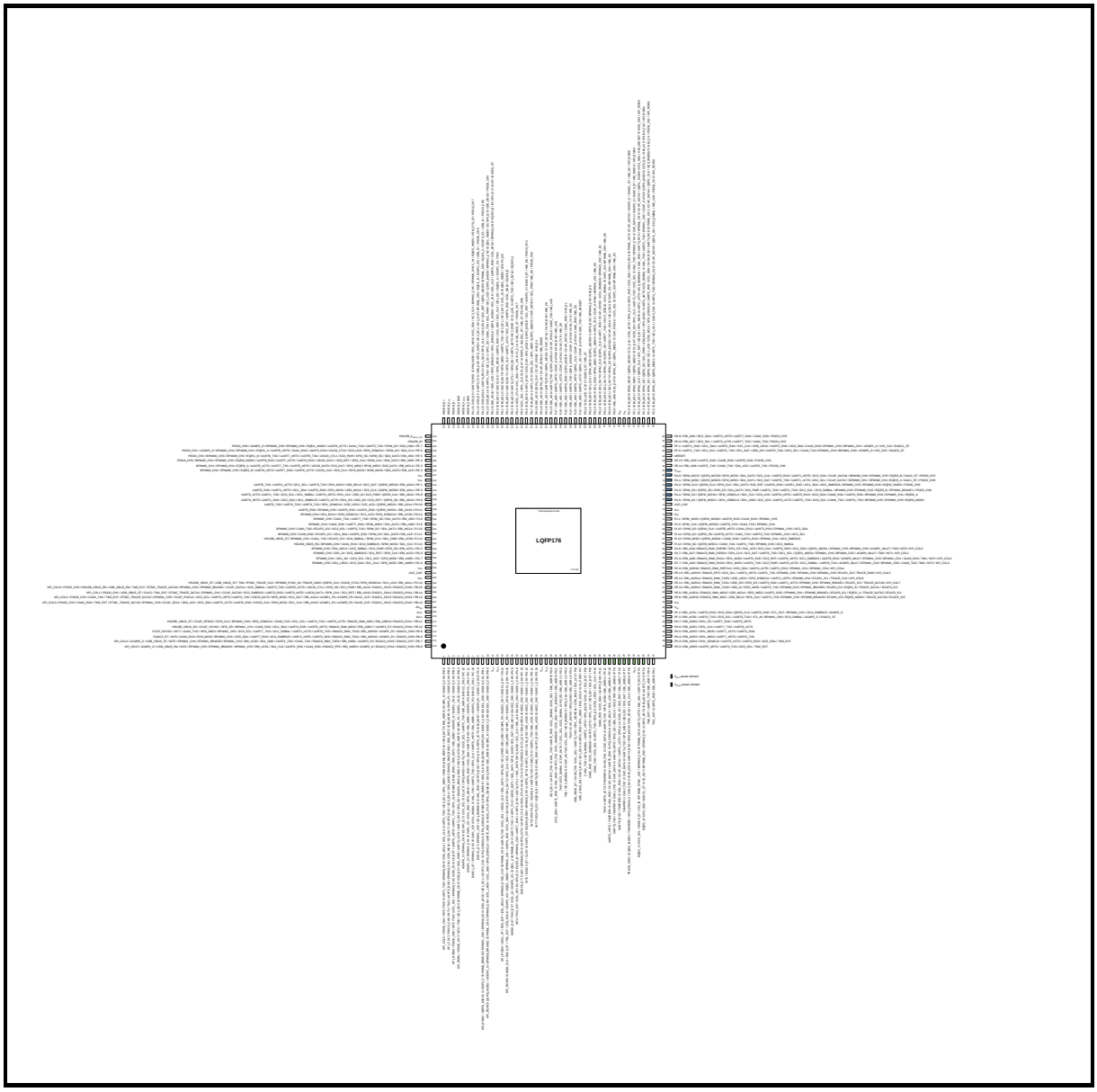


Figure 4.1-30 M467H2JHAE Multi-function Pin Diagram

Pin	Type	M467H2JHAE Pin Function
1	I/O	PB.5 / EADC0_CH5 / ACMP1_N / EBI_ADR0 / SD0_DAT3 / EMAC0_RMII_REFCLK / SPI1_MISO / I2C0_SCL / UART5_TXD / SC0_CLK / I2S0_BCLK / EPWM0_CH0 / UART2_TXD / TM0 / INT0 / PSIO0_CH4 / KPI_COL6
2	I/O	PB.4 / EADC0_CH4 / ACMP1_P1 / EBI_ADR1 / SD0_DAT2 / EMAC0_RMII_RXD0 / SPI1_MOSI / I2C0_SDA / UART5_RXD / SC0_DAT / I2S0_MCLK / EPWM0_CH1 / UART2_RXD / TM1 / INT1 / PSIO0_CH5 / KPI_COL7
3	I/O	PB.3 / EADC0_CH3 / EADC1_CH11 / ACMP0_N / EBI_ADR2 / SD0_DAT1 / EMAC0_RMII_RXD1 / SPI1_CLK / UART1_TXD / UART5_nRTS / SC0_RST / I2S0_DI / EPWM0_CH2 / I2C1_SCL / TM2 / INT2 / PSIO0_CH6 / KPI_ROW0
4	I/O	PB.2 / EADC0_CH2 / EADC1_CH10 / ACMP0_P1 / EBI_ADR3 / SD0_DAT0 / EMAC0_RMII_CRSDV / SPI1_SS / UART1_RXD / UART5_nCTS / SC0_PWR / I2S0_DO / EPWM0_CH3 / I2C1_SDA / TM3 / INT3 / PSIO0_CH7 / KPI_ROW1
5	I/O	PC.12 / EADC2_CH13 / EBI_ADR4 / UART0_TXD / I2C0_SCL / UART6_TXD / SPI3_MISO /

Pin	Type	M467H2JHAE Pin Function
		SC0_nCD / I2C4_SCL / ECAP1_IC2 / EPWM1_CH0 / ACMP0_O
6	I/O	PC.11 / EADC2_CH12 / ACMP3_P3 / EBI_ADR5 / UART0_RXD / I2C0_SDA / UART6_RXD / SPI3_MOSI / I2C4_SDA / ECAP1_IC1 / EPWM1_CH1 / ACMP1_O
7	I/O	PC.10 / EADC2_CH11 / ACMP3_P2 / EBI_ADR6 / UART6_nRTS / SPI3_CLK / UART3_TXD / CAN1_TXD / I2C4_SMBAL / ECAP1_IC0 / EPWM1_CH2 / EADC1_ST
8	I/O	PC.9 / EADC2_CH10 / ACMP3_P1 / EBI_ADR7 / UART6_nCTS / SPI3_SS / UART3_RXD / CAN1_RXD / I2C4_SMBSUS / EPWM1_CH3 / EADC1_ST
9	I/O	PB.1 / EADC0_CH1 / EADC1_CH9 / EADC2_CH9 / ACMP3_P0 / EBI_ADR8 / SD0_CLK / EMAC0_RMII_RXERR / SPI1_I2SMCLK / SPI3_I2SMCLK / UART2_TXD / I2C1_SCL / I2S0_LRCK / EPWM0_CH4 / EPWM1_CH4 / EPWM0_BRAKE0 / ACMP2_O / QSPI0_MISO1 / KPI_ROW2
10	I/O	PB.0 / EADC0_CH0 / EADC1_CH8 / EADC2_CH8 / ACMP3_N / EBI_ADR9 / SD0_CMD / SPI2_I2SMCLK / USCI0_CTL0 / UART2_RXD / SPI0_I2SMCLK / I2C1_SDA / I2S1_LRCK / EPWM0_CH5 / EPWM1_CH5 / EPWM0_BRAKE1 / ACMP3_O / QSPI0_MOSI1 / KPI_ROW3
11	P	V_{SS}
12	P	V_{DD}
13	I/O	PA.11 / EADC1_CH7 / EADC2_CH7 / ACMP0_P0 / EBI_nRD / SC2_PWR / SPI2_SS / SD1_DAT3 / USCI0_CLK / I2C2_SCL / UART6_TXD / BPWM0_CH0 / EPWM0_SYNC_OUT / I2S1_BCLK / TM0_EXT / DAC1_ST / KPI_ROW4
14	I/O	PA.10 / EADC1_CH6 / EADC2_CH6 / ACMP1_P0 / EBI_nWR / SC2_RST / SPI2_CLK / SD1_DAT2 / USCI0_DAT0 / I2C2_SDA / UART6_RXD / BPWM0_CH1 / EQEI1_INDEX / ECAP0_IC0 / I2S1_MCLK / TM1_EXT / DAC0_ST / SWDH_CLK / KPI_ROW5
15	I/O	PA.9 / EADC1_CH5 / EADC2_CH5 / EBI_MCLK / SC2_DAT / SPI2_MISO / SD1_DAT1 / USCI0_DAT1 / UART1_TXD / UART7_TXD / BPWM0_CH2 / EQEI1_A / ECAP0_IC1 / I2S1_DI / TM2_EXT / SWDH_DAT
16	I/O	PA.8 / EADC1_CH4 / EADC2_CH4 / EBI_ALE / SC2_CLK / SPI2_MOSI / SD1_DAT0 / USCI0_CTL1 / UART1_RXD / UART7_RXD / BPWM0_CH3 / EQEI1_B / ECAP0_IC2 / I2S1_DO / TM3_EXT / INT4
17	I/O	PC.13 / EADC1_CH3 / EADC2_CH3 / EBI_ADR10 / SC2_nCD / SPI2_I2SMCLK / CAN1_TXD / USCI0_CTL0 / UART2_TXD / UART8_nCTS / BPWM0_CH4 / CLKO / EADC0_ST
18	I/O	PD.12 / EADC1_CH2 / EADC2_CH2 / EBI_nCS0 / CAN1_RXD / UART2_RXD / UART8_nRTS / BPWM0_CH5 / EQEI0_INDEX / ECAP3_IC0 / CLKO / EADC0_ST / INT5
19	I/O	PD.11 / EADC1_CH1 / EADC2_CH1 / EBI_nCS1 / UART1_TXD / CAN0_TXD / UART8_TXD / EQEI0_A / ECAP3_IC1 / INT6
20	I/O	PD.10 / EADC1_CH0 / EADC2_CH0 / EBI_nCS2 / UART1_RXD / CAN0_RXD / UART8_RXD / EQEI0_B / ECAP3_IC2 / INT7
21	P	V_{SS}
22	P	V_{DD}
23	I/O	PG.0 / EBI_ADR8 / I2C0_SCL / I2C1_SMBAL / UART2_RXD / CAN1_TXD / UART1_TXD / I2C3_SCL
24	I/O	PG.1 / EBI_ADR9 / SPI2_I2SMCLK / I2C0_SDA / I2C1_SMBSUS / UART2_TXD / CAN1_RXD / UART1_RXD / I2C3_SDA
25	I/O	PG.2 / EBI_ADR11 / SPI2_SS / I2C0_SMBAL / I2C1_SCL / CCAP_DATA7 / I2C3_SMBAL / TM0
26	I/O	PG.3 / EBI_ADR12 / SPI2_CLK / I2C0_SMBSUS / I2C1_SDA / CCAP_DATA6 / I2C3_SMBSUS / TM1
27	I/O	PG.4 / EBI_ADR13 / SPI2_MISO / CCAP_DATA5 / TM2
28	I/O	PI.6 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_TXD / I2C1_SCL / CAN3_TXD / USB_VBUS_ST
29	I/O	PI.7 / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_RXD / I2C1_SDA / CAN3_RXD / USB_VBUS_EN
30	I/O	PI.8 / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nRTS / I2C0_SMBAL / CAN2_TXD
31	I/O	PI.9 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_nCTS / I2C0_SMBSUS / CAN2_RXD
32	I/O	PI.10 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_TXD / I2C0_SCL / CAN3_TXD
33	I/O	PI.11 / UART2_RXD / I2C0_SDA / CAN3_RXD
34	I/O	PF.11 / EBI_ADR14 / SPI2_MOSI / UART5_TXD / CCAP_DATA4 / CAN3_TXD / TAMPER5 / UART9_nCTS / TM3
35	I/O	PF.10 / EBI_ADR15 / SC0_nCD / I2S0_BCLK / SPI0_I2SMCLK / UART5_RXD / CCAP_DATA3 / CAN3_RXD / TAMPER4 / UART9_nRTS
36	I/O	PF.9 / EBI_ADR16 / SC0_PWR / I2S0_MCLK / SPI0_SS / UART5_nRTS / CCAP_DATA2 / CAN1_TXD / TAMPER3 / UART9_TXD
37	I/O	PF.8 / EBI_ADR17 / SC0_RST / I2S0_DI / SPI0_CLK / UART5_nCTS / CCAP_DATA1 / CAN1_RXD / TAMPER2 / UART9_RXD
38	I/O	PF.7 / EBI_ADR18 / SC0_DAT / I2S0_DO / SPI0_MISO / UART4_TXD / CCAP_DATA0 / CAN2_TXD / TAMPER1
39	I/O	PF.6 / EBI_ADR19 / SC0_CLK / I2S0_LRCK / SPI0_MOSI / UART4_RXD / EBI_nCS0 / CAN2_RXD / SPI3_I2SMCLK / TAMPER0 / EQEI2_INDEX / TRACE_SWO
40	P	V_{BAT}
41	I/O	PF.5 / UART2_RXD / EBI_AD1 / UART2_nCTS / EPWM0_CH0 / BPWM0_CH4 / EPWM0_SYNC_OUT / X32_IN / EADC0_ST / I2C4_SCL / EQEI2_A

Pin	Type	M467H2JHAE Pin Function
42	I/O	PF.4 / UART2_TXD / EBI_AD0 / UART2_nRTS / EPWM0_CH1 / BPWM0_CH5 / X32_OUT / EADC1_ST / I2C4_SDA / EQEI2_B
43	I/O	PH.0 / EBI_ADR7 / UART5_TXD / TM0_EXT
44	I/O	PH.1 / EBI_ADR6 / UART5_RXD / TM1_EXT
45	I/O	PH.2 / EBI_ADR5 / UART5_nRTS / UART4_TXD / I2C0_SCL / TM2_EXT
46	I/O	PH.3 / EBI_ADR4 / SPI1_I2SMCLK / UART5_nCTS / UART4_RXD / I2C0_SDA / TM3_EXT
47	I/O	PH.4 / EBI_ADR3 / SPI1_MISO / UART7_nRTS / UART6_TXD
48	I/O	PH.5 / EBI_ADR2 / SPI1_MOSI / UART7_nCTS / UART6_RXD
49	I/O	PH.6 / EBI_ADR1 / SPI1_CLK / UART7_TXD / UART9_nCTS
50	I/O	PH.7 / EBI_ADR0 / SPI1_SS / UART7_RXD / UART9_nRTS
51	I/O	PF.3 / EBI_nCS0 / UART0_TXD / I2C0_SCL / UART9_TXD / XT1_IN / BPWM1_CH0 / I2C4_SMBAL / ACMP2_O / EADC2_ST
52	I/O	PF.2 / EBI_nCS1 / UART0_RXD / I2C0_SDA / QSPI0_CLK / UART9_RXD / XT1_OUT / BPWM1_CH1 / I2C4_SMBSUS / ACMP3_O
53	P	V_{SS}
54	P	V_{DD}
55	I/O	PE.8 / EBI_ADR10 / EMAC0_RMII_MDC / I2S0_BCLK / SPI2_CLK / UART2_TXD / EPWM0_CH0 / EPWM0_BRAKE0 / ECAP0_IC0 / EQEI2_INDEX / TRACE_DATA3 / ECAP3_IC0
56	I/O	PE.9 / EBI_ADR11 / EMAC0_RMII_MDIO / I2S0_MCLK / SPI2_MISO / UART2_RXD / EPWM0_CH1 / EPWM0_BRAKE1 / ECAP0_IC1 / EQEI2_A / TRACE_DATA2 / ECAP3_IC1
57	I/O	PE.10 / EBI_ADR12 / EMAC0_RMII_TXD0 / I2S0_DI / SPI2_MOSI / UART3_TXD / EPWM0_CH2 / EPWM1_BRAKE0 / ECAP0_IC2 / EQEI2_B / TRACE_DATA1 / ECAP3_IC2
58	I/O	PE.11 / EBI_ADR13 / EMAC0_RMII_TXD1 / I2S0_DO / SPI2_SS / UART3_RXD / UART1_nCTS / EPWM0_CH3 / EPWM1_BRAKE1 / ECAP1_IC2 / TRACE_DATA0 / KPI_COL7
59	I/O	PE.12 / EBI_ADR14 / EMAC0_RMII_TXEN / I2S0_LRCK / SPI2_I2SMCLK / UART1_nRTS / EPWM0_CH4 / ECAP1_IC1 / TRACE_CLK / KPI_COL6
60	I/O	PE.13 / EBI_ADR15 / EMAC0_PPS / I2C0_SCL / UART4_nRTS / UART1_TXD / EPWM0_CH5 / EPWM1_CH0 / BPWM1_CH5 / ECAP1_IC0 / TRACE_SWO / KPI_COL5
61	I/O	PC.8 / EBI_ADR16 / EMAC0_RMII_REFCLK / I2C0_SDA / UART4_nCTS / UART1_RXD / EPWM1_CH1 / BPWM1_CH4 / KPI_COL4
62	I/O	PC.7 / EBI_AD9 / EMAC0_RMII_RXD0 / SPI1_MISO / UART4_TXD / SC2_PWR / UART0_nCTS / I2C1_SMBAL / UART6_TXD / ACMP2_WLAT / EPWM1_CH2 / BPWM1_CH0 / CAN3_TXD / TM0 / INT3 / KPI_COL3
63	I/O	PC.6 / EBI_AD8 / EMAC0_RMII_RXD1 / SPI1_MOSI / UART4_RXD / SC2_RST / UART0_nRTS / I2C1_SMBSUS / UART6_RXD / ACMP3_WLAT / EPWM1_CH3 / BPWM1_CH1 / CAN3_RXD / TM1 / INT2 / KPI_COL2
64	I/O	PA.7 / EBI_AD7 / EMAC0_RMII_CRSDV / SPI1_CLK / SC2_DAT / UART0_TXD / I2C1_SCL / QSPI1_MISO1 / EPWM1_CH4 / BPWM1_CH2 / ACMP0_WLAT / TM2 / INT1 / KPI_COL1
65	I/O	PA.6 / EBI_AD6 / EMAC0_RMII_RXERR / SPI1_SS / SD1_nCD / SC2_CLK / UART0_RXD / I2C1_SDA / QSPI1_MOSI1 / EPWM1_CH5 / BPWM1_CH3 / ACMP1_WLAT / TM3 / INT0 / KPI_COL0
66	I/O	PI.12 / SPIM_SS / QSPI0_MISO1 / CAN0_TXD / UART4_TXD / EPWM1_CH0 / I2C3_SMBAL
67	I/O	PI.13 / SPIM_MISO / QSPI0_MOSI1 / CAN0_RXD / UART4_RXD / EPWM1_CH1 / I2C3_SMBSUS
68	I/O	PI.14 / SPIM_D2 / QSPI0_SS / UART8_nCTS / CAN1_TXD / UART3_TXD / EPWM1_CH2 / I2C3_SCL
69	I/O	PI.15 / SPIM_D3 / QSPI0_CLK / UART8_nRTS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / I2C3_SDA
70	I/O	PJ.0 / SPIM_CLK / QSPI0_MISO0 / UART8_TXD / CAN2_TXD / EPWM1_CH4
71	I/O	PJ.1 / SPIM_MOSI / QSPI0_MOSI0 / UART8_RXD / CAN2_RXD / EPWM1_CH5
72	P	V_{SS}
73	P	V_{DD}
74	A	LDO_CAP
75	I/O	PA.5 / SPIM_D2 / QSPI0_MISO1 / SPI1_I2SMCLK / SD1_CMD / SC2_nCD / UART0_nCTS / UART5_TXD / I2C0_SCL / CAN0_TXD / UART0_TXD / BPWM0_CH5 / EPWM0_CH0 / EQEI0_INDEX
76	I/O	PA.4 / SPIM_D3 / QSPI0_MOSI1 / SPI0_I2SMCLK / SD1_CLK / SC0_nCD / UART0_nRTS / UART5_RXD / I2C0_SDA / CAN0_RXD / UART0_RXD / BPWM0_CH4 / EPWM0_CH1 / EQEI0_A
77	I/O	PA.3 / SPIM_SS / QSPI0_SS / SPI0_SS / SD1_DAT3 / SC0_PWR / UART4_TXD / UART1_TXD / I2C1_SCL / I2C0_SMBAL / BPWM0_CH3 / EPWM0_CH2 / EQEI0_B / EPWM1_BRAKE1 / PSIO0_CH4
78	I/O	PA.2 / SPIM_CLK / QSPI0_CLK / SPI0_CLK / SD1_DAT2 / SC0_RST / UART4_RXD / UART1_RXD / I2C1_SDA / I2C0_SMBSUS / BPWM0_CH2 / EPWM0_CH3 / EQEI3_INDEX / PSIO0_CH5
79	I/O	PA.1 / SPIM_MISO / QSPI0_MISO0 / SPI0_MISO / SD1_DAT1 / SC0_DAT / UART0_TXD / UART1_nCTS / I2C2_SCL / CCAP_DATA7 / BPWM0_CH1 / EPWM0_CH4 / EQEI3_A / DAC1_ST / PSIO0_CH6
80	I/O	PA.0 / SPIM_MOSI / QSPI0_MOSI0 / SPI0_MOSI / SD1_DAT0 / SC0_CLK / UART0_RXD / UART1_nRTS / I2C2_SDA / CCAP_DATA6 / BPWM0_CH0 / EPWM0_CH5 / EQEI3_B / DAC0_ST /

Pin	Type	M467H2JHAE Pin Function
		PSIO0_CH7
81	P	V_{DDIO}
82	I/O	PE.14 / EBI_AD8 / UART2_TXD / CAN0_TXD / SD1_nCD / UART6_TXD / PSIO0_CH0
83	I/O	PE.15 / EBI_AD9 / UART2_RXD / CAN0_RXD / UART6_RXD / PSIO0_CH1
84	I	nRESET
85	I/O	PF.0 / UART1_TXD / I2C1_SCL / UART0_TXD / SC1_DAT / I2S0_DO / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / BPWM1_CH0 / ACMP0_O / ICE_DAT / EADC0_ST
86	I/O	PF.1 / UART1_RXD / I2C1_SDA / UART0_RXD / SC1_CLK / I2S0_LRCK / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / BPWM1_CH1 / ACMP1_O / ICE_CLK / EADC1_ST
87	I/O	PD.9 / EBI_AD7 / I2C2_SCL / UART2_nCTS / UART7_TXD / CAN2_TXD / PSIO0_CH2
88	I/O	PD.8 / EBI_AD6 / I2C2_SDA / UART2_nRTS / UART7_RXD / CAN2_RXD / PSIO0_CH3
89	I/O	PC.5 / EBI_AD5 / SPIM_D2 / QSPI0_MISO1 / UART2_TXD / I2C1_SCL / CAN0_TXD / UART4_TXD / EPWM1_CH0 / CCAP_DATA5 / QSPI1_SS / I2C3_SMBAL / HBI_nCK / PSIO0_CH0 / KPI_ROW0
90	I/O	PC.4 / EBI_AD4 / SPIM_D3 / QSPI0_MOSI1 / SC1_nCD / I2S0_BCLK / SPI1_I2SMCLK / UART2_RXD / I2C1_SDA / CAN0_RXD / UART4_RXD / EPWM1_CH1 / CCAP_DATA4 / QSPI1_CLK / I2C3_SMBSUS / HBI_CK / PSIO0_CH1 / KPI_ROW1
91	I/O	PC.3 / EBI_AD3 / SPIM_SS / QSPI0_SS / SC1_PWR / I2S0_MCLK / SPI1_MISO / UART2_nRTS / I2C0_SMBAL / CAN1_TXD / UART3_TXD / EPWM1_CH2 / CCAP_DATA3 / QSPI1_MISO0 / I2C3_SCL / HBI_nCS / PSIO0_CH2 / KPI_ROW2
92	I/O	PC.2 / EBI_AD2 / SPIM_CLK / QSPI0_CLK / SC1_RST / I2S0_DI / SPI1_MOSI / UART2_nCTS / I2C0_SMBSUS / CAN1_RXD / UART3_RXD / EPWM1_CH3 / CCAP_DATA2 / QSPI1_MOSI0 / I2C3_SDA / HBI_nRESET / PSIO0_CH3 / KPI_ROW3
93	I/O	PC.1 / EBI_AD1 / SPIM_MISO / QSPI0_MISO0 / SC1_DAT / I2S0_DO / SPI1_CLK / UART2_TXD / I2C0_SCL / CAN2_TXD / EPWM1_CH4 / CCAP_DATA1 / ACMP0_O / EADC0_ST / HBI_RWDS / KPI_ROW4
94	I/O	PC.0 / EBI_AD0 / SPIM_MOSI / QSPI0_MOSI0 / SC1_CLK / I2S0_LRCK / SPI1_SS / UART2_RXD / I2C0_SDA / CAN2_RXD / EPWM1_CH5 / CCAP_DATA0 / ACMP1_O / EADC1_ST / HBI_D2 / KPI_ROW5
95	P	V_{SS}
96	P	V_{DD}
97	I/O	PG.9 / EBI_AD0 / SD1_DAT3 / SPIM_D2 / QSPI1_MISO1 / CCAP_PIXCLK / I2C4_SCL / ECAP2_IC0 / BPWM0_CH5 / HBI_D4
98	I/O	PG.10 / EBI_AD1 / SD1_DAT2 / SPIM_D3 / QSPI1_MOSI1 / CCAP_SCLK / I2C4_SDA / ECAP2_IC1 / BPWM0_CH4 / HBI_D3
99	I/O	PG.11 / EBI_AD2 / SD1_DAT1 / SPIM_SS / QSPI1_SS / UART7_TXD / CCAP_SFIELD / I2C4_SMBAL / ECAP2_IC2 / BPWM0_CH3 / HBI_D0
100	I/O	PG.12 / EBI_AD3 / SD1_DAT0 / SPIM_CLK / QSPI1_CLK / UART7_RXD / CCAP_VSYNC / I2C4_SMBSUS / BPWM0_CH2 / HBI_D1
101	I/O	PG.13 / EBI_AD4 / SD1_CMD / SPIM_MISO / QSPI1_MISO0 / UART6_TXD / CCAP_HSYNC / BPWM0_CH1 / HBI_D5
102	I/O	PG.14 / EBI_AD5 / SD1_CLK / SPIM_MOSI / QSPI1_MOSI0 / UART6_RXD / BPWM0_CH0 / HBI_D6
103	I/O	PG.15 / SD1_nCD / CLKO / EADC0_ST / HBI_D7
104	I/O	PJ.2 / EBI_AD5 / UART8_nCTS / QSPI1_SS / CCAP_DATA5 / CAN0_TXD / HBI_nRESET
105	I/O	PJ.3 / EBI_AD4 / UART8_nRTS / QSPI1_CLK / CCAP_DATA4 / CAN0_RXD / HBI_D3
106	I/O	PJ.4 / EBI_AD3 / UART8_TXD / QSPI1_MISO0 / CCAP_DATA3 / CAN1_TXD / HBI_D2
107	I/O	PJ.5 / EBI_AD2 / UART8_RXD / QSPI1_MOSI0 / CCAP_DATA2 / CAN1_RXD / HBI_D1
108	I/O	PJ.6 / EBI_AD1 / UART9_nCTS / CCAP_DATA1 / CAN2_TXD / HBI_D0
109	I/O	PJ.7 / EBI_AD0 / UART9_nRTS / CCAP_DATA0 / CAN2_RXD / HBI_nCS
110	I/O	PH.12 / EBI_AD0 / UART9_TXD / QSPI1_MISO1 / CCAP_PIXCLK / CAN3_TXD / HBI_nCK
111	I/O	PH.13 / EBI_AD1 / UART9_RXD / QSPI1_MOSI1 / CCAP_SCLK / CAN3_RXD / HBI_CK
112	I/O	PH.14 / EBI_AD2 / QSPI1_SS / CCAP_SFIELD / HBI_RWDS
113	I/O	PH.15 / EBI_AD3 / QSPI1_CLK / CCAP_VSYNC / HBI_D4
114	I/O	PD.7 / EBI_AD4 / UART1_TXD / I2C0_SCL / SPI1_MISO / QSPI1_MISO0 / CCAP_HSYNC / SC1_PWR / HBI_D5 / PSIO0_CH4
115	I/O	PD.6 / EBI_AD5 / UART1_RXD / I2C0_SDA / SPI1_MOSI / QSPI1_MOSI0 / SC1_RST / ACMP0_O / EADC0_ST / HBI_D6 / PSIO0_CH5
116	I/O	PD.5 / I2C1_SCL / SPI1_CLK / SC1_DAT / ACMP1_O / EADC1_ST / HBI_D7 / PSIO0_CH6
117	I/O	PD.4 / USCI0_CTL0 / I2C1_SDA / SPI1_SS / SC1_CLK / USB_VBUS_ST / PSIO0_CH7
118	I/O	PD.3 / EBI_AD10 / USCI0_CTL1 / SPI0_SS / UART3_nRTS / SC2_PWR / SC1_nCD / UART0_TXD / I2S1_BCLK / EQEI3_A
119	I/O	PD.2 / EBI_AD11 / USCI0_DAT1 / SPI0_CLK / UART3_nCTS / SC2_RST / UART0_RXD / I2S1_MCLK / EQEI3_B
120	I/O	PD.1 / EBI_AD12 / USCI0_DAT0 / SPI0_MISO / UART3_TXD / I2C2_SCL / SC2_DAT / I2S1_DI / EQEI2_INDEX / ECAP2_IC0
121	I/O	PD.0 / EBI_AD13 / USCI0_CLK / SPI0_MOSI / UART3_RXD / I2C2_SDA / SC2_CLK / I2S1_DO / EQEI2_A / ECAP2_IC1 / TM2

Pin	Type	M467H2JHAE Pin Function
122	I/O	PD.13 / EBI_AD10 / SD0_nCD / SPI0_I2SMCLK / SPI1_I2SMCLK / QSPI1_MOSI0 / SC2_nCD / SD1_CLK / UART6_RXD / I2S1_LRCK / BPWM0_CH0 / EQEI2_B / ECAP2_IC2 / CLKO / EADC0_ST
123	I/O	PA.12 / I2S0_BCLK / UART4_TXD / I2C1_SCL / SPI2_SS / CAN0_TXD / SC2_PWR / SD1_nCD / QSPI1_MISO0 / BPWM1_CH2 / EQEI1_INDEX / ECAP3_IC0 / USB_VBUS / PSIO0_CH4
124	I/O	PA.13 / I2S0_MCLK / UART4_RXD / I2C1_SDA / SPI2_CLK / CAN0_RXD / SC2_RST / QSPI1_MOSI0 / BPWM1_CH3 / EQEI1_A / ECAP3_IC1 / USB_D- / PSIO0_CH5
125	I/O	PA.14 / I2S0_DI / UART0_TXD / EBI_AD5 / SPI2_MISO / I2C2_SCL / SC2_DAT / BPWM1_CH4 / EQEI1_B / ECAP3_IC2 / USB_D+ / PSIO0_CH6
126	I/O	PA.15 / I2S0_DO / UART0_RXD / SPIM_MOSI / SPI2_MOSI / I2C2_SDA / SC2_CLK / BPWM1_CH5 / EPWM0_SYNC_IN / EQEI3_INDEX / USB_OTG_ID / PSIO0_CH7
127	A	HSUSB_VRES
128	P	HSUSB_VDD33
129	I/O	HSUSB_VBUS
130	A	HSUSB_D-
131	P	HSUSB_VSS
132	A	HSUSB_D+
133	A	HSUSB_VDD12_CAP
134	I	HSUSB_ID
135	I/O	PE.7 / SD0_CMD / SPIM_D2 / UART5_TXD / CAN1_TXD / UART9_nCTS / EQEI1_INDEX / EPWM0_CH0 / BPWM0_CH5 / ACMP2_O / PSIO0_CH0
136	I/O	PE.6 / SD0_CLK / SPIM_D3 / SPI3_I2SMCLK / SC0_nCD / USCI0_CTL0 / UART5_RXD / CAN1_RXD / UART9_nRTS / EQEI1_A / EPWM0_CH1 / BPWM0_CH4 / ACMP3_O / PSIO0_CH1
137	I/O	PE.5 / EBI_nRD / SD0_DAT3 / SPIM_SS / SPI3_SS / SC0_PWR / USCI0_CTL1 / UART6_TXD / UART7_nRTS / UART9_TXD / EQEI1_B / EPWM0_CH2 / BPWM0_CH3 / PSIO0_CH2
138	I/O	PE.4 / EBI_nWR / SD0_DAT2 / SPIM_CLK / SPI3_CLK / SC0_RST / USCI0_DAT1 / UART6_RXD / UART7_nCTS / UART9_RXD / EQEI0_INDEX / EPWM0_CH3 / BPWM0_CH2 / PSIO0_CH3
139	I/O	PE.3 / EBI_MCLK / SD0_DAT1 / SPIM_MISO / SPI3_MISO / SC0_DAT / USCI0_DAT0 / UART6_nRTS / UART7_TXD / UART8_nCTS / EQEI0_A / EPWM0_CH4 / BPWM0_CH1
140	I/O	PE.2 / EBI_ALE / SD0_DAT0 / SPIM_MOSI / SPI3_MOSI / SC0_CLK / USCI0_CLK / UART6_nCTS / UART7_RXD / UART8_nRTS / EQEI0_B / EPWM0_CH5 / BPWM0_CH0
141	P	V_{SS}
142	P	V_{DD}
143	I/O	PE.1 / EBI_AD10 / QSPI0_MISO0 / SC2_DAT / I2S0_BCLK / SPI1_MISO / UART3_TXD / I2C1_SCL / UART4_nCTS / UART8_TXD
144	I/O	PE.0 / EBI_AD11 / QSPI0_MOSI0 / SC2_CLK / I2S0_MCLK / SPI1_MOSI / UART3_RXD / I2C1_SDA / UART4_nRTS / UART8_RXD
145	I/O	PH.8 / EBI_AD12 / QSPI0_CLK / SC2_PWR / I2S0_DI / SPI1_CLK / UART3_nRTS / I2C1_SMBAL / I2C2_SCL / UART1_TXD / UART9_nCTS
146	I/O	PH.9 / EBI_AD13 / QSPI0_SS / SC2_RST / I2S0_DO / SPI1_SS / UART3_nCTS / I2C1_SMBSUS / I2C2_SDA / UART1_RXD / UART9_nRTS
147	I/O	PH.10 / EBI_AD14 / QSPI0_MISO1 / SC2_nCD / I2S0_LRCK / SPI1_I2SMCLK / UART4_TXD / UART0_TXD / UART9_TXD
148	I/O	PH.11 / EBI_AD15 / QSPI0_MOSI1 / UART4_RXD / UART0_RXD / EPWM0_CH5 / UART9_RXD
149	I/O	PD.14 / EBI_nCS0 / SPI3_I2SMCLK / SC1_nCD / SPI0_I2SMCLK / I2S1_BCLK / EPWM0_CH4
150	I/O	PJ.8 / EBI_nRD / SD1_DAT3 / SPIM_SS / UART7_TXD / CAN2_TXD / BPWM0_CH5
151	I/O	PJ.9 / EBI_nWR / SD1_DAT2 / SPIM_MISO / UART7_RXD / CAN2_RXD / BPWM0_CH4
152	I/O	PJ.10 / EBI_MCLK / SD1_DAT1 / SPIM_D2 / UART6_TXD / I2C4_SCL / ECAP2_IC0 / CAN0_TXD / BPWM0_CH3
153	I/O	PJ.11 / EBI_ALE / SD1_DAT0 / SPIM_D3 / UART6_RXD / I2C4_SDA / ECAP2_IC1 / CAN0_RXD / BPWM0_CH2
154	I/O	PJ.12 / EBI_nCS0 / SD1_CMD / SPIM_CLK / I2C4_SMBAL / ECAP2_IC2 / CAN1_TXD / BPWM0_CH1 / HSUSB_VBUS_ST
155	I/O	PJ.13 / SD1_CLK / SPIM_MOSI / I2C4_SMBSUS / CAN1_RXD / BPWM0_CH0 / HSUSB_VBUS_EN
156	I/O	PG.5 / EBI_nCS1 / SPI3_SS / SC1_PWR / I2C3_SMBAL / I2S1_MCLK / EPWM0_CH3
157	I/O	PG.6 / EBI_nCS2 / SPI3_CLK / SC1_RST / I2C3_SMBSUS / I2S1_DI / EPWM0_CH2
158	I/O	PG.7 / EBI_nWRL / SPI3_MISO / SC1_DAT / I2C3_SCL / I2S1_DO / EPWM0_CH1
159	I/O	PG.8 / EBI_nWRH / SPI3_MOSI / SC1_CLK / I2C3_SDA / I2S1_LRCK / EPWM0_CH0
160	P	V_{SS}
161	A	LDO_CAP
162	P	V_{DD}
163	I/O	PC.14 / EBI_AD11 / SC1_nCD / SPI0_I2SMCLK / USCI0_CTL0 / QSPI0_CLK / TRACE_SWO / EPWM0_SYNC_IN / ETMC_TRACE_CLK / TM1 / USB_VBUS_ST / HSUSB_VBUS_ST
164	I/O	PB.15 / EADC0_CH15 / EADC1_CH15 / EBI_AD12 / SC1_PWR / SPI0_SS / USCI0_CTL1 / UART0_nCTS / UART3_TXD / I2C2_SMBAL / CCAP_DATA1 / EPWM0_BRAKE1 / EPWM1_CH0 / ETMC_TRACE_DATA0 / TM0_EXT / USB_VBUS_EN / HSUSB_VBUS_EN / PSIO0_CH0 / KPI_COL0

Pin	Type	M467H2JHAE Pin Function
165	I/O	PB.14 / EADC0_CH14 / EADC1_CH14 / EBI_AD13 / SC1_RST / SPI0_CLK / USCI0_DAT1 / UART0_nRTS / UART3_RXD / I2C2_SMBSUS / CCAP_DATA0 / EPWM1_CH1 / ETMC_TRACE_DATA1 / TM1_EXT / CLKO / USB_VBUS_ST / PSIO0_CH1 / KPI_COL1
166	I/O	PB.13 / EADC0_CH13 / EADC1_CH13 / DAC1_OUT / ACMP0_P3 / ACMP1_P3 / EBI_AD14 / SC1_DAT / SPI0_MISO / USCI0_DAT0 / UART0_TXD / UART3_nRTS / I2C2_SCL / CCAP_PIXCLK / EPWM1_CH2 / ETMC_TRACE_DATA2 / TM2_EXT / CAN3_TXD / PSIO0_CH2 / KPI_COL2
167	I/O	PB.12 / EADC0_CH12 / EADC1_CH12 / DAC0_OUT / ACMP0_P2 / ACMP1_P2 / EBI_AD15 / SC1_CLK / SPI0_MOSI / USCI0_CLK / UART0_RXD / UART3_nCTS / I2C2_SDA / SD0_nCD / CCAP_SCLK / EPWM1_CH3 / ETMC_TRACE_DATA3 / TM3_EXT / CAN3_RXD / PSIO0_CH3 / KPI_COL3
168	P	AV_{DD}
169	A	V_{REF}
170	P	AV_{SS}
171	I/O	PB.11 / EADC0_CH11 / EBI_ADR16 / EMAC0_RMII_MDC / UART0_nCTS / UART4_TXD / I2C1_SCL / CAN0_TXD / SPI0_I2SMCLK / BPWM1_CH0 / SPI3_CLK / CCAP_SFIELD / HSUSB_VBUS_ST
172	I/O	PB.10 / EADC0_CH10 / ACMP2_P3 / EBI_ADR17 / EMAC0_RMII_MDIO / UART0_nRTS / UART4_RXD / I2C1_SDA / CAN0_RXD / BPWM1_CH1 / SPI3_SS / CCAP_VSYNC / HSUSB_VBUS_EN
173	I/O	PB.9 / EADC0_CH9 / ACMP2_P2 / EBI_ADR18 / EMAC0_RMII_TXD0 / UART0_TXD / UART1_nCTS / I2C1_SMBAL / UART7_TXD / I2C0_SCL / BPWM1_CH2 / SPI3_MISO / CAN2_TXD / INT7 / CCAP_HSYNC
174	I/O	PB.8 / EADC0_CH8 / ACMP2_P1 / EBI_ADR19 / EMAC0_RMII_TXD1 / UART0_RXD / UART1_nRTS / I2C1_SMBSUS / UART7_RXD / I2C0_SDA / BPWM1_CH3 / SPI3_MOSI / CAN2_RXD / INT6 / EADC2_ST
175	I/O	PB.7 / EADC0_CH7 / EADC2_CH15 / ACMP2_P0 / EBI_nWRL / EMAC0_RMII_TXEN / CAN1_TXD / UART1_TXD / SD1_CMD / EBI_nCS0 / BPWM1_CH4 / EPWM1_BRAKE0 / EPWM1_CH4 / INT5 / USB_VBUS_ST / ACMP0_O / KPI_COL4
176	I/O	PB.6 / EADC0_CH6 / EADC2_CH14 / ACMP2_N / EBI_nWRH / EMAC0_PPS / CAN1_RXD / UART1_RXD / SD1_CLK / EBI_nCS1 / BPWM1_CH5 / EPWM1_BRAKE1 / EPWM1_CH5 / INT4 / USB_VBUS_EN / ACMP1_O / KPI_COL5
177	-	NC

Table 4.1-15 M467H2JHAE Multi-function Pin Table

4.2 Pin Mapping Table

4.2.1 M463 Series Mapping Table

The following is the pin mapping table of M463 series

	M463 Series						
Pin Name	M463YG	M463LG	M463L2G	M463SG	M463S2G	M463KG	M463K2G
PB.5	1	1	1	2	2	1	1
PB.4	2	2	2	3	3	2	2
PB.3	3	3	3	4	4	3	3
PB.2	4	4	4	5	5	4	4
PC.12						5	5
PC.11						6	6
PC.10						7	7
PC.9						8	8
PB.1	5	5	5	6	6	9	9
PB.0	6	6	6	7	7	10	10
V_{SS}						11	11
V_{DD}						12	12
PA.11	7	7	7	8	8	13	13
PA.10	8	8	8	9	9	14	14
PA.9	9	9	9	10	10	15	15
PA.8	10	10	10	11	11	16	16
PC.13						17	17
PD.12						18	18
PD.11						19	19
PD.10						20	20
V_{SS}							
V_{DD}							
PG.2						21	21
PG.3						22	22
PG.4						23	23
PF.11						24	24
PF.10						25	25
PF.9						26	26
PF.8						27	27

PF.7						28	28
PF.6				12	12	29	29
V_{BAT}				13	13	30	30
PF.5	11	11	11	14	14	31	31
PF.4	12	12	12	15	15	32	32
PH.4						33	33
PH.5						34	34
PH.6						35	35
PH.7						36	36
PF.3	13	13	13	16	16	37	37
PF.2	14	14	14	17	17	38	38
V_{SS}						39	39
V_{DD}						40	40
PE.8						41	41
PE.9						42	42
PE.10						43	43
PE.11						44	44
PE.12						45	45
PE.13						46	46
PC.8						47	47
PC.7				18	18	48	48
PC.6				19	19	49	49
PA.7	15	15	15	20	20	50	50
PA.6	16	16	16	21	21	51	51
V_{SS}				22	22	52	52
V_{DD}				23	23	53	53
LDO_CAP				24	24	54	54
PA.5	17	17	17	25	25	55	55
PA.4	18	18	18	26	26	56	56
PA.3	19	19	19	27	27	57	57
PA.2	20	20	20	28	28	58	58
PA.1	21	21	21	29	29	59	59
PA.0	22	22	22	30	30	60	60
V_{DDIO}	23	23	23	31	31	61	61
PE.14						62	62

PE.15						63	63
nRESET	24	24	24	32	32	64	64
PF.0	25	25	25	33	33	65	65
PF.1	26	26	26	34	34	66	66
PD.9						67	67
PD.8						68	68
PC.5				35	35	69	69
PC.4				36	36	70	70
PC.3						71	71
PC.2						72	72
PC.1	27	27	27			73	73
PC.0	28	28	28			74	74
V_{SS}						75	75
V_{DD}						76	76
PG.9						77	77
PG.10						78	78
PG.11						79	79
PG.12						80	80
PG.13						81	81
PG.14						82	82
PG.15						83	83
PD.13						84	84
PA.12				37	37	85	85
PA.13				38	38	86	86
PA.14				39	39	87	87
PA.15				40	40	88	88
HSUSB_V_{RES}	29	29	29	41	41	89	89
HSUSB_VDD_{33}	30	30	30	42	42	90	90
HSUSB_VBUS	31	31	31	43	43	91	91
HSUSB_D-	32	32	32	44	44	92	92
HSUSB_VSS	33	33	33	45	45	93	93
HSUSB_D+	34	34	34	46	46	94	94
HSUSB_VDD_{12}_CAP	35	35	35	47	47	95	95
HSUSB_ID	36	36	36	48	48	96	96

PE.7						97	97
PE.6						98	98
PE.5						99	99
PE.4						100	100
PE.3						101	101
PE.2						102	102
V_{SS}	37	37	37	49	49	103	103
V_{DD}						104	104
PE.1						105	105
PE.0						106	106
PH.8						107	107
PH.9						108	108
PH.10						109	109
PH.11						110	110
PD.14						111	111
V_{SS}						112	112
LDO_CAP	38	38	38	50	50	113	113
V_{DD}	39	39	39	51	51	114	114
PC.14	40	40	40	52	52	115	115
PB.15	41	41	41	53	53	116	116
PB.14	42	42	42	54	54	117	117
PB.13	43	43	43	55	55	118	118
PB.12	44	44	44	56	56	119	119
AV_{DD}	45	45	45	57	57	120	120
V_{REF}				58	58	121	121
AV_{SS}	46	46	46	59	59	122	122
PB.11				60	60	123	123
PB.10				61	61	124	124
PB.9				62	62	125	125
PB.8				63	63	126	126
PB.7	47	47	47	64	64	127	127
PB.6	48	48	48	1	1	128	128
V_{SS}	49		49		65		129

4.2.2 M467 Series Mapping Table

The following is the pin mapping table of M467 series.

	M467 Series							
Pin Name	M467SJ	M467S2J	M467KJ	M467K2J	M467JJ	M467J2J	M467HJ	M467H2J
PB.5	2	2	1	1	1	1	1	1
PB.4	3	3	2	2	2	2	2	2
PB.3	4	4	3	3	3	3	3	3
PB.2	5	5	4	4	4	4	4	4
PC.12			5	5	5	5	5	5
PC.11			6	6	6	6	6	6
PC.10			7	7	7	7	7	7
PC.9			8	8	8	8	8	8
PB.1	6	6	9	9	9	9	9	9
PB.0	7	7	10	10	10	10	10	10
V_{SS}			11	11	11	11	11	11
V_{DD}			12	12	12	12	12	12
PA.11	8	8	13	13	13	13	13	13
PA.10	9	9	14	14	14	14	14	14
PA.9	10	10	15	15	15	15	15	15
PA.8	11	11	16	16	16	16	16	16
PC.13			17	17	17	17	17	17
PD.12			18	18	18	18	18	18
PD.11			19	19	19	19	19	19
PD.10			20	20	20	20	20	20
V_{SS}					21	21	21	21
V_{DD}					22	22	22	22
PG.0					23	23	23	23
PG.1					24	24	24	24
PG.2			21	21	25	25	25	25
PG.3			22	22	26	26	26	26
PG.4			23	23	27	27	27	27
PI.6							28	28
PI.7							29	29
PI.8							30	30
PI.9							31	31

PI.10							32	32
PI.11							33	33
PF.11			24	24	28	28	34	34
PF.10			25	25	29	29	35	35
PF.9			26	26	30	30	36	36
PF.8			27	27	31	31	37	37
PF.7			28	28	32	32	38	38
PF.6	12	12	29	29	33	33	39	39
V_{BAT}	13	13	30	30	34	34	40	40
PF.5	14	14	31	31	35	35	41	41
PF.4	15	15	32	32	36	36	42	42
PH.0					37	37	43	43
PH.1					38	38	44	44
PH.2					39	39	45	45
PH.3					40	40	46	46
PH.4			33	33	41	41	47	47
PH.5			34	34	42	42	48	48
PH.6			35	35	43	43	49	49
PH.7			36	36	44	44	50	50
PF.3	16	16	37	37	45	45	51	51
PF.2	17	17	38	38	46	46	52	52
V_{SS}			39	39	47	47	53	53
V_{DD}			40	40	48	48	54	54
PE.8			41	41	49	49	55	55
PE.9			42	42	50	50	56	56
PE.10			43	43	51	51	57	57
PE.11			44	44	52	52	58	58
PE.12			45	45	53	53	59	59
PE.13			46	46	54	54	60	60
PC.8			47	47	55	55	61	61
PC.7	18	18	48	48	56	56	62	62
PC.6	19	19	49	49	57	57	63	63
PA.7	20	20	50	50	58	58	64	64
PA.6	21	21	51	51	59	59	65	65
PI.12							66	66

PI.13							67	67
PI.14							68	68
PI.15							69	69
PJ.0							70	70
PJ.1							71	71
V_{SS}	22	22	52	52	60	60	72	72
V_{DD}	23	23	53	53	61	61	73	73
LDO_CAP	24	24	54	54	62	62	74	74
PA.5	25	25	55	55	63	63	75	75
PA.4	26	26	56	56	64	64	76	76
PA.3	27	27	57	57	65	65	77	77
PA.2	28	28	58	58	66	66	78	78
PA.1	29	29	59	59	67	67	79	79
PA.0	30	30	60	60	68	68	80	80
V_{DDIO}	31	31	61	61	69	69	81	81
PE.14			62	62	70	70	82	82
PE.15			63	63	71	71	83	83
nRESET	32	32	64	64	72	72	84	84
PF.0	33	33	65	65	73	73	85	85
PF.1	34	34	66	66	74	74	86	86
PD.9			67	67	75	75	87	87
PD.8			68	68	76	76	88	88
PC.5	35	35	69	69	77	77	89	89
PC.4	36	36	70	70	78	78	90	90
PC.3			71	71	79	79	91	91
PC.2			72	72	80	80	92	92
PC.1			73	73	81	81	93	93
PC.0			74	74	82	82	94	94
V_{SS}			75	75	83	83	95	95
V_{DD}			76	76	84	84	96	96
PG.9			77	77	85	85	97	97
PG.10			78	78	86	86	98	98
PG.11			79	79	87	87	99	99
PG.12			80	80	88	88	100	100
PG.13			81	81	89	89	101	101

PG.14			82	82	90	90	102	102
PG.15			83	83	91	91	103	103
HBI_PWR								
PJ.2							104	104
PJ.3							105	105
PJ.4							106	106
PJ.5							107	107
PJ.6							108	108
PJ.7							109	109
PH.12							110	110
PH.13							111	111
PH.14							112	112
PH.15							113	113
PD.7							114	114
PD.6							115	115
PD.5							116	116
HBI_PWR								
PD.4							117	117
PD.3					92	92	118	118
PD.2					93	93	119	119
PD.1					94	94	120	120
PD.0					95	95	121	121
PD.13			84	84	96	96	122	122
PA.12	37	37	85	85	97	97	123	123
PA.13	38	38	86	86	98	98	124	124
PA.14	39	39	87	87	99	99	125	125
PA.15	40	40	88	88	100	100	126	126
HSUSB_V_{RES}	41	41	89	89	101	101	127	127
HSUSB_VDD_{33}	42	42	90	90	102	102	128	128
HSUSB_VBUS	43	43	91	91	103	103	129	129
HSUSB_D-	44	44	92	92	104	104	130	130
HSUSB_VSS	45	45	93	93	105	105	131	131
HSUSB_D+	46	46	94	94	106	106	132	132
HSUSB_VDD_{12_CAP}	47	47	95	95	107	107	133	133

HSUSB_ID	48	48	96	96	108	108	134	134
PE.7			97	97	109	109	135	135
PE.6			98	98	110	110	136	136
PE.5			99	99	111	111	137	137
PE.4			100	100	112	112	138	138
PE.3			101	101	113	113	139	139
PE.2			102	102	114	114	140	140
V_{SS}	49	49	103	103	115	115	141	141
V_{DD}			104	104	116	116	142	142
PE.1			105	105	117	117	143	143
PE.0			106	106	118	118	144	144
PH.8			107	107	119	119	145	145
PH.9			108	108	120	120	146	146
PH.10			109	109	121	121	147	147
PH.11			110	110	122	122	148	148
PD.14			111	111	123	123	149	149
PJ.8							150	150
PJ.9							151	151
PJ.10							152	152
PJ.11							153	153
PJ.12							154	154
PJ.13							155	155
PG.5					124	124	156	156
PG.6					125	125	157	157
PG.7					126	126	158	158
PG.8					127	127	159	159
V_{SS}			112	112	128	128	160	160
LDO_CAP	50	50	113	113	129	129	161	161
V_{DD}	51	51	114	114	130	130	162	162
PC.14	52	52	115	115	131	131	163	163
PB.15	53	53	116	116	132	132	164	164
PB.14	54	54	117	117	133	133	165	165
PB.13	55	55	118	118	134	134	166	166
PB.12	56	56	119	119	135	135	167	167
AV_{DD}	57	57	120	120	136	136	168	168

V_{REF}	58	58	121	121	137	137	169	169
AV_{SS}	59	59	122	122	138	138	170	170
PB.11	60	60	123	123	139	139	171	171
PB.10	61	61	124	124	140	140	172	172
PB.9	62	62	125	125	141	141	173	173
PB.8	63	63	126	126	142	142	174	174
PB.7	64	64	127	127	143	143	175	175
PB.6	1	1	128	128	144	144	176	176
V_{SS}		65						
NC				129		145		177

4.3 Pin Functional Description

4.3.1 M463/M467 Series Summary Function Pin Description

Group	Pin Name	Type	Description
ACMP0	ACMP0_N	A	Analog comparator 0 negative input pin.
	ACMP0_O	O	Analog comparator 0 output pin.
	ACMP0_P0	A	Analog comparator 0 positive input 0 pin.
	ACMP0_P1	A	Analog comparator 0 positive input 1 pin.
	ACMP0_P2	A	Analog comparator 0 positive input 2 pin.
	ACMP0_P3	A	Analog comparator 0 positive input 3 pin.
	ACMP0_WLAT	I	Analog comparator 0 window latch input pin
ACMP1	ACMP1_N	A	Analog comparator 1 negative input pin.
	ACMP1_O	O	Analog comparator 1 output pin.
	ACMP1_P0	A	Analog comparator 1 positive input 0 pin.
	ACMP1_P1	A	Analog comparator 1 positive input 1 pin.
	ACMP1_P2	A	Analog comparator 1 positive input 2 pin.
	ACMP1_P3	A	Analog comparator 1 positive input 3 pin.
	ACMP1_WLAT	I	Analog comparator 1 window latch input pin
ACMP2	ACMP2_N	A	Analog comparator 2 negative input pin.
	ACMP2_O	O	Analog comparator 2 output pin.
	ACMP2_P0	A	Analog comparator 2 positive input 0 pin.
	ACMP2_P1	A	Analog comparator 2 positive input 1 pin.
	ACMP2_P2	A	Analog comparator 2 positive input 2 pin.
	ACMP2_P3	A	Analog comparator 2 positive input 3 pin.
	ACMP2_WLAT	I	Analog comparator 2 window latch input pin
ACMP3	ACMP3_N	A	Analog comparator 3 negative input pin.
	ACMP3_O	O	Analog comparator 3 output pin.
	ACMP3_P0	A	Analog comparator 3 positive input 0 pin.
	ACMP3_P1	A	Analog comparator 3 positive input 1 pin.
	ACMP3_P2	A	Analog comparator 3 positive input 2 pin.
	ACMP3_P3	A	Analog comparator 3 positive input 3 pin.
	ACMP3_WLAT	I	Analog comparator 3 window latch input pin
BPWM0	BPWM0_CH0	I/O	BPWM0 channel 0 output/capture input.
	BPWM0_CH1	I/O	BPWM0 channel 1 output/capture input.
	BPWM0_CH2	I/O	BPWM0 channel 2 output/capture input.
	BPWM0_CH3	I/O	BPWM0 channel 3 output/capture input.

Group	Pin Name	Type	Description
	BPWM0_CH4	I/O	BPWM0 channel 4 output/capture input.
	BPWM0_CH5	I/O	BPWM0 channel 5 output/capture input.
BPWM1	BPWM1_CH0	I/O	BPWM1 channel 0 output/capture input.
	BPWM1_CH1	I/O	BPWM1 channel 1 output/capture input.
	BPWM1_CH2	I/O	BPWM1 channel 2 output/capture input.
	BPWM1_CH3	I/O	BPWM1 channel 3 output/capture input.
	BPWM1_CH4	I/O	BPWM1 channel 4 output/capture input.
	BPWM1_CH5	I/O	BPWM1 channel 5 output/capture input.
CAN0	CAN0_RXD	I	CAN0 bus receiver input.
	CAN0_TXD	O	CAN0 bus transmitter output.
CAN1	CAN1_RXD	I	CAN1 bus receiver input.
	CAN1_TXD	O	CAN1 bus transmitter output.
CAN2	CAN2_RXD	I	CAN2 bus receiver input.
	CAN2_TXD	O	CAN2 bus transmitter output.
CAN3	CAN3_RXD	I	CAN3 bus receiver input.
	CAN3_TXD	O	CAN3 bus transmitter output.
CCAP	CCAP_DATA0	I	Camera capture data input bus bit 0.
	CCAP_DATA1	I	Camera capture data input bus bit 1.
	CCAP_DATA2	I	Camera capture data input bus bit 2.
	CCAP_DATA3	I	Camera capture data input bus bit 3.
	CCAP_DATA4	I	Camera capture data input bus bit 4.
	CCAP_DATA5	I	Camera capture data input bus bit 5.
	CCAP_DATA6	I	Camera capture data input bus bit 6.
	CCAP_DATA7	I	Camera capture data input bus bit 7.
	CCAP_HSYNC	I	Camera capture interface hsync input pin.
	CCAP_PIXCLK	I	Camera capture interface pix clock input pin.
	CCAP_SCLK	O	Camera capture interface sensor clock pin.
	CCAP_SFIELD	I	Camera capture interface SFIELD input pin.
	CCAP_VSYNC	I	Camera capture interface vsync input pin.
CLKO	CLKO	O	Clock Out
DAC0	DAC0_OUT	A	DAC0 channel analog output.
	DAC0_ST	I	DAC0 external trigger input.
DAC1	DAC1_OUT	A	DAC1 channel analog output.
	DAC1_ST	I	DAC1 external trigger input.

Group	Pin Name	Type	Description
EADC0	EADC0_CH0	A	EADC0 channel 0 analog input.
	EADC0_CH1	A	EADC0 channel 1 analog input.
	EADC0_CH2	A	EADC0 channel 2 analog input.
	EADC0_CH3	A	EADC0 channel 3 analog input.
	EADC0_CH4	A	EADC0 channel 4 analog input.
	EADC0_CH5	A	EADC0 channel 5 analog input.
	EADC0_CH6	A	EADC0 channel 6 analog input.
	EADC0_CH7	A	EADC0 channel 7 analog input.
	EADC0_CH8	A	EADC0 channel 8 analog input.
	EADC0_CH9	A	EADC0 channel 9 analog input.
	EADC0_CH10	A	EADC0 channel 10 analog input.
	EADC0_CH11	A	EADC0 channel 11 analog input.
	EADC0_CH12	A	EADC0 channel 12 analog input.
	EADC0_CH13	A	EADC0 channel 13 analog input.
	EADC0_CH14	A	EADC0 channel 14 analog input.
	EADC0_CH15	A	EADC0 channel 15 analog input.
	EADC0_ST	I	EADC0 external trigger input.
EADC1	EADC1_CH0	A	EADC1 channel 0 analog input.
	EADC1_CH1	A	EADC1 channel 1 analog input.
	EADC1_CH2	A	EADC1 channel 2 analog input.
	EADC1_CH3	A	EADC1 channel 3 analog input.
	EADC1_CH4	A	EADC1 channel 4 analog input.
	EADC1_CH5	A	EADC1 channel 5 analog input.
	EADC1_CH6	A	EADC1 channel 6 analog input.
	EADC1_CH7	A	EADC1 channel 7 analog input.
	EADC1_CH8	A	EADC1 channel 8 analog input.
	EADC1_CH9	A	EADC1 channel 9 analog input.
	EADC1_CH10	A	EADC1 channel 10 analog input.
	EADC1_CH11	A	EADC1 channel 11 analog input.
	EADC1_CH12	A	EADC1 channel 12 analog input.
	EADC1_CH13	A	EADC1 channel 13 analog input.
	EADC1_CH14	A	EADC1 channel 14 analog input.
	EADC1_CH15	A	EADC1 channel 15 analog input.
	EADC1_ST	I	EADC1 external trigger input.

Group	Pin Name	Type	Description
EADC2	EADC2_CH0	A	EADC2 channel 0 analog input.
	EADC2_CH1	A	EADC2 channel 1 analog input.
	EADC2_CH2	A	EADC2 channel 2 analog input.
	EADC2_CH3	A	EADC2 channel 3 analog input.
	EADC2_CH4	A	EADC2 channel 4 analog input.
	EADC2_CH5	A	EADC2 channel 5 analog input.
	EADC2_CH6	A	EADC2 channel 6 analog input.
	EADC2_CH7	A	EADC2 channel 7 analog input.
	EADC2_CH8	A	EADC2 channel 8 analog input.
	EADC2_CH9	A	EADC2 channel 9 analog input.
	EADC2_CH10	A	EADC2 channel 10 analog input.
	EADC2_CH11	A	EADC2 channel 11 analog input.
	EADC2_CH12	A	EADC2 channel 12 analog input.
	EADC2_CH13	A	EADC2 channel 13 analog input.
	EADC2_CH14	A	EADC2 channel 14 analog input.
	EADC2_CH15	A	EADC2 channel 15 analog input.
	EADC2_ST	I	EADC2 external trigger input.
EBI	EBI_AD0	I/O	EBI address/data bus bit 0.
	EBI_AD1	I/O	EBI address/data bus bit 1.
	EBI_AD2	I/O	EBI address/data bus bit 2.
	EBI_AD3	I/O	EBI address/data bus bit 3.
	EBI_AD4	I/O	EBI address/data bus bit 4.
	EBI_AD5	I/O	EBI address/data bus bit 5.
	EBI_AD6	I/O	EBI address/data bus bit 6.
	EBI_AD7	I/O	EBI address/data bus bit 7.
	EBI_AD8	I/O	EBI address/data bus bit 8.
	EBI_AD9	I/O	EBI address/data bus bit 9.
	EBI_AD10	I/O	EBI address/data bus bit 10.
	EBI_AD11	I/O	EBI address/data bus bit 11.
	EBI_AD12	I/O	EBI address/data bus bit 12.
	EBI_AD13	I/O	EBI address/data bus bit 13.
	EBI_AD14	I/O	EBI address/data bus bit 14.
	EBI_AD15	I/O	EBI address/data bus bit 15.
	EBI_ADR0	O	EBI address bus bit 0.

Group	Pin Name	Type	Description
	EBI_ADR1	O	EBI address bus bit 1.
	EBI_ADR2	O	EBI address bus bit 2.
	EBI_ADR3	O	EBI address bus bit 3.
	EBI_ADR4	O	EBI address bus bit 4.
	EBI_ADR5	O	EBI address bus bit 5.
	EBI_ADR6	O	EBI address bus bit 6.
	EBI_ADR7	O	EBI address bus bit 7.
	EBI_ADR8	O	EBI address bus bit 8.
	EBI_ADR9	O	EBI address bus bit 9.
	EBI_ADR10	O	EBI address bus bit 10.
	EBI_ADR11	O	EBI address bus bit 11.
	EBI_ADR12	O	EBI address bus bit 12.
	EBI_ADR13	O	EBI address bus bit 13.
	EBI_ADR14	O	EBI address bus bit 14.
	EBI_ADR15	O	EBI address bus bit 15.
	EBI_ADR16	O	EBI address bus bit 16.
	EBI_ADR17	O	EBI address bus bit 17.
	EBI_ADR18	O	EBI address bus bit 18.
	EBI_ADR19	O	EBI address bus bit 19.
	EBI_ALE	O	EBI address latch enable output pin.
	EBI_MCLK	O	EBI external clock output pin.
	EBI_nCS0	O	EBI chip select 0 output pin.
	EBI_nCS1	O	EBI chip select 1 output pin.
	EBI_nCS2	O	EBI chip select 2 output pin.
	EBI_nRD	O	EBI read enable output pin.
	EBI_nWR	O	EBI write enable output pin.
	EBI_nWRH	O	EBI high byte write enable output pin
	EBI_nWRL	O	EBI low byte write enable output pin.
ECAP0	ECAP0_IC0	I	Enhanced capture unit 0 input 0 pin.
	ECAP0_IC1	I	Enhanced capture unit 0 input 1 pin.
	ECAP0_IC2	I	Enhanced capture unit 0 input 2 pin.
ECAP1	ECAP1_IC0	I	Enhanced capture unit 1 input 0 pin.
	ECAP1_IC1	I	Enhanced capture unit 1 input 1 pin.
	ECAP1_IC2	I	Enhanced capture unit 1 input 2 pin.

Group	Pin Name	Type	Description
ECAP2	ECAP2_IC0	I	Enhanced capture unit 2 input 0 pin.
	ECAP2_IC1	I	Enhanced capture unit 2 input 1 pin.
	ECAP2_IC2	I	Enhanced capture unit 2 input 2 pin.
ECAP3	ECAP3_IC0	I	Enhanced capture unit 3 input 0 pin.
	ECAP3_IC1	I	Enhanced capture unit 3 input 1 pin.
	ECAP3_IC2	I	Enhanced capture unit 3 input 2 pin.
EMAC0	EMAC0_PPS	O	EMAC0 Pulse Per Second output pin.
	EMAC0_RMII_CRSDV	I	EMAC0 RMII Carrier Sense/Receive Data input pin.
	EMAC0_RMII_MDC	O	EMAC0 RMII PHY Management Clock output pin.
	EMAC0_RMII_MDIO	I/O	EMAC0 RMII PHY Management Data pin.
	EMAC0_RMII_REFCLK	I	EMAC0 RMII reference clock input pin.
	EMAC0_RMII_RXD0	I	EMAC0 RMII Receive Data bus bit 0.
	EMAC0_RMII_RXD1	I	EMAC0 RMII Receive Data bus bit 1.
	EMAC0_RMII_RXERR	I	EMAC0 RMII Receive Data Error input pin.
	EMAC0_RMII_TXD0	O	EMAC0 RMII Transmit Data bus bit 0.
	EMAC0_RMII_TXD1	O	EMAC0 RMII Transmit Data bus bit 1.
	EMAC0_RMII_TXEN	O	EMAC0 RMII Transmit Enable output pin.
EPWM0	EPWM0_BRAKE0	I	EPWM0 Brake 0 input pin.
	EPWM0_BRAKE1	I	EPWM0 Brake 1 input pin.
	EPWM0_CH0	I/O	EPWM0 channel 0 output/capture input.
	EPWM0_CH1	I/O	EPWM0 channel 1 output/capture input.
	EPWM0_CH2	I/O	EPWM0 channel 2 output/capture input.
	EPWM0_CH3	I/O	EPWM0 channel 3 output/capture input.
	EPWM0_CH4	I/O	EPWM0 channel 4 output/capture input.
	EPWM0_CH5	I/O	EPWM0 channel 5 output/capture input.
	EPWM0_SYNC_IN	I	EPWM0 counter synchronous trigger input pin.
	EPWM0_SYNC_OUT	O	EPWM0 counter synchronous trigger output pin.
EPWM1	EPWM1_BRAKE0	I	EPWM1 Brake 0 input pin.
	EPWM1_BRAKE1	I	EPWM1 Brake 1 input pin.
	EPWM1_CH0	I/O	EPWM1 channel 0 output/capture input.
	EPWM1_CH1	I/O	EPWM1 channel 1 output/capture input.
	EPWM1_CH2	I/O	EPWM1 channel 2 output/capture input.
	EPWM1_CH3	I/O	EPWM1 channel 3 output/capture input.
	EPWM1_CH4	I/O	EPWM1 channel 4 output/capture input.

Group	Pin Name	Type	Description
	EPWM1_CH5	I/O	EPWM1 channel 5 output/capture input.
EQEI0	EQEI0_A	I	EQEI0 phase A input
	EQEI0_B	I	EQEI0 phase B input
	EQEI0_INDEX	I	EQEI0 index input
EQEI1	EQEI1_A	I	EQEI1 phase A input
	EQEI1_B	I	EQEI1 phase B input
	EQEI1_INDEX	I	EQEI1 index input
EQEI2	EQEI2_A	I	EQEI2 phase A input
	EQEI2_B	I	EQEI2 phase B input
	EQEI2_INDEX	I	EQEI2 index input
EQEI3	EQEI3_A	I	EQEI3 phase A input
	EQEI3_B	I	EQEI3 phase B input
	EQEI3_INDEX	I	EQEI3 index input
ETMC	ETMC_TRACE_CLK	I	ETM receiver Trace Clock input pin
	ETMC_TRACE_DATA0	I	ETM receiver Trace Data 0 input pin
	ETMC_TRACE_DATA1	I	ETM receiver Trace Data 1 input pin
	ETMC_TRACE_DATA2	I	ETM receiver Trace Data 2 input pin
	ETMC_TRACE_DATA3	I	ETM receiver Trace Data 3 input pin
HBI	HBI_CK	O	HyperBus clock pin.
	HBI_D0	I/O	HyperBus data 0 pin.
	HBI_D1	I/O	HyperBus data 1 pin.
	HBI_D2	I/O	HyperBus data 2 pin.
	HBI_D3	I/O	HyperBus data 3 pin.
	HBI_D4	I/O	HyperBus data 4 pin.
	HBI_D5	I/O	HyperBus data 5 pin.
	HBI_D6	I/O	HyperBus data 6 pin.
	HBI_D7	I/O	HyperBus data 7 pin.
	HBI_PWR		
	HBI_RWDS	I/O	HyperBus read-write data strobe pin.
	HBI_nCK	O	HyperBus reverse clock pin.
	HBI_nCS	O	HyperBus chip select pin.
	HBI_nRESET	O	HyperBus reset pin.

Group	Pin Name	Type	Description
HSUSB	HSUSB_VBUS_EN	O	HSUSB external V_{BUS} regulator enable pin.
	HSUSB_VBUS_ST	I	HSUSB external V_{BUS} regulator status pin.
I2C0	I2C0_SCL	I/O	I2C0 clock pin.
	I2C0_SDA	I/O	I2C0 data input/output pin.
	I2C0_SMBAL	O	I2C0 SMBus SMBALTER pin
	I2C0_SMBSUS	O	I2C0 SMBus SMBSUS pin (PMBus CONTROL pin)
I2C1	I2C1_SCL	I/O	I2C1 clock pin.
	I2C1_SDA	I/O	I2C1 data input/output pin.
	I2C1_SMBAL	O	I2C1 SMBus SMBALTER pin
	I2C1_SMBSUS	O	I2C1 SMBus SMBSUS pin (PMBus CONTROL pin)
I2C2	I2C2_SCL	I/O	I2C2 clock pin.
	I2C2_SDA	I/O	I2C2 data input/output pin.
	I2C2_SMBAL	O	I2C2 SMBus SMBALTER pin
	I2C2_SMBSUS	O	I2C2 SMBus SMBSUS pin (PMBus CONTROL pin)
I2C3	I2C3_SCL	I/O	I2C3 clock pin.
	I2C3_SDA	I/O	I2C3 data input/output pin.
	I2C3_SMBAL	O	I2C3 SMBus SMBALTER pin
	I2C3_SMBSUS	O	I2C3 SMBus SMBSUS pin (PMBus CONTROL pin)
I2C4	I2C4_SCL	I/O	I2C4 clock pin.
	I2C4_SDA	I/O	I2C4 data input/output pin.
	I2C4_SMBAL	O	I2C4 SMBus SMBALTER pin
	I2C4_SMBSUS	O	I2C4 SMBus SMBSUS pin (PMBus CONTROL pin)
I2S0	I2S0_BCLK	O	I2S0 bit clock output pin.
	I2S0_DI	I	I2S0 data input pin.
	I2S0_DO	O	I2S0 data output pin.
	I2S0_LRCK	O	I2S0 left right channel clock output pin.
	I2S0_MCLK	O	I2S0 master clock output pin.
I2S1	I2S1_BCLK	O	I2S1 bit clock output pin.
	I2S1_DI	I	I2S1 data input pin.
	I2S1_DO	O	I2S1 data output pin.
	I2S1_LRCK	O	I2S1 left right channel clock output pin.
	I2S1_MCLK	O	I2S1 master clock output pin.
ICE	ICE_CLK	I	Serial wired debugger clock pin. **Note:** It is recommended to use 100 kΩ pull-up resistor on ICE_CLK pin.
	ICE_DAT	I/O	Serial wired debugger data pin.

Group	Pin Name	Type	Description
			Note: It is recommended to use 100 kΩ pull-up resistor on ICE_DAT pin.
INT0	INT0	I	External interrupt 0 input pin.
INT1	INT1	I	External interrupt 1 input pin.
INT2	INT2	I	External interrupt 2 input pin.
INT3	INT3	I	External interrupt 3 input pin.
INT4	INT4	I	External interrupt 4 input pin.
INT5	INT5	I	External interrupt 5 input pin.
INT6	INT6	I	External interrupt 6 input pin.
INT7	INT7	I	External interrupt 7 input pin.
KPI	KPI_COL0	I	Keypad Interface Column 0 input pin.
	KPI_COL1	I	Keypad Interface Column 1 input pin.
	KPI_COL2	I	Keypad Interface Column 2 input pin.
	KPI_COL3	I	Keypad Interface Column 3 input pin.
	KPI_COL4	I	Keypad Interface Column 4 input pin.
	KPI_COL5	I	Keypad Interface Column 5 input pin.
	KPI_COL6	I	Keypad Interface Column 6 input pin.
	KPI_COL7	I	Keypad Interface Column 7 input pin.
	KPI_ROW0	O	Keypad Interface Row 0 output pin.
	KPI_ROW1	O	Keypad Interface Row 1 output pin.
	KPI_ROW2	O	Keypad Interface Row 2 output pin.
	KPI_ROW3	O	Keypad Interface Row 3 output pin.
	KPI_ROW4	O	Keypad Interface Row 4 output pin.
	KPI_ROW5	O	Keypad Interface Row 5 output pin.
PSIO0	PSIO0_CH0	I/O	PSIO 0 channel 0 input/output pin.
	PSIO0_CH1	I/O	PSIO 0 channel 1 input/output pin.
	PSIO0_CH2	I/O	PSIO 0 channel 2 input/output pin.
	PSIO0_CH3	I/O	PSIO 0 channel 3 input/output pin.
	PSIO0_CH4	I/O	PSIO 0 channel 4 input/output pin.
	PSIO0_CH5	I/O	PSIO 0 channel 5 input/output pin.
	PSIO0_CH6	I/O	PSIO 0 channel 6 input/output pin.
	PSIO0_CH7	I/O	PSIO 0 channel 7 input/output pin.
QSPI0	QSPI0_CLK	I/O	Quad SPI0 serial clock pin.
	QSPI0_MISO0	I/O	Quad SPI0 MISO0 (Master In, Slave Out) pin.
	QSPI0_MISO1	I/O	Quad SPI0 MISO1 (Master In, Slave Out) pin.
	QSPI0_MOSI0	I/O	Quad SPI0 MOSI0 (Master Out, Slave In) pin.

Group	Pin Name	Type	Description
	QSPI0_MOSI1	I/O	Quad SPI0 MOSI1 (Master Out, Slave In) pin.
	QSPI0_SS	I/O	Quad SPI0 slave select pin.
QSPI1	QSPI1_CLK	I/O	Quad SPI1 serial clock pin.
	QSPI1_MISO0	I/O	Quad SPI1 MISO0 (Master In, Slave Out) pin.
	QSPI1_MISO1	I/O	Quad SPI1 MISO1 (Master In, Slave Out) pin.
	QSPI1_MOSI0	I/O	Quad SPI1 MOSI0 (Master Out, Slave In) pin.
	QSPI1_MOSI1	I/O	Quad SPI1 MOSI1 (Master Out, Slave In) pin.
	QSPI1_SS	I/O	Quad SPI1 slave select pin.
SC0	SC0_CLK	O	Smart Card 0 clock pin.
	SC0_DAT	I/O	Smart Card 0 data pin.
	SC0_PWR	O	Smart Card 0 power pin.
	SC0_RST	O	Smart Card 0 reset pin.
	SC0_nCD	I	Smart Card 0 card detect pin.
SC1	SC1_CLK	O	Smart Card 1 clock pin.
	SC1_DAT	I/O	Smart Card 1 data pin.
	SC1_PWR	O	Smart Card 1 power pin.
	SC1_RST	O	Smart Card 1 reset pin.
	SC1_nCD	I	Smart Card 1 card detect pin.
SC2	SC2_CLK	O	Smart Card 2 clock pin.
	SC2_DAT	I/O	Smart Card 2 data pin.
	SC2_PWR	O	Smart Card 2 power pin.
	SC2_RST	O	Smart Card 2 reset pin.
	SC2_nCD	I	Smart Card 2 card detect pin.
SD0	SD0_CLK	O	SD/SDIO0 clock output pin
	SD0_CMD	I/O	SD/SDIO0 command/response pin
	SD0_DAT0	I/O	SD/SDIO0 data line bit 0.
	SD0_DAT1	I/O	SD/SDIO0 data line bit 1.
	SD0_DAT2	I/O	SD/SDIO0 data line bit 2.
	SD0_DAT3	I/O	SD/SDIO0 data line bit 3.
	SD0_nCD	I	SD/SDIO0 card detect input pin
SD1	SD1_CLK	O	SD/SDIO1 clock output pin
	SD1_CMD	I/O	SD/SDIO1 command/response pin
	SD1_DAT0	I/O	SD/SDIO1 data line bit 0.
	SD1_DAT1	I/O	SD/SDIO1 data line bit 1.

Group	Pin Name	Type	Description
	SD1_DAT2	I/O	SD/SDIO1 data line bit 2.
	SD1_DAT3	I/O	SD/SDIO1 data line bit 3.
	SD1_nCD	I	SD/SDIO1 card detect input pin
SPI0	SPI0_CLK	I/O	SPI0 serial clock pin.
	SPI0_I2SMCLK	I/O	SPI0 I^2S master clock output pin
	SPI0_MISO	I/O	SPI0 MISO (Master In, Slave Out) pin.
	SPI0_MOSI	I/O	SPI0 MOSI (Master Out, Slave In) pin.
	SPI0_SS	I/O	SPI0 slave select pin.
SPI1	SPI1_CLK	I/O	SPI1 serial clock pin.
	SPI1_I2SMCLK	I/O	SPI1 I^2S master clock output pin
	SPI1_MISO	I/O	SPI1 MISO (Master In, Slave Out) pin.
	SPI1_MOSI	I/O	SPI1 MOSI (Master Out, Slave In) pin.
	SPI1_SS	I/O	SPI1 slave select pin.
SPI2	SPI2_CLK	I/O	SPI2 serial clock pin.
	SPI2_I2SMCLK	I/O	SPI2 I^2S master clock output pin
	SPI2_MISO	I/O	SPI2 MISO (Master In, Slave Out) pin.
	SPI2_MOSI	I/O	SPI2 MOSI (Master Out, Slave In) pin.
	SPI2_SS	I/O	SPI2 slave select pin.
SPI3	SPI3_CLK	I/O	SPI3 serial clock pin.
	SPI3_I2SMCLK	I/O	SPI3 I^2S master clock output pin
	SPI3_MISO	I/O	SPI3 MISO (Master In, Slave Out) pin.
	SPI3_MOSI	I/O	SPI3 MOSI (Master Out, Slave In) pin.
	SPI3_SS	I/O	SPI3 slave select pin.
SPI4	SPI4_CLK	I/O	SPI4 serial clock pin.
	SPI4_MISO	I/O	SPI4 MISO (Master In, Slave Out) pin.
	SPI4_MOSI	I/O	SPI4 MOSI (Master Out, Slave In) pin.
	SPI4_SS	I/O	SPI4 slave select pin.
SPI5	SPI5_CLK	I/O	SPI5 serial clock pin.
	SPI5_MISO	I/O	SPI5 MISO (Master In, Slave Out) pin.
	SPI5_MOSI	I/O	SPI5 MOSI (Master Out, Slave In) pin.
	SPI5_SS	I/O	SPI5 slave select pin.
SPI6	SPI6_CLK	I/O	SPI6 serial clock pin.
	SPI6_MISO	I/O	SPI6 MISO (Master In, Slave Out) pin.
	SPI6_MOSI	I/O	SPI6 MOSI (Master Out, Slave In) pin.

Group	Pin Name	Type	Description
	SPI6_SS	I/O	SPI6 slave select pin.
SPI7	SPI7_CLK	I/O	SPI7 serial clock pin.
	SPI7_MISO	I/O	SPI7 MISO (Master In, Slave Out) pin.
	SPI7_MOSI	I/O	SPI7 MOSI (Master Out, Slave In) pin.
	SPI7_SS	I/O	SPI7 slave select pin.
SPI8	SPI8_CLK	I/O	SPI8 serial clock pin.
	SPI8_MISO	I/O	SPI8 MISO (Master In, Slave Out) pin.
	SPI8_MOSI	I/O	SPI8 MOSI (Master Out, Slave In) pin.
	SPI8_SS	I/O	SPI8 slave select pin.
SPI9	SPI9_CLK	I/O	SPI9 serial clock pin.
	SPI9_MISO	I/O	SPI9 MISO (Master In, Slave Out) pin.
	SPI9_MOSI	I/O	SPI9 MOSI (Master Out, Slave In) pin.
	SPI9_SS	I/O	SPI9 slave select pin.
SPI10	SPI10_CLK	I/O	SPI10 serial clock pin.
	SPI10_MISO	I/O	SPI10 MISOO (Master In, Slave Out) pin.
	SPI10_MOSI	I/O	SPI10 MOSII (Master Out, Slave In) pin.
	SPI10_SS	I/O	SPI10 slave select S pin.
SPIM	SPIM_CLK	I/O	SPIM serial clock pin.
	SPIM_D2	I/O	SPIM data 2 pin for Quad Mode I/O.
	SPIM_D3	I/O	SPIM data 3 pin for Quad Mode I/O.
	SPIM_MISO	I/O	SPIM MISO (Master In, Slave Out) pin.
	SPIM_MOSI	I/O	SPIM MOSI (Master Out, Slave In) pin.
	SPIM_SS	I/O	SPIM slave select pin.
SWDH	SWDH_CLK	O	Serial Wire Debug Host Clock output
	SWDH_DAT	I/O	Serial Wire Debug Host Data input/output pin
TAMPER0	TAMPER0	I/O	TAMPER detector loop pin 0.
TAMPER1	TAMPER1	I/O	TAMPER detector loop pin 1.
TAMPER2	TAMPER2	I/O	TAMPER detector loop pin 2.
TAMPER3	TAMPER3	I/O	TAMPER detector loop pin 3.
TAMPER4	TAMPER4	I/O	TAMPER detector loop pin 4.
TAMPER5	TAMPER5	I/O	TAMPER detector loop pin 5.
TM0	TM0	I/O	Timer0 event counter input/toggle output pin.
	TM0_EXT	I/O	Timer0 external capture input/toggle output pin.
TM1	TM1	I/O	Timer1 event counter input/toggle output pin.

Group	Pin Name	Type	Description
	TM1_EXT	I/O	Timer1 external capture input/toggle output pin.
TM2	TM2	I/O	Timer2 event counter input/toggle output pin.
	TM2_EXT	I/O	Timer2 external capture input/toggle output pin.
TM3	TM3	I/O	Timer3 event counter input/toggle output pin.
	TM3_EXT	I/O	Timer3 external capture input/toggle output pin.
TRACE	TRACE_CLK	O	ETM Trace Clock output pin
	TRACE_DATA0	O	ETM Trace Data 0 output pin
	TRACE_DATA1	O	ETM Trace Data 1 output pin
	TRACE_DATA2	O	ETM Trace Data 2 output pin
	TRACE_DATA3	O	ETM Trace Data 3 output pin
	TRACE_SWO	O	Trace Single Wire output pin.
UART0	UART0_RXD	I	UART0 data receiver input pin.
	UART0_TXD	O	UART0 data transmitter output pin.
	UART0_nCTS	I	UART0 clear to Send input pin.
	UART0_nRTS	O	UART0 request to Send output pin.
UART1	UART1_RXD	I	UART1 data receiver input pin.
	UART1_TXD	O	UART1 data transmitter output pin.
	UART1_nCTS	I	UART1 clear to Send input pin.
	UART1_nRTS	O	UART1 request to Send output pin.
UART2	UART2_RXD	I	UART2 data receiver input pin.
	UART2_TXD	O	UART2 data transmitter output pin.
	UART2_nCTS	I	UART2 clear to Send input pin.
	UART2_nRTS	O	UART2 request to Send output pin.
UART3	UART3_RXD	I	UART3 data receiver input pin.
	UART3_TXD	O	UART3 data transmitter output pin.
	UART3_nCTS	I	UART3 clear to Send input pin.
	UART3_nRTS	O	UART3 request to Send output pin.
UART4	UART4_RXD	I	UART4 data receiver input pin.
	UART4_TXD	O	UART4 data transmitter output pin.
	UART4_nCTS	I	UART4 clear to Send input pin.
	UART4_nRTS	O	UART4 request to Send output pin.
UART5	UART5_RXD	I	UART5 data receiver input pin.
	UART5_TXD	O	UART5 data transmitter output pin.
	UART5_nCTS	I	UART5 clear to Send input pin.

Group	Pin Name	Type	Description
	UART5_nRTS	O	UART5 request to Send output pin.
UART6	UART6_RXD	I	UART6 data receiver input pin.
	UART6_TXD	O	UART6 data transmitter output pin.
	UART6_nCTS	I	UART6 clear to Send input pin.
	UART6_nRTS	O	UART6 request to Send output pin.
UART7	UART7_RXD	I	UART7 data receiver input pin.
	UART7_TXD	O	UART7 data transmitter output pin.
	UART7_nCTS	I	UART7 clear to Send input pin.
	UART7_nRTS	O	UART7 request to Send output pin.
UART8	UART8_RXD	I	UART8 data receiver input pin.
	UART8_TXD	O	UART8 data transmitter output pin.
	UART8_nCTS	I	UART8 clear to Send input pin.
	UART8_nRTS	O	UART8 request to Send output pin.
UART9	UART9_RXD	I	UART9 data receiver input pin.
	UART9_TXD	O	UART9 data transmitter output pin.
	UART9_nCTS	I	UART9 clear to Send input pin.
	UART9_nRTS	O	UART9 request to Send output pin.
USB	USB_D+	A	USB differential signal D+.
	USB_D-	A	USB differential signal D-.
	USB_OTG_ID	I	USB_ identification.
	USB_VBUS	I/O	Power supply from USB host or HUB.
	USB_VBUS_EN	O	USB external V_{BUS} regulator enable pin.
	USB_VBUS_ST	I	USB external V_{BUS} regulator status pin.
	HSUSB_D+	A	HSUSB differential signal D+.
	HSUSB_D-	A	HSUSB differential signal D-.
	HSUSB_ID	I	HSUSB identification.
	HSUSB_VBUS	I/O	HSUSB Power supply from USB host or HUB.
	HSUSB_VDD12_CAP	A	HSUSB Internal power regulator output 1.2V decoupling pin. **Note:** This pin needs to be connected with a 1uF capacitor.
	HSUSB_VDD33	P	Power supply for HSUSB VDD33
	HSUSB_VRES	A	HSUSB module reference resister
	HSUSB_VSS	P	Ground pin for HSUSB.
USCI0	USCI0_CLK	I/O	USCI0 clock pin.
	USCI0_CTL0	I/O	USCI0 control 0 pin.
	USCI0_CTL1	I/O	USCI0 control 1 pin.

Group	Pin Name	Type	Description
	USCI0_DAT0	I/O	USCI0 data 0 pin.
	USCI0_DAT1	I/O	USCI0 data 1 pin.
X32	X32_IN	I	External 32.768 kHz crystal input pin.
	X32_OUT	O	External 32.768 kHz crystal output pin.
XT1	XT1_IN	I	External 4~24 MHz (high speed) crystal input pin.
	XT1_OUT	O	External 4~24 MHz (high speed) crystal output pin.
Power	AV_{DD}	P	Power supply for internal analog circuit.
	AV_{SS}	P	Ground pin for analog circuit.
	HBI_PWR	P	Power supply for internal HyperRAM.
	LDO_CAP	A	LDO output pin. **Note:** This pin needs to be connected with a capacitor whose value can be found in General operating conditions table in Datasheet.
	V_{BAT}	P	Power supply by batteries for RTC.
	V_{DD}	P	Power supply for I/O ports and LDO source for internal PLL and digital circuit.
	V_{DDIO}	P	Power supply for PA.0~PA.5.
	V_{REF}	A	ADC reference voltage input. **Note:** This pin needs to be connected with a 1uF capacitor.
	V_{SS}	P	Ground pin for digital circuit.
	nRESET	I	External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. **Note:** It is recommended to use 10 kΩ pull-up resistor and 10 uF capacitor on nRESET pin.

5 BLOCK DIAGRAM

5.1 M463/M467 Series Block Diagram

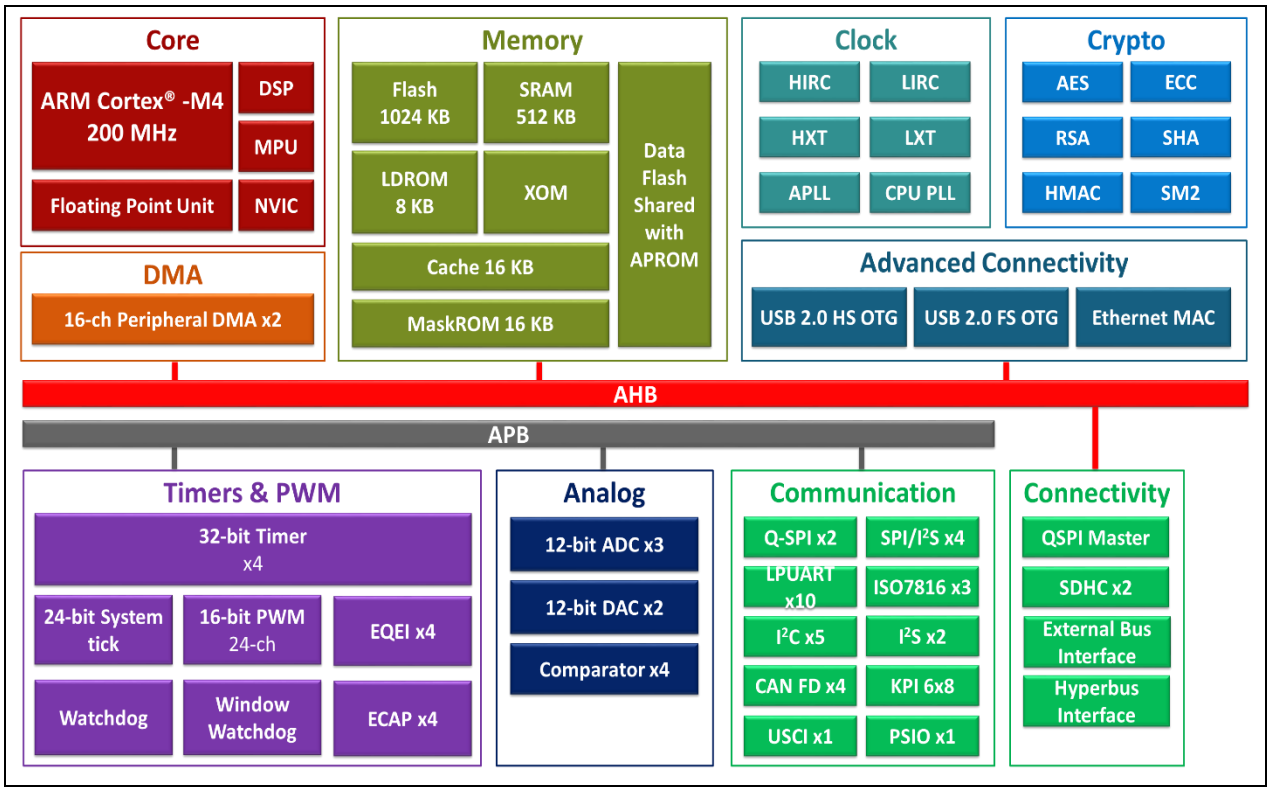


Figure 5.1-1 M463/M467 Series Block Diagram

6 FUNCTIONAL DESCRIPTION

6.1 ARM Cortex-M4 Core

The Cortex-M4 processor, a configurable, multistage, 32-bit RISC processor, has three AMBA AHB-Lite interfaces for best parallel performance and includes an NVIC component. The processor with optional hardware debug functionality can execute Thumb code and is compatible with other Cortex-M profile processors. The profile supports two modes -Thread mode and Handler mode. Handler mode is entered as a result of an exception. An exception return can only be issued in Handler mode. Thread mode is entered on Reset, and can be entered as a result of an exception return. The Cortex-M4F is a processor with the same capability as the Cortex-M4 processor and includes floating point arithmetic functionality. The NuMicro M463/M467 series is embedded with Cortex-M4F processor. Throughout this document, the name Cortex-M4 refers to both Cortex-M4 and Cortex-M4F processors. Figure 6.1-1 shows the functional controller of the processor.

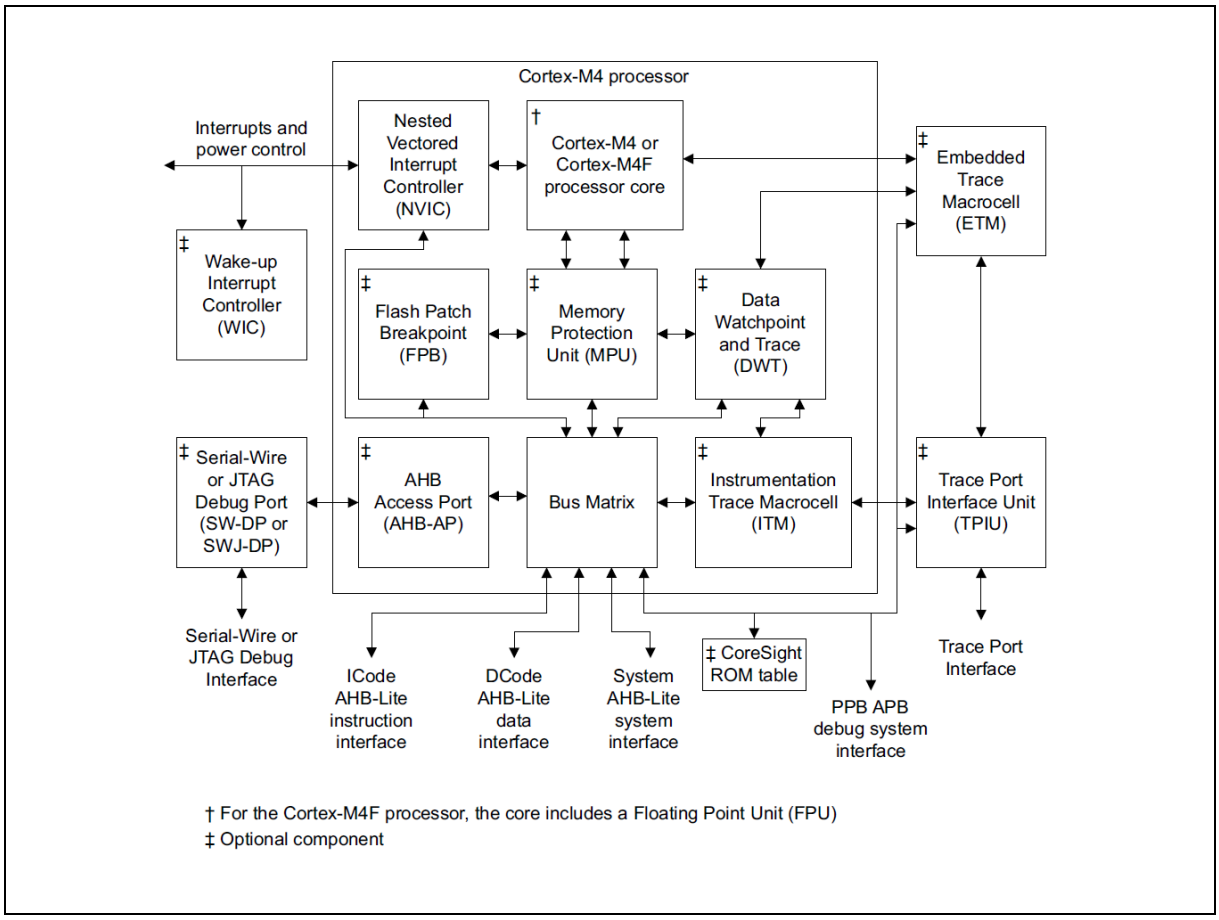


Figure 6.1-1 Cortex-M4 Block Diagram

Cortex-M4 processor features:

- A low gate count processor core, with low latency interrupt processing that has:
  - A subset of the Thumb instruction set, defined in the *ARMv7-M Architecture Reference Manual*
  - Banked Stack Pointer (SP)
  - Hardware integer divide instructions, SDIV and UDIV
  - Handler and Thread modes

- Thumb and Debug states
- Support for interruptible-continued instructions LDM, STM, PUSH, and POP for low interrupt latency
- Automatic processor state saving and restoration for low latency *Interrupt Service Routine (ISR)* entry and exit
- Support for ARMv6 big-endian byte-invariant or little-endian accesses
- Support for ARMv6 unaligned accesses

- Floating Point Unit (FPU) in the Cortex-M4F processor providing:
  - 32-bit instructions for single-precision (C float) data-processing operations
  - Combined Multiply and Accumulate instructions for increased precision (Fused MAC)
  - Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root
  - Hardware support for denormals and all IEEE rounding modes
  - 32 dedicated 32-bit single precision registers, also addressable as 16 double-word registers
  - Decoupled three stage pipeline
- Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing. Features include:
  - External interrupts. Configurable from 1 to 240
  - Bits of priority, configurable from 3 to 8
  - Dynamic reprioritization of interrupts
  - Priority grouping which enables selection of preempting interrupt levels and nonpreempting interrupt levels
  - Support for tril-chaining and late arrival of interrupts, which enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
  - Processor state automatically saved on interrupt entry, and restored on interrupt exit with on instruction overhead
  - Support for Wake-up Interrupt Controller (WIC) with Ultra-low Power Sleep mode
- Memory Protection Unit (MPU). An optional MPU for memory protection, including:
  - Eight memory regions
  - Sub Region Disable (SRD), enabling efficient use of memory regions
  - The ability to enable a background region that implements the default memory map attributes
- Low-cost debug solution that features:
  - Debug access to all memory and registers in the system, including access to memory mapped devices, access to internal core registers when the core is halted, and access to debug control registers even while SYSRESETn is asserted.
  - Serial Wire Debug Port(SW-DP) or Serial Wire JTAG Debug Port (SWJ-DP) debug access
  - Optional Flash Patch and Breakpoint (FPB) unit for implementing breakpoints and code patches

- Optional Data Watchpoint and Trace (DWT) unit for implementing watchpoints, data tracing, and system profiling
- Optional Instrumentation Trace Macrocell (ITM) for support of printf() style debugging
- Optional Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer (TPA), including Single Wire Output (SWO) mode
- Optional Embedded Trace Macrocell (ETM) for instruction trace.

- Bus interfaces:
  - Three Advanced High-performance Bus-Lite (AHB-Lite) interfaces: ICode, Dcode, and System bus interfaces
  - Private Peripheral Bus (PPB) based on Advanced Peripheral Bus (APB) interface
  - Bit-band support that includes atomic bit-band write and read operations.
  - Memory access alignment
  - Write buffer for buffering of write data
  - Exclusive access transfers for multiprocessor systems

6.2 System Manager

6.2.1 Overview

System management includes the following sections:

- System Reset
- System Power Distribution
- SRAM Memory Organization
- System Timer (SysTick)
- Nested Vectored Interrupt Controller (NVIC)
- System Control register

6.2.2 System Reset

The system reset can be issued by one of the events listed below. These reset event flags can be read from SYS_RSTSTS register to determine the reset source. Hardware reset sources are from peripheral signals. Software reset can trigger reset through setting control registers.

- Hardware Reset Sources
  - Power-on Reset
  - Low level on the nRESET pin
  - Watchdog Time-out Reset and Window Watchdog Reset (WDT/WWDT Reset)
  - Low Voltage Reset (LVR)
  - Brown-out Detector Reset (BOD Reset)
  - CPU Lockup Reset
- Software Reset Sources
  - CHIP Reset will reset whole chip by writing 1 to CHIPRST (SYS_IPRST0[0])
  - MCU Reset to reboot but keeping the booting setting from APROM or LDROM by writing 1 to SYSRESETREQ (AIRCR[2])
  - CPU Reset for Cortex-M4 core only by writing 1 to CPURST (SYS_IPRST0[1])

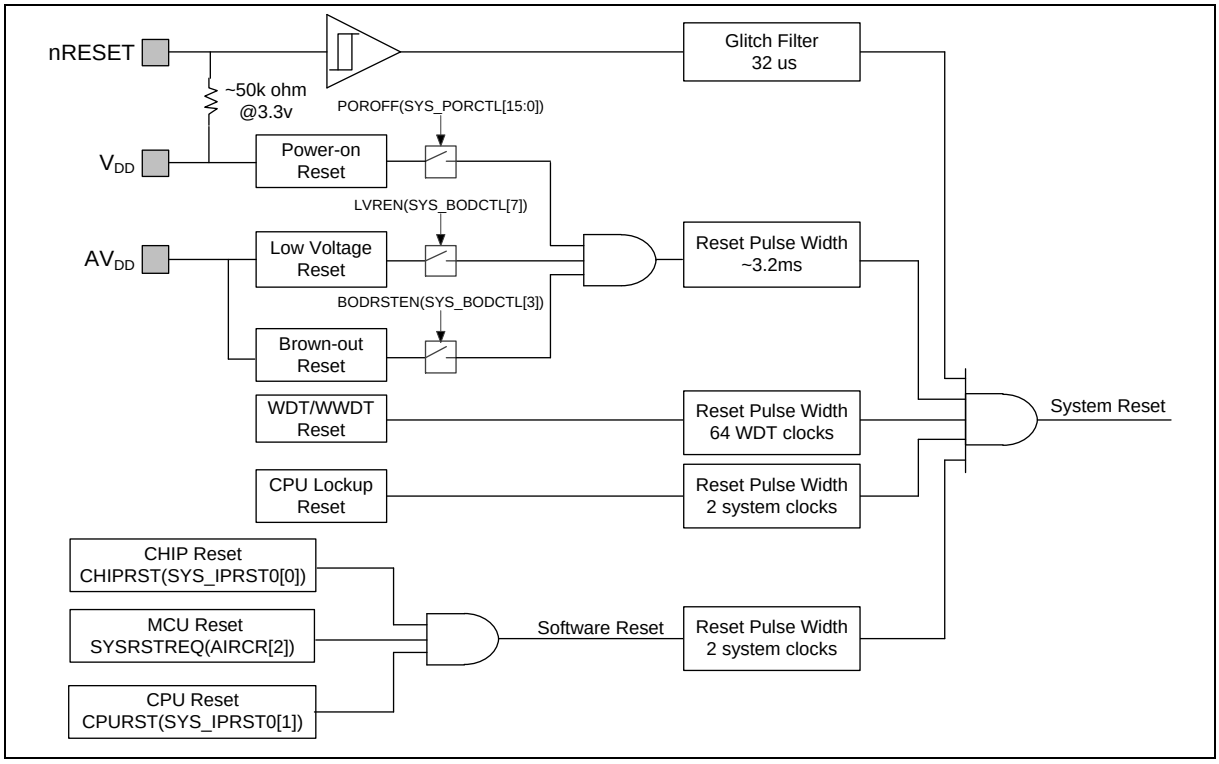


Figure 6.2-1 System Reset Sources

The Mask ROM which is built-in secure boot and ROM ISP function is the initial boot source after chip is powered on or cold reset. User can refer to Mask ROM application note for the detailed features and functions. System cold reset events include nRESET pin, WDT/WWDT, LVR, BOD, CHIP reset and wake-up from Deep Power-down mode (DPD). Other reset events include MCU, CPU, Lockup reset and wake-up from Standby Power-down mode (SPD) are defined as warm reset events. Figure 6.2-2 shows the Reset Event State Machine.

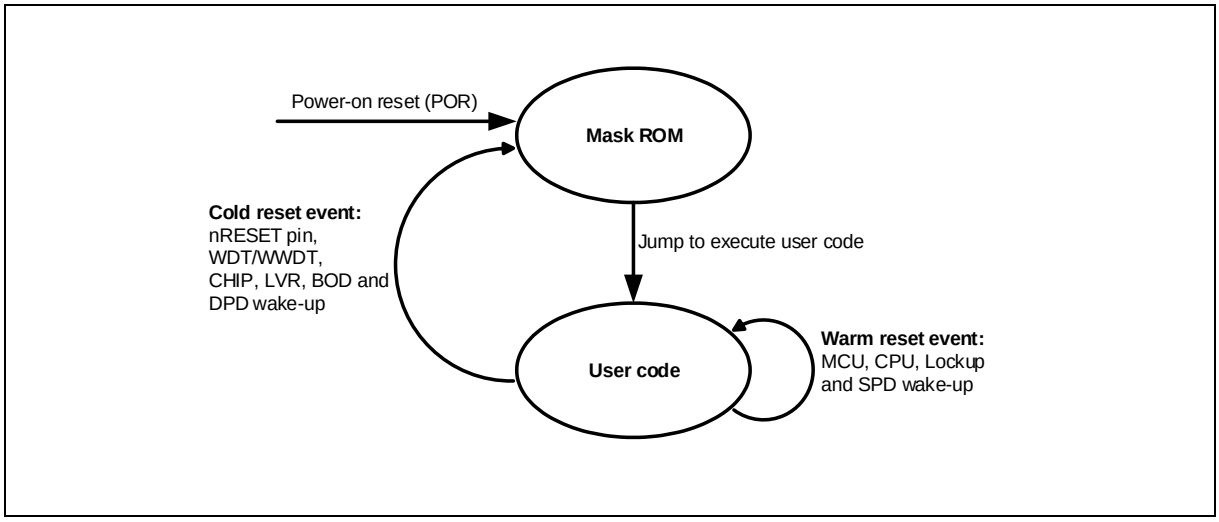


Figure 6.2-2 Reset Event State Machine

There are a total of 9 reset sources in the NuMicro family. In general, CPU reset is used to reset Cortex-M4 only; the other reset sources will reset Cortex-M4 and all peripherals. However, there are small differences between each reset source and they are listed in Table 6.2-1.

Reset Sources Register	POR	NRESET	WDT	LVR	BOD	Lockup	CHIP	MCU	CPU
SYS_RSTSTS	Bit 0 = 1	Bit 1 = 1	Bit 2 = 1	Bit 3 = 1	Bit 4 = 1	Bit 8 = 1	Bit 0 = 1	Bit 5 = 1	Bit 7 = 1
CHIPRST (SYS_IPRST0[0])	0x0	-	-	-	-	-	-	-	-
BODEN (SYS_BODCTL[0])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-
BODVL (SYS_BODCTL[18:16])									
BODRSTEN (SYS_BODCTL[3])									
HXTEN (CLK_PWRCTL[0])	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	-
LXTEN (CLK_PWRCTL[1])	0x0	-	-	-	-	-	-	-	-
WDTCKEN (CLK_APBCLK0[0])	0x1	-	0x1	-	-	-	0x1	-	-
HCLKSEL (CLK_CLKSEL0[2:0])	0x2 PLL	0x2 PLL	0x2 PLL	0x2 PLL	0x2 PLL	0x7 HIRC	0x2 PLL	0x7 HIRC	-
CLK_PCLKDIV	0x11	0x11	0x11	0x11	0x11	0x0	0x11	0x0	-
CLK_PLLCTL	0x8421E	0x8421E	0x8421E	0x8421E	0x8421E	0x5842B	0x8421E	0x5842B	-
WDTSEL (CLK_CLKSEL1[1:0])	0x3	0x3	-	-	-	-	-	-	-
HXTSTB (CLK_STATUS[0])	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	-
LXTSTB (CLK_STATUS[1])	0x0	-	-	-	-	-	-	-	-
PLLSTB (CLK_STATUS[2])	0x1	0x1	0x1	0x1	0x1	0x0	0x1	0x0	-
HIRCSTB (CLK_STATUS[4])	0x1	0x1	0x1	0x1	0x1	0x1	0x1	0x1	-
CLKSFAIL (CLK_STATUS[7])	0x0	0x0	0x0	0x0	0x0	0x0	0x0	0x0	-
PDMSEL (CLK_PMUCTL[2:0])	0x0	-	-	-	-	-	-	-	-
RSTEN (WDT_CTL[1])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-	Reload from CONFIG0	-	-
WDTEN (WDT_CTL[7])									
WDT_CTL except bit 1 and bit 7.	0x0800	0x0800	0x0800	0x0800	0x0800	-	0x0800	-	-

WDT_ALTCTL	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_RLDCNT	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_CTL	0x3F0800	0x3F0800	0x3F0800	0x3F0800	0x3F0800	-	0x3F0800	-	-
WWDT_STATUS	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_CNT	0x3F	0x3F	0x3F	0x3F	0x3F	-	0x3F	-	-
FMC_DFBA	Reload from CONFIG1	Reload from CONFIG1	Reload from CONFIG1	Reload from CONFIG1	Reload from CONFIG1	-	Reload from CONFIG1	-	-
VECMAP (FMC_ISPSTS[23:9])	Reload base on CONFIG0	Reload base on CONFIG0	Reload base on CONFIG0	Reload base on CONFIG0	Reload base on CONFIG0	-	Reload base on CONFIG0	-	-
FMC_CYCCTL	0x8	0x8	0x8	0x8	0x8	0x1	0x8	0x1	-
Debug lock active	Reload base on CONFIG2	Reload base on CONFIG2	Reload base on CONFIG2	Reload base on CONFIG2	Reload base on CONFIG2	-	Reload base on CONFIG2	-	-
Other Peripheral Registers	Reset Value								-
FMC Registers	Reset Value								
Note: '-' means that the value of register keeps original setting.									

Table 6.2-1 Reset Value of Registers

6.2.2.1 nRESET Reset

The nRESET reset means to generate a reset signal by pulling low nRESET pin, which is an asynchronous reset input pin and can be used to reset system at any time. When the nRESET voltage is lower than 0.2 V_{DD} and the state keeps longer than 32 us (glitch filter), chip will be reset. The nRESET reset will control the chip in reset state until the nRESET voltage rises above 0.7 V_{DD} and the state keeps longer than 32 us (glitch filter). The PINRF(SYS_RSTSTS[1]) will be set to 1 if the previous reset source is nRESET reset. Figure 6.2-3 shows the nRESET reset waveform. Note that the nRESET de-glitch time is typically 300ns in SPDx and DPDx mode.

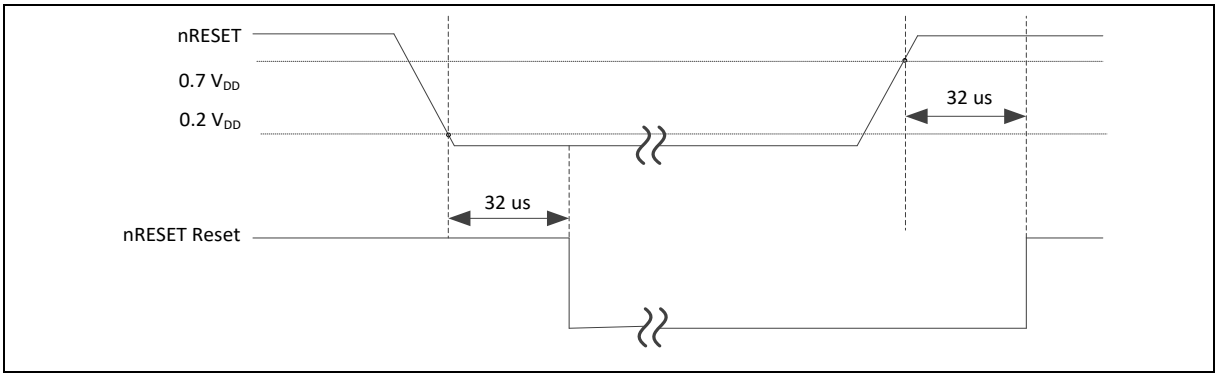


Figure 6.2-3 nRESET Reset Waveform(Except SPDx and DPDx mode)

6.2.2.2 Power-on Reset (POR)

The Power-on reset (POR) is used to generate a stable system reset signal and forces the system to be reset when power-on to avoid unexpected behavior of MCU. When applying the power to MCU, the

POR module will detect the rising voltage and generate reset signal to system until the voltage is ready for MCU operation. At POR reset, the PORF(SYS_RSTSTS[0]) will be set to 1 to indicate there is a POR reset event. The PORF(SYS_RSTSTS[0]) bit can be cleared by writing 1 to it. Figure 6.2-4 shows the power-on reset waveform.

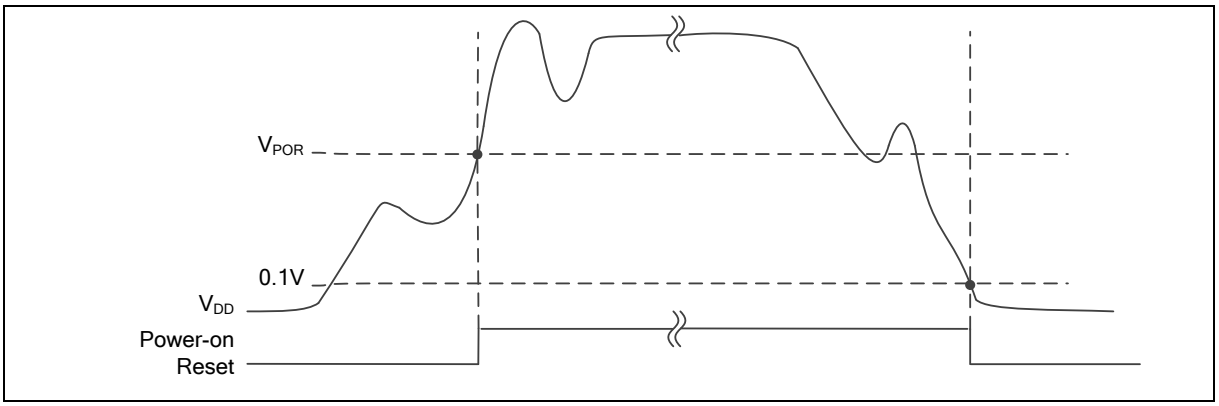


Figure 6.2-4 Power-on Reset (POR) Waveform

6.2.2.3 Low Voltage Reset (LVR)

If the Low Voltage Reset function is enabled by setting the Low Voltage Reset Enable Bit LVREN (SYS_BODCTL[7]) to 1, after 100us~200us delay, LVR detection circuit will be stable and the LVR function will be active. Then LVR function will detect AV_{DD} during system operation. When the AV_{DD} voltage is lower than V_{LVR} and the state keeps longer than De-glitch time set by LVRDGSEL (SYS_BODCTL[14:12]), chip will be reset. The LVR reset will control the chip in reset state until the AV_{DD} voltage rises above V_{LVR} and the state keeps longer than De-glitch time set by LVRDGSEL (SYS_BODCTL[14:12]). The default setting of Low Voltage Reset is enabled without De-glitch function. Figure 6.2-5 shows the Low Voltage Reset waveform.

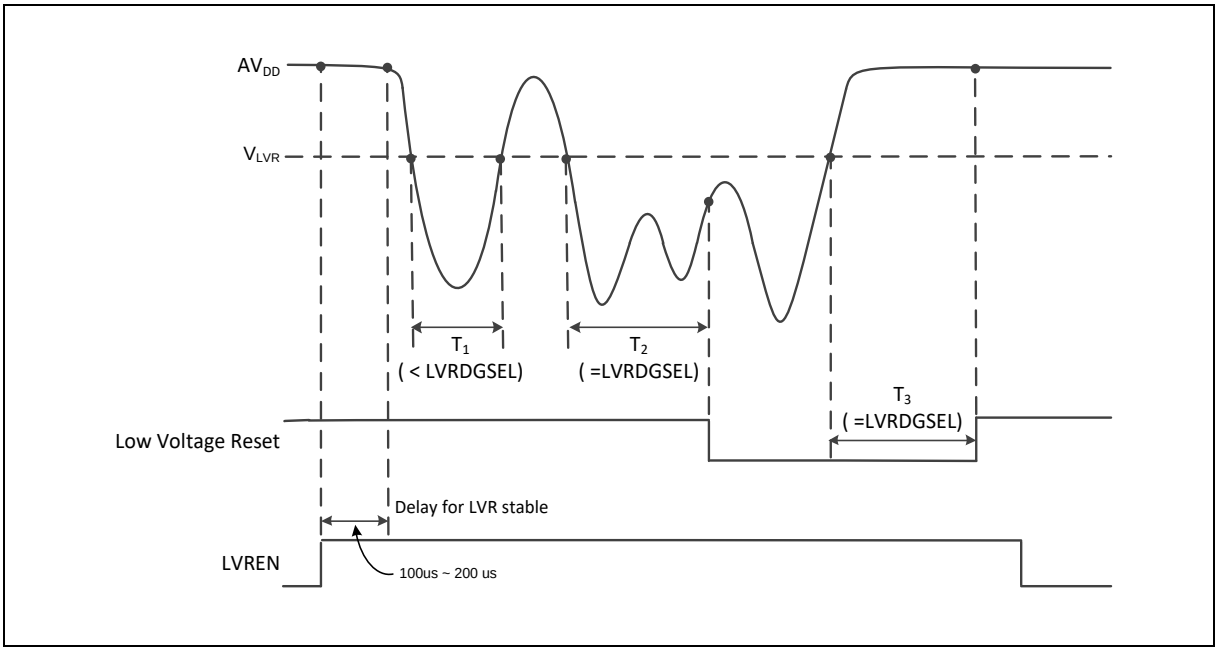


Figure 6.2-5 Low Voltage Reset (LVR) Waveform

6.2.2.4 Brown-out Detector Reset (BOD Reset)

If the Brown-out Detector (BOD) function is enabled by setting the Brown-out Detector Enable Bit BODEN (SYS_BODCTL[0]), Brown-out Detector function will detect AV_{DD} during system operation. When the AV_{DD} voltage is lower than V_{BOD} which is decided by BODEN and BODVL (SYS_BODCTL[18:16]) and the state keeps longer than De-glitch time set by BODDGSEL (SYS_BODCTL[10:8]), chip will be reset. The BOD reset will control the chip in reset state until the AV_{DD} voltage rises above V_{BOD} and the state keeps longer than De-glitch time set by BODDGSEL. The default value of BODEN, BODVL and BODRSTEN (SYS_BODCTL[3]) is set by Flash controller user configuration register CBODEN (CONFIG0 [19]), CBOV (CONFIG0 [23:21]) and CBORST(CONFIG0[20]) respectively. User can determine the initial BOD setting by setting the CONFIG0 register. Figure 6.2-6 shows the Brown-out Detector waveform.

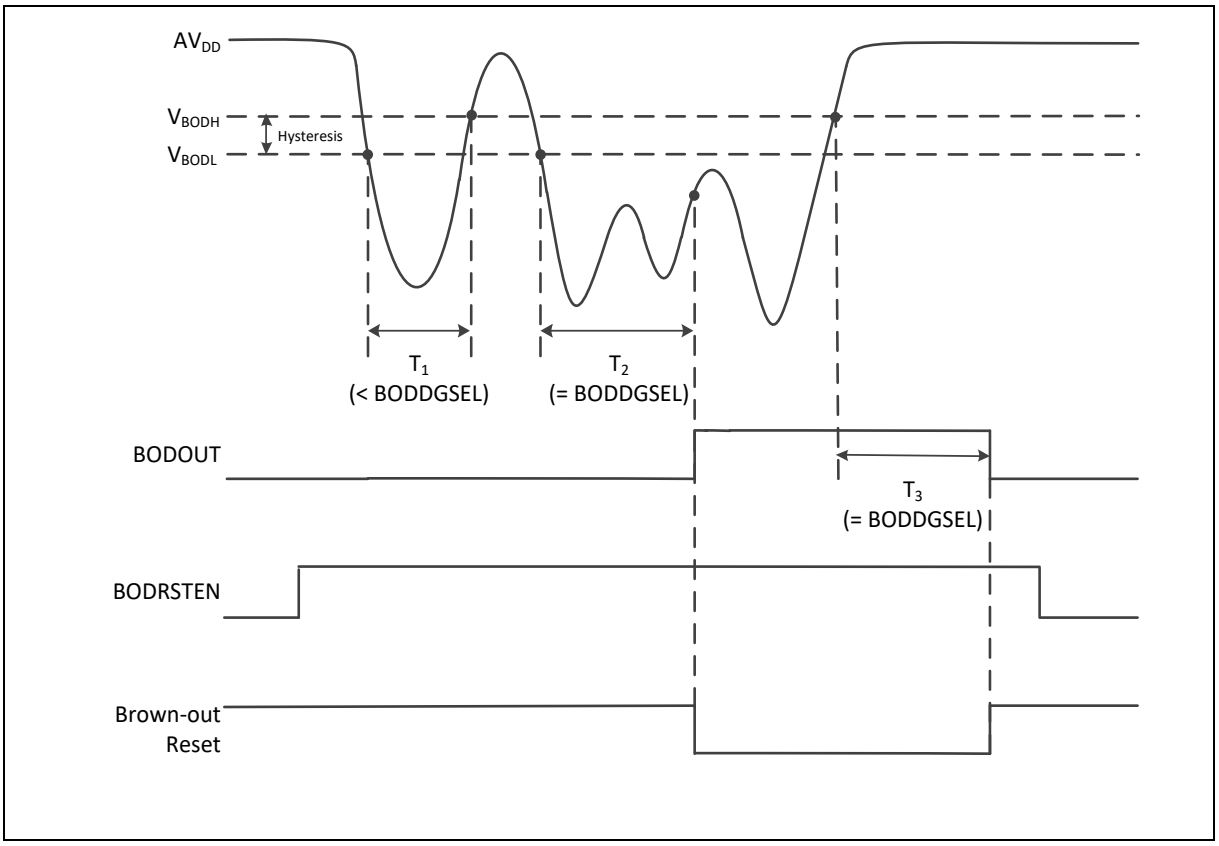


Figure 6.2-6 Brown-out Detector (BOD) Waveform

6.2.2.5 Watchdog Timer Reset (WDT)

In most industrial applications, system reliability is very important. To automatically recover the MCU from failure status is one way to improve system reliability. The watchdog timer(WDT) is widely used to check if the system works fine. If the MCU is crashed or out of control, it may cause the watchdog time-out. User may decide to enable system reset during watchdog time-out to recover the system and take action for the system crash/out-of-control after reset.

Software can check if the reset is caused by watchdog time-out to indicate the previous reset is a watchdog reset and handle the failure of MCU after watchdog time-out reset by checking WDTRF(SYS_RSTSTS[2]).

6.2.2.6 CPU Lockup Reset

CPU enters lockup status after CPU produces hardfault at hardfault handler and chip gives immediate indication of seriously errant kernel software. This is the result of the CPU being locked because of an unrecoverable exception following the activation of the processor's built in system state protection

hardware. When chip enters debug mode, the CPU lockup reset will be ignored.

6.2.2.7 CPU Reset, CHIP Reset and MCU Reset

The CPU Reset means only Cortex-M4 core is reset and all other peripherals remain the same status after CPU reset. User can set the CPURST(SYS_IPRST0[1]) to 1 to assert the CPU Reset signal.

The CHIP Reset is same with Power-on Reset. The CPU and all peripherals are reset and BS(FMC_ISPCTL[1]) bit is automatically reloaded from CONFIG0 setting. User can set the CHIPRST(SYS_IPRST0[1]) to 1 to assert the CHIP Reset signal.

The MCU Reset is similar with CHIP Reset. The difference is that BS(FMC_ISPCTL[1]) will not be reloaded from CONFIG0 setting and keep its original software setting for booting from APROM or LDROM. User can set the SYSRESETREQ(AIRCR[2]) to 1 to assert the MCU Reset.

6.2.3 System Power Distribution

In this chip, power distribution is divided into five segments:

- Analog power from AV_{DD} and AV_{SS} provides the power for analog components operation.
- Digital power from V_{DD} and V_{SS} supplies the power to the internal regulator which provides power levels(PL0/PL1) for digital operation and I/O pins.
- I/O power from V_{DDIO} supplies the power for PA.0 ~ PA.5
- USB 2.0 transceiver power from HSUSB_VDD33 offers the power for operating the USB transceiver.
- RTC power from V_{BAT} provides the power for RTC, backup registers and PF.4 ~ PF.11.

The outputs of internal voltage regulator, LDO_CAP, require an external capacitor which should be located close to the corresponding pin. Analog power (AV_{DD}) should be the same voltage level of the digital power (V_{DD}). Figure 6.2-7 shows the NuMicro M463/M467 series power distribution.

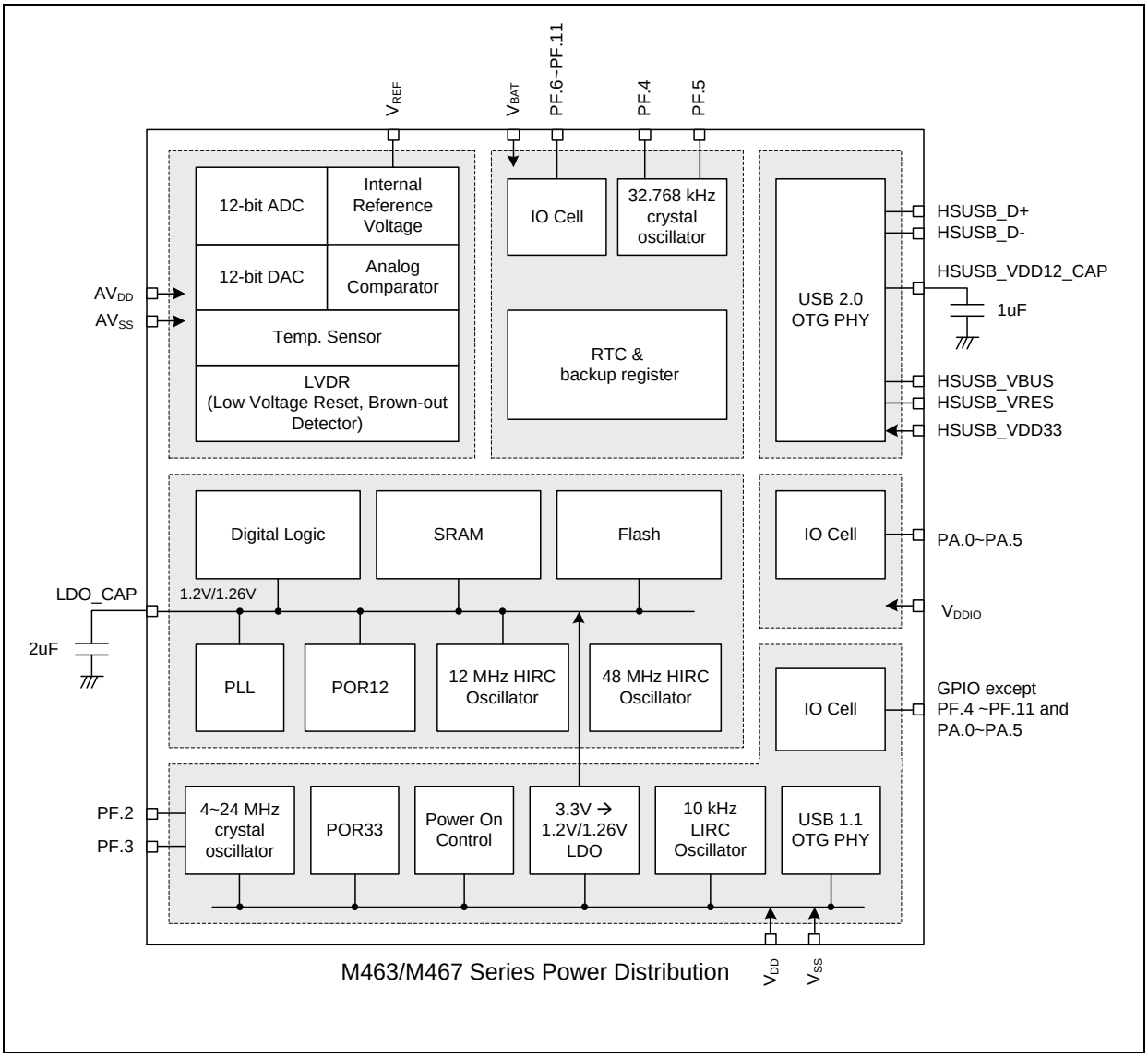


Figure 6.2-7 NuMicro M463/M467 series Power Distribution Diagram

Note:

1. When VBAT power source first power-on, the power-on reset will happened and reset all VBAT domain circuit. The I/O in VBAT domain (PF.4 ~ PF.11) will become floating state and make additional leakage in VBAT domain. User should power on VDD first to reset chip and set I/O control to make these I/Os becomes a static state to prevent additional leakage.

6.2.4 Power Modes and Wake-up Sources

The NuMicro M463/M467 series has power manager unit to support several operating modes for saving power. Table 6.2-2 lists all power mode at NuMicro M463/M467 series.

Mode	CPU Operating and HCLK Maximum Speed (MHz)	APB and PCLK Maximum Speed (MHz)	LDO_CAP (V)	Clock Disable
Normal mode Power Level 1 (PL1)	180	90	1.20	All clocks are disabled by control register.

Turbo mode Power Level 0 (PL0)	200	100	1.26	All clocks are disabled by control register.
Idle mode	CPU enters Sleep mode	CPU enter Sleep mode	1.20/1.26	Only CPU clock is disabled.
Fast Wakeup Power-down mode (FWPD)	CPU enters Deep Sleep mode	CPU enters Deep Sleep mode	1.20/1.26	Most clocks are disabled except LIRC/LXT, and only RTC/WDT/Timer/UART peripheral clocks still enable if their clock sources are selected as LIRC/LXT.
Normal Power-down mode (NPD)	CPU enters Deep Sleep mode	CPU enters Deep Sleep mode	1.20/1.26	Most clocks are disabled except LIRC/LXT, and only RTC/WDT/Timer/UART peripheral clocks still enable if their clock sources are selected as LIRC/LXT.
Low leakage Power-down mode (LLPD)	CPU enters Deep Sleep mode	CPU enters Deep Sleep mode	0.9	Most clocks are disabled except LIRC/LXT, and only RTC/WDT/Timer/UART peripheral clocks still enable if their clock sources are selected as LIRC/LXT.
Standby Power-down mode (SPD)	Power off	Power off	Floating	Only LIRC still enable for wake-up timer usage. Only LXT still enabled for RTC function usage.
Deep Power-down mode (DPD)	Power off	Power off	Floating	Only LIRC still enabled for wake-up timer usage. Only LXT still enabled for RTC function usage.

Table 6.2-2 Power Mode Table

Note:

1. User must turn on LIRC before entering SPD mode.

There are different power mode entry setting for each power mode, with different entry setting and leaving condition. Table 6.2-3 shows the entry setting for each power mode. When chip power-on, chip is running at normal mode. User can enter each mode by setting SLEEPDEEP (SCR[2]), PDEN (CLK_PWRCTL[7]) and PDMSEL (CLK_PMUCTL[2:0]) and execute WFI instruction.

Register/Instruction Mode	SLEEPDEEP (SCR[2])	PDEN (CLK_PWRCTL[7])	PDMSEL (CLK_PMUCTL[2:0])	CPU Run WFI Instruction
Normal mode	0	0	0	NO
Idle mode	0	0	0	YES
Fast Wakeup Power-down mode	1	1	2	YES
Normal Power-down mode	1	1	0	YES
Low leakage Power-down mode	1	1	1	YES
Standby Power-down mode	1	1	4	YES
Deep Power-down mode	1	1	6	YES

Table 6.2-3 Power Mode Difference Table

There are several wake-up sources in Idle mode and Power-down mode. Table 6.2-4 lists the available clocks for each power mode.

Power Mode	Normal Mode	Idle Mode	Power-Down Mode
Definition	CPU is in active state	CPU is in sleep state	CPU is in sleep state and all

			clocks stop except LXT and LIRC. SRAM content retain.
Entry Condition	Chip is in normal mode after system reset released	CPU executes WFI instruction.	CPU sets sleep mode enable and power down enable and executes WFI instruction.
Wake-up Sources	N/A	All interrupts	RTC, WDT, I²C, Timer, UART, BOD, GPIO, INT, SDH, USCI, USBD, USBH, OTG, ACMP, EMAC, HSUSBD, HSUSBH and HSOTG.
Available Clocks	All	All except CPU clock	LXT and LIRC
After Wake-up	N/A	CPU back to normal mode	CPU back to normal mode

Table 6.2-4 Power Mode Definition Table

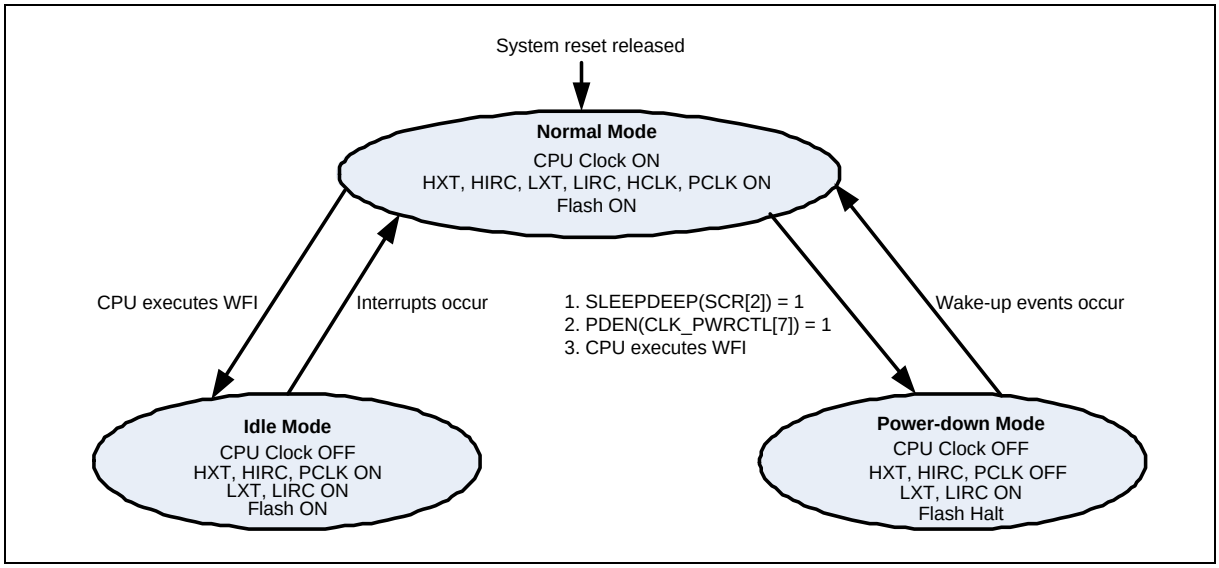


Figure 6.2-8 Power Mode State Machine

	Idle Mode	NPD, LLPD, FWPD	SPD	DPD
HXT	ON	Halt	Halt	Halt
HIRC	ON	Halt	Halt	Halt
LXT	ON	ON/OFF[1]	ON/OFF[1]	ON/OFF[1]
LIRC	ON	ON/OFF[2]	ON/OFF[2]	ON/OFF[2,8]
PLL	ON	Halt	Halt	Halt
HCLK/PCLK	ON	Halt	Halt	Halt
CPU	Halt	Halt	Halt	Halt
SRAM retention	ON	ON	ON/OFF[7]	Halt
FLASH	ON	Halt	Halt	Halt
TIMER	ON	ON/OFF[3]	Halt	Halt
WDT	ON	ON/OFF[4]	Halt	Halt
RTC	ON	ON/OFF[5]	ON/OFF[9]	ON/OFF[9]
UART	ON	ON/OFF[6]	Halt	Halt
Others	ON	Halt	Halt	Halt

Table 6.2-5 Clocks in Power Modes

Note:

1. LXT ON or OFF depends on software setting in normal mode.
2. LIRC ON or OFF depends on software setting in normal mode.
3. If TIMER clock source is selected as LIRC/LXT and LIRC/LXT is on.
4. If WDT clock source is selected as LIRC/LXT and LIRC/LXT is on.
5. If RTC clock source is selected as LIRC/LXT and LIRC/LXT is on.
6. If UART clock source is selected as LXT and LXT is on.
7. SRAM retention size depends on software setting in normal mode.
8. If timer wake-up function is disabled, LIRC will be disabled automatically when chip enters DPD mode for power saving.
9. If RTC clock source is selected as LXT and LXT is on.

Wake-up sources in Normal Power-down mode (NPD):

RTC, WDT, I²C, Timer, UART, USCI, BOD, GPIO, INT, SDH, USBD, USBH, OTG, ACMP, EMAC, HSUSBD, HSUSBH and HSOTG.

After chip enters power down, the following wake-up sources can wake chip up to normal mode. Table 6.2-6 lists the condition about how to enter Power-down mode again for each peripheral.

User needs to wait this condition before setting PDEN(CLK_PWRCTL[7]) and execute WFI to enter Power-down mode.

Wake-Up	Wake-Up Condition	Power-Down Mode	Re-Entering Power-Down Mode Condition

Source		NPD/ FWPD/ LLPD	SPD	DPD	
BOD	Brown-Out Detector Reset / Interrupt	V	-	-	After software writes 1 to clear BODIF (SYS_BODCTL[4]).
	Brown-Out Detector Reset	-	V	-	After software writes 1 (CLK_PMUSTS[31]) to clear BODWK (CLK_PMUSTS[13]) when SPD mode is entered.
LVR	LVR Reset	V	-	-	After software writes 1 to clear LVRF (SYS_RSTSTS[3])
		-	V	-	After software writes 1 (CLK_PMUSTS[31]) to clear LVRWK (CLK_PMUSTS[12]) when SPD mode is entered.
POR	POR Reset	V	V	V	After software writes 1 to clear PORF (SYS_RSTSTS[0]).
RST pin	RST pin Reset	V	-	-	After software writes 1 to clear PINRF (SYS_RSTSTS[1]).
		-	V	V	After software writes 1 (CLK_PMUSTS[31]) to clear RSTWK (CLK_PMUSTS[15]) when SPD or DPD mode is entered.
INT	External Interrupt	V	-	-	After software write 1 to clear the Px_INTSRC[n] bit.
GPIO	GPIO Interrupt	V	-	-	After software write 1 to clear the Px_INTSRC[n] bit.
GPIO(PA~PD) Wake-up pin	rising or falling edge event, 62-pin	-	V	-	After software writes 1 (CLK_PMUSTS[31]) to clear GPxWK (CLK_PMUSTS[11:8]) when SPD mode is entered.
GPIO(PC.0/PB.0/PB.2/PB.12/PF.6) Wake-up pin	rising or falling edge event, 5-pin	-	-	V	After software writes 1 (CLK_PMUSTS[31]) to clear PINWKx (CLK_PMUSTS[6:3] and CLK_PMUSTS[0]) when DPD mode is entered.
VBUS(PA.12)	rising or falling edge event	-	-	V	After software writes 1 (CLK_PMUSTS[31]) to clear VBUSWK (CLK_PMUSTS[7]) when DPD mode is entered.
TIMER	Timer Interrupt	V	-	-	After software writes 1 to clear TWKF (TIMERx_INTSTS[1]) and TIF (TIMERx_INTSTS[0]).
Wakeup timer	Wakeup by wake-up timer time-out	-	V	V	After software writes 1 (CLK_PMUSTS[31]) to clear TMRWK (CLK_PMUSTS[1]) when SPD or DPD mode is entered.
WDT	WDT Interrupt	V	-	-	After software writes 1 to clear WKF (WDT_CTL[5]) (Write Protect).
RTC	Alarm Interrupt	V	-	-	After software writes 1 to clear ALMIF (RTC_INTSTS[0]).
	Time Tick Interrupt	V	-	-	After software writes 1 to clear TICKIF (RTC_INTSTS[1]).
	Wakeup by RTC alarm	-	V	V	After software writes 1 (CLK_PMUSTS[31]) to clear RTCWK (CLK_PMUSTS[2]) when DPD or SPD mode is entered.
	Wakeup by RTC tick time	-	V	V	After software writes 1 (CLK_PMUSTS[31]) to clear RTCWK (CLK_PMUSTS[2]) when DPD or SPD mode is entered.

	Wakeup by tamper event	-	V	V	After software writes 1 (CLK_PMUSTS[31]) to clear RTCWK (CLK_PMUSTS[2]) when DPD or SPD mode is entered.
UART	nCTS wake-up	V	-	-	After software writes 1 to clear CTSWKF (UARTx_WKSTS[0]).
	RX Data wake-up	V	-	-	After software writes 1 to clear DATWKF (UARTx_WKSTS[1]).
	Received FIFO Threshold Wake-up	V	-	-	After software writes 1 to clear RFRTWKF (UARTx_WKSTS[2]).
	RS-485 AAD Mode Wake-up	V	-	-	After software writes 1 to clear RS485WKF (UARTx_WKSTS[3]).
	Received FIFO Threshold Time-out Wake-up	V	-	-	After software writes 1 to clear TOUTWKF (UARTx_WKSTS[4]).
USCI UART	CTS Toggle	V	-	-	After software writes 1 to clear WKF (UUART_WKSTS[0]).
	Data Toggle	V	-	-	After software writes 1 to clear WKF (UUART_WKSTS[0]).
USCI I^2C	Data toggle	V	-	-	After software writes 1 to clear WKF (UI2C_WKSTS[0]).
	Address match	V	-	-	After software writes 1 to clear WKAKDONE (UI2C_PROTSTS[16], then writes 1 to clear WKF (UI2C_WKSTS[0]).
USCI SPI	SS Toggle	V	-	-	After software writes 1 to clear WKF (USPI_WKSTS[0]).
I^2C	Address match wake-up	V	-	-	After software writes 1 to clear WKAKDONE (I2C_WKSTS[1]). Then software writes 1 to clear WKIF(I2C_WKSTS[0]).
SDH	Card detection	V	-	-	Clear CDIF (SDH_INTSTS[8]) after SDH wake-up.
USBD	1.Remote Wake-up 2.Plug in wake-up	V	-	-	After software writes 1 to clear BUSIF (USBD_INTSTS[0]).
USBH	1.Connection detected 2.Disconnect detected 3.Remote-wakeup	V	-	-	1.After write 1 to clear RHSC (HcInterruptStatus[7]). 2.After write 1 to clear RHSC (HcInterruptStatus[7]). 3.After write 1 to clear RHSC (HcInterruptStatus[7]). and port suspended.
OTG	ID pin state be change	V	-	-	After software writes 1 to set WKEN(OTG_CTL[5]).
ACMP	Comparator Power-Down Wake-Up Interrupt	V	-	-	After software writes 1 to clear WKIF0 (ACMP_STATUS[8]), WKIF1 (ACMP_STATUS[9]), WKIF2 (ACMP_STATUS2[8]) and WKIF3 (ACMP_STATUS2[9]).
	ACMPO status change	-	V	-	After software writes 1 (CLK_PMUSTS[31]) to clear ACMPWKx (CLK_PMUSTS[19:16]) when SPD mode is entered.
EMAC	Magic Packet Detection	V	-	-	After software read to clear MGKPRCVD(PMT_Control_Status[5]), Magic Packet Received.
HSUSBD	1.V_{BUS} plug in/out 2.USB Resume/Reset	V	-	-	1. After software writes 1 to clear HSUSBD_BUSINTSTS[8](V_{BUS}) 2. After software writes 1 to clear HSUSBD_BUSINTSTS[2](Resume) or

					HSUSBD_BUSINTSTS[1](Reset)
HSUSBH	1. Remote-wakeup (while port in suspend state) 2. Connector plug-in detect 3. Connector plug-out detect (while port in suspend state) 4. Overcurrent detect	V	-	-	After software writes 1 to clear PCD (HSUSBH_USTSR[2])
HSOTG	ID pin state be change	V	-	-	After software writes 1 to set WKEN(HSOTG_CTL[5]).

Table 6.2-6 Re-Entering Power-down Mode Condition

6.2.5 Power Modes and Power Level Transition

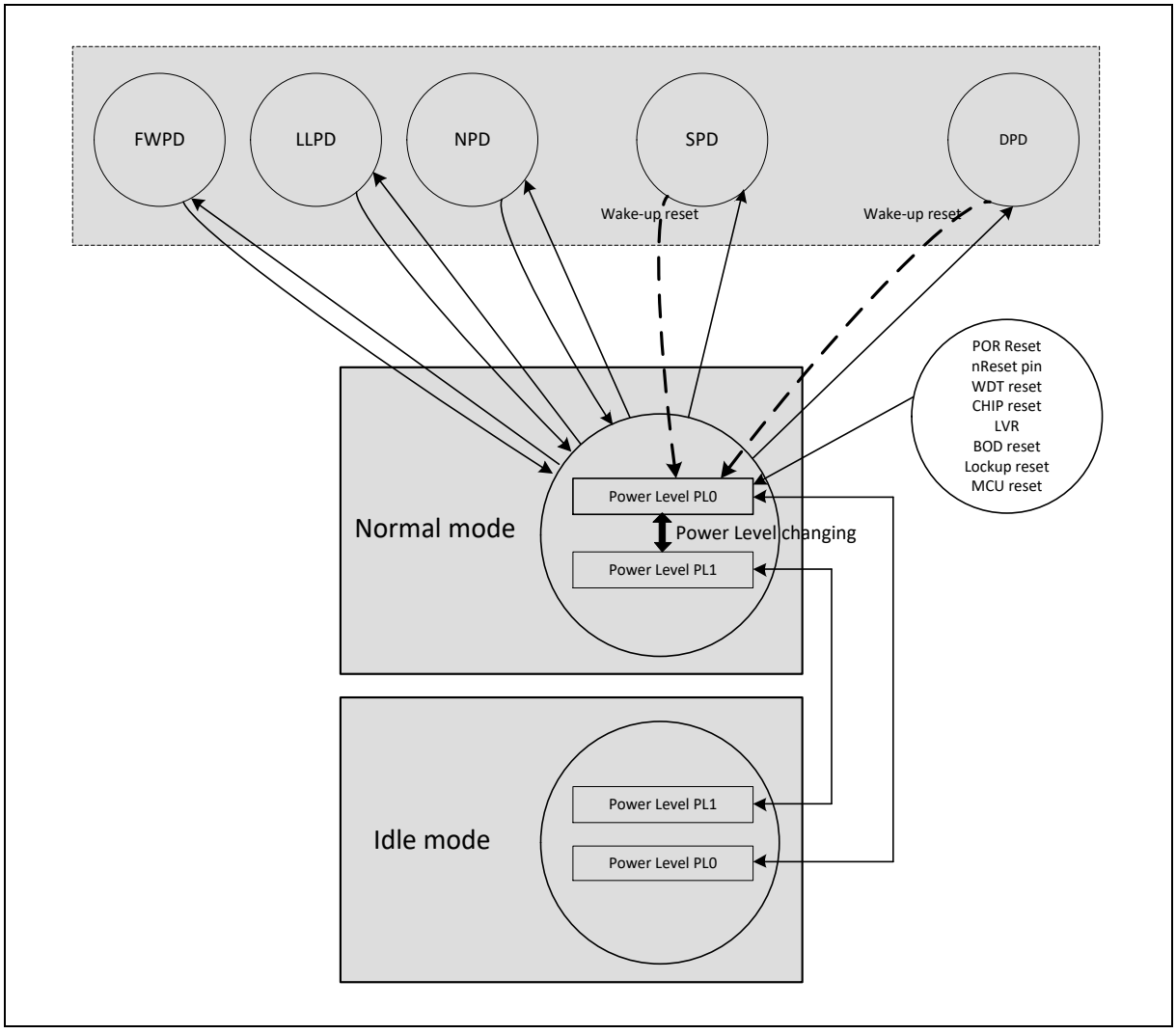


Figure 6.2-9 NuMicro M463/M467 Series Power Distribution Diagram

6.2.6 System Memory Map

The NuMicro M463/M467 series provides 4G-byte addressing space. The memory locations assigned to each on-chip controllers are shown in Table 6.2-7. The detailed register definition, memory space, and programming will be described in the following sections for each on-chip peripheral. The NuMicro M463/M467 series only supports little-endian data format.

Address Space	Token	Controllers
Flash and SRAM Memory Space		
0x0000_0000 – 0x000F_FFFF	FLASH_BA	FLASH APROM Memory Space (1024 Kbytes)
0x0010_0000 – 0x020F_FFFF	SPIM_BA	SPIM Memory Space (32 Mbytes) with FLASH 1024 Kbytes
0x0A00_0000 – 0x0BFF_FFFF	HRAM_BA	HyperRAM Memory Executable Space (32 Mbytes)
0x0F10_0000 – 0x0F10_1FFF	FLASH_BA	FLASH LDROM Memory Space (8 Kbytes)
0x1000_0000 – 0x1007_FFFF	SRAM_BA	SRAM Memory Space Remapping (512 Kbytes)
0x2000_0000 – 0x2007_FFFF	SRAM_BA	SRAM Memory Space (512 Kbytes)
0x6000_0000 – 0x602F_FFFF	EXTMEM_BA	External Memory Space (3 Mbytes)
0x8000_0000 – 0x81FF_FFFF	EXTHRAM_BA	External HyperRAM Memory Space (32 Mbytes)
Peripheral Controllers Space (0x4000_0000 – 0x400F_FFFF)		
0x4000_0000 – 0x4000_01FF	SYS_BA	System Control Registers
0x4000_0200 – 0x4000_02FF	CLK_BA	Clock Control Registers
0x4000_0300 – 0x4000_03FF	NMI_BA	NMI Control Registers
0x4000_4000 – 0x4000_4FFF	GPIO_BA	GPIO Control Registers
0x4000_7000 – 0x4000_7FFF	SPIM_BA	SPIM Control Registers
0x4000_8000 – 0x4000_8FFF	PDMA0_BA	Peripheral DMA0 Control Registers
0x4000_9000 – 0x4000_9FFF	USBH_BA	USB Host Control Registers
0x4000_C000 – 0x4000_CFFF	FMC_BA	Flash Memory Control Registers
0x4000_D000 – 0x4000_DFFF	SDH0_BA	SDHOST0 Control Registers
0x4000_E000 – 0x4000_EFFF	SDH1_BA	SDHOST1 Control Registers
0x4001_0000 – 0x4001_0FFF	EBI_BA	External Bus Interface Control Registers
0x4001_2000 – 0x4001_1FFF	EMAC0_BA	EMAC0 Control Registers
0x4001_8000 – 0x4001_8FFF	PDMA1_BA	Peripheral DMA1 Control Registers
0x4001_9000 – 0x4001_9FFF	HSUSBD_BA	HSUSBD Control Registers
0x4001_A000 – 0x4001_AFFF	HSUSBH _BA	HSUSBH Host Control Registers
0x4002_0000 – 0x4002_1FFF	CANFD0_BA	CANFD0 Control Registers
0x4002_4000 – 0x4002_5FFF	CANFD1_BA	CANFD1 Control Registers
0x4002_8000 – 0x4002_9FFF	CANFD2_BA	CANFD2 Control Registers
0x4002_C000 – 0x4002_DFFF	CANFD3_BA	CANFD3 Control Registers
0x4003_0000 – 0x4003_0FFF	CCAP_BA	CCAP Control Registers
0x4003_1000 – 0x4003_1FFF	CRC_BA	CRC Generator Registers

0x4003_2000 – 0x4003_4FFF	CRYPTO_BA	Cryptographic Accelerator Registers
0x4003_5000 – 0x4003_5FFF	KS_BA	Key Store Control Registers
APB Controllers Space (0x4000_0000 ~ 0x400F_FFFF)		
0x4004_0000 – 0x4004_00FF	WDT_BA	Watchdog Timer Control Registers
0x4004_0100 – 0x4004_0FFF	WWDT_BA	Window Watchdog Timer Control Registers
0x4004_1000 – 0x4004_1FFF	RTC_BA	Real Time Clock (RTC) Control Register
0x4004_3000 – 0x4004_3FFF	EADC0_BA	Enhanced Analog-Digital-Converter 0 (EADC0) Control Registers
0x4004_5000 – 0x4004_5FFF	ACMP01_BA	Analog Comparator 0/1 Control Registers
0x4004_7000 – 0x4004_7FFF	DAC_BA	DAC Control Registers
0x4004_8000 – 0x4004_8FFF	I2S0_BA	I2S0 Interface Control Registers
0x4004_9000 – 0x4004_9FFF	I2S1_BA	I2S1 Interface Control Registers
0x4004_B000 – 0x4004_BFFF	EADC1_BA	Enhanced Analog-Digital-Converter 1 (EADC1) Control Registers
0x4004_D000 – 0x4004_DFFF	OTG_BA	OTG Control Registers
0x4004_F000 – 0x4004_FFFF	HSOTG_BA	HSOTG Control Registers
0x4005_0000 – 0x4005_0FFF	TMR01_BA	Timer0/Timer1 Control Registers
0x4005_1000 – 0x4005_1FFF	TMR23_BA	Timer2/Timer3 Control Registers
0x4005_8000 – 0x4005_8FFF	EPWM0_BA	EPWM0 Control Registers
0x4005_9000 – 0x4005_9FFF	EPWM1_BA	EPWM1 Control Registers
0x4005_A000 – 0x4005_AFFF	BPWM0_BA	BPWM0 Control Registers
0x4005_B000 – 0x4005_BFFF	BPWM1_BA	BPWM1 Control Registers
0x4006_0000 – 0x4006_0FFF	QSPI0_BA	QSPI0 Control Registers
0x4006_1000 – 0x4006_1FFF	SPI0_BA	SPI0 Control Registers
0x4006_2000 – 0x4006_2FFF	SPI1_BA	SPI1 Control Registers
0x4006_3000 – 0x4006_3FFF	SPI2_BA	SPI2 Control Registers
0x4006_4000 – 0x4006_4FFF	SPI3_BA	SPI3 Control Registers
0x4006_5000 – 0x4006_5FFF	SPI4_BA	SPI4 Control Registers
0x4006_6000 – 0x4006_6FFF	SPI5_BA	SPI5 Control Registers
0x4006_7000 – 0x4006_7FFF	SPI6_BA	SPI6 Control Registers
0x4006_8000 – 0x4006_8FFF	SPI7_BA	SPI7 Control Registers
0x4006_9000 – 0x4006_9FFF	QSPI1_BA	QSPI1 Control Registers
0x4006_B000 – 0x4006_BFFF	SPI8_BA	SPI8 Control Registers
0x4006_C000 – 0x4006_CFFF	SPI9_BA	SPI9 Control Registers
0x4006_D000 – 0x4006_DFFF	SPI10_BA	SPI10 Control Registers
0x4007_0000 – 0x4007_0FFF	UART0_BA	UART0 Control Registers
0x4007_1000 – 0x4007_1FFF	UART1_BA	UART1 Control Registers

0x4007_2000 – 0x4007_2FFF	UART2_BA	UART2 Control Registers
0x4007_3000 – 0x4007_3FFF	UART3_BA	UART3 Control Registers
0x4007_4000 – 0x4007_4FFF	UART4_BA	UART4 Control Registers
0x4007_5000 – 0x4007_5FFF	UART5_BA	UART5 Control Registers
0x4007_6000 – 0x4007_6FFF	UART6_BA	UART6 Control Registers
0x4007_7000 – 0x4007_7FFF	UART7_BA	UART7 Control Registers
0x4007_8000 – 0x4007_8FFF	UART8_BA	UART8 Control Registers
0x4007_9000 – 0x4007_9FFF	UART9_BA	UART9 Control Registers
0x4008_0000 – 0x4008_0FFF	I2C0_BA	I2C0 Control Registers
0x4008_1000 – 0x4008_1FFF	I2C1_BA	I2C1 Control Registers
0x4008_2000 – 0x4008_2FFF	I2C2_BA	I2C2 Control Registers
0x4008_3000 – 0x4008_3FFF	I2C3_BA	I2C3 Control Registers
0x4008_4000 – 0x4008_4FFF	I2C4_BA	I2C4 Control Registers
0x4009_0000 – 0x4009_0FFF	SC0_BA	Smartcard Host 0 Control Registers
0x4009_1000 – 0x4009_1FFF	SC1_BA	Smartcard Host 1 Control Registers
0x4009_2000 – 0x4009_2FFF	SC2_BA	Smartcard Host 2 Control Registers
0x4009_7000 – 0x4009_7FFF	EADC2_BA	Enhanced Analog-Digital-Converter 2(EADC2) Control Registers
0x400B_0000 – 0x400B_0FFF	EQEI0_BA	EQEI0 Control Registers
0x400B_1000 – 0x400B_1FFF	EQEI1_BA	EQEI1 Control Registers
0x400B_2000 – 0x400B_2FFF	EQEI2_BA	EQEI2 Control Registers
0x400B_3000 – 0x400B_3FFF	EQEI3_BA	EQEI3 Control Registers
0x400B_4000 – 0x400B_4FFF	ECAP0_BA	ECAP0 Control Registers
0x400B_5000 – 0x400B_5FFF	ECAP1_BA	ECAP1 Control Registers
0x400B_6000 – 0x400B_6FFF	ECAP2_BA	ECAP2 Control Registers
0x400B_7000 – 0x400B_7FFF	ECAP3_BA	ECAP3 Control Registers
0x400B_9000 – 0x400B_9FFF	TRNG_BA	TRNG Control Registers
0x400C_0000 – 0x400C_0FFF	USBD_BA	USB Device Control Register
0x400C_2000 – 0x400C_2FFF	KPI_BA	Key Pad Control Register
0x400C_3000 – 0x400C_3FFF	PSIO_BA	PSIO Device Control Register
0x400C_9000 – 0x400C_9FFF	ACMP23_BA	Analog Comparator 2/3 Control Registers
0x400C_E000 – 0x400C_EFFF	HBI_BA	HBI Control Registers
0x400D_0000 – 0x400D_0FFF	USCI0_BA	USCI0 Control Registers
System Controllers Space (0xE000_E000 ~ 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SCS_BA	System Timer Control Registers
0xE000_E100 – 0xE000_ECFF	SCS_BA	External Interrupt Controller Control Registers

0xE000_ED00 – 0xE000_ED8F	SCS_BA	System Control Registers

Table 6.2-7 Address Space Assignments for On-Chip Controllers

6.2.7 SRAM Memory Organization

The M463/M467 series supports embedded SRAM with total up to 512 Kbytes size and the SRAM organization is separated to three banks: SRAM bank0, SRAM bank1 and SRAM bank2. The first bank has 128 Kbytes address space, the second bank has 128 Kbyte address space and the third bank has 256 Kbyte address space. These three banks address space can be accessed simultaneously. The SRAM bank0 first 64 Kbyte supports parity error check at 512 Kbytes device and all 32 Kbytes at 128 Kbytes device to make sure chip operating more stable.

- Supports total up to 512 Kbytes SRAM
- Supports byte / half word / word write
- Supports parity error check function for SRAM bank0 first 64 Kbytes at 512 Kbytes device
- Supports parity error check function for SRAM bank0 all 32 Kbytes at 128 Kbytes device
- Supports oversize response error
- Supports remap address to 0x1000_0000

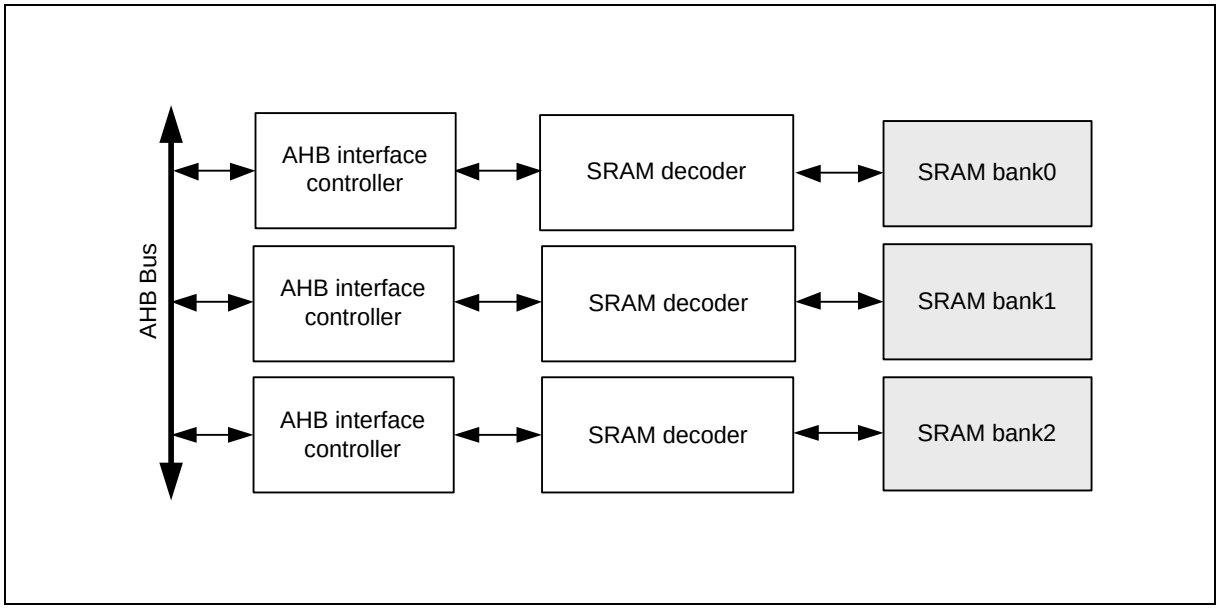


Figure 6.2-10 SRAM Block Diagram

Figure 6.2-10 shows the SRAM organization of M463/M467 series. There are three SRAM banks in M463/M467 series. The bank0 is addressed to 128 Kbytes, bank1 is addressed to 128 Kbytes and the bank2 is addressed to 256 Kbytes. The bank0 address space is from 0x2000_0000 to 0x2001_FFFF. The bank1 address space is from 0x2002_0000 to 0x2003_FFFF. The bank2 address space is from 0x2004_0000 to 0x2007_FFFF. The address between 0x2008_0000 to 0x3FFF_FFFF is illegal memory space and chip will enter hardfault if CPU accesses these illegal memory addresses.

The address of each bank is remapping from 0x2000_0000 to 0x1000_0000. CPU can access SRAM bank0 through 0x2000_0000 to 0x2001_FFFF or 0x1000_0000 to 0x1001_FFFF, and access SRAM bank1 through 0x2002_0000 to 0x2003_FFFF or 0x1002_0000 to 0x1003_FFFF and access SRAM bank2 through 0x2004_0000 to 0x2007_FFFF or 0x1004_0000 to 0x1007_FFFF.

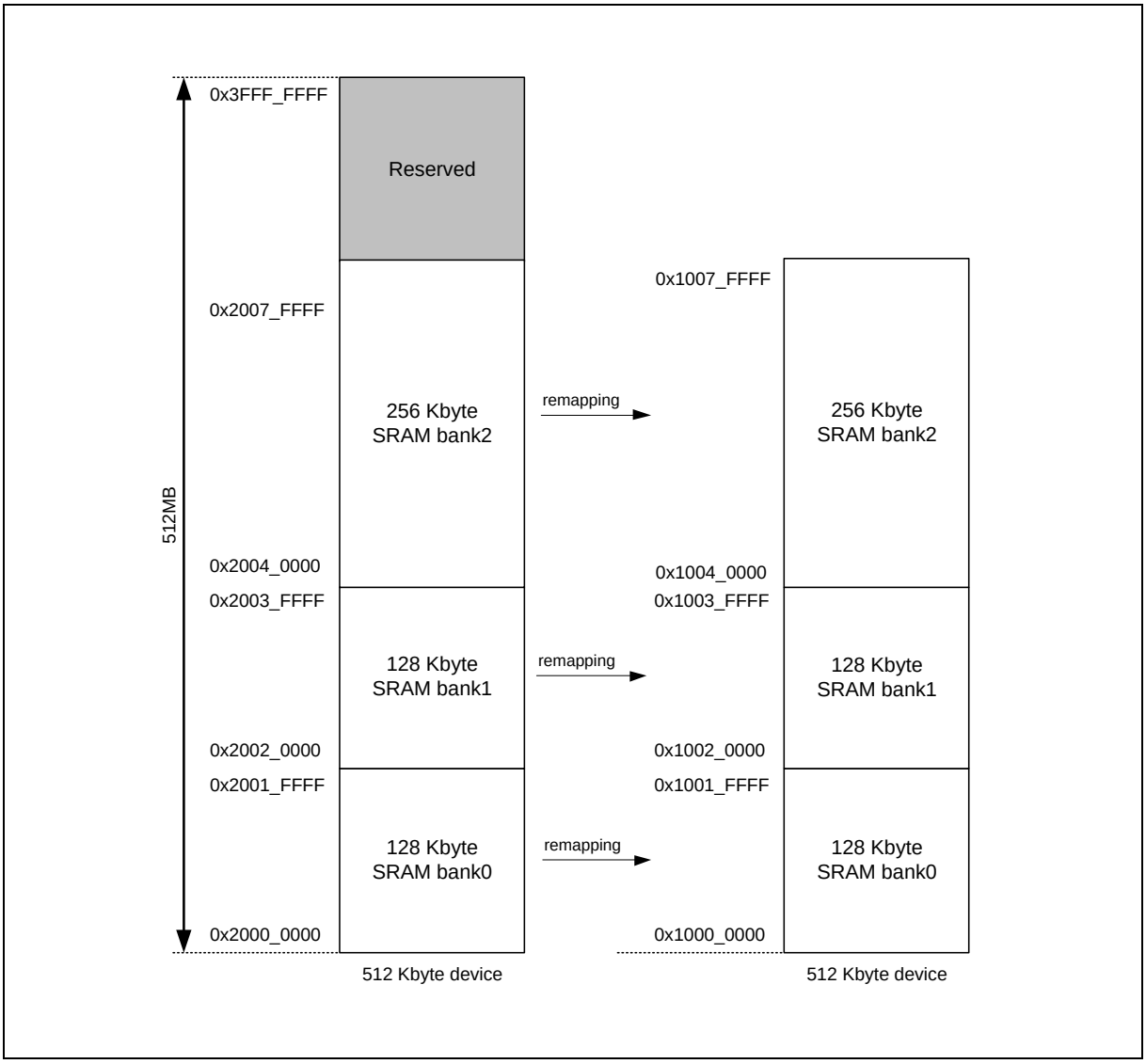


Figure 6.2-11 SRAM Memory Organization for 512 Kbyte Device

For 512 Kbytes device, SRAM bank0 first 64 Kbytes has byte parity error check function. When CPU is accessing the first 64 Kbytes of SRAM bank0, the parity error checking mechanism is dynamic operating. As parity error occurred, the PERRIF (SYS_SRAM_STATUS[0]) will be asserted to 1 and the SYS_SRAM_ERRADDR register will recode the address with parity error. Chip will enter interrupt when SRAM parity error occurred if PERRIEN (SYS_SRAM_INTCTL[0]) is set to 1. When SRAM parity error occurred, chip will stop detecting SRAM parity error until user writes 1 to clear the PERRIF(SYS_SRAM_STATUS[0]) bit.

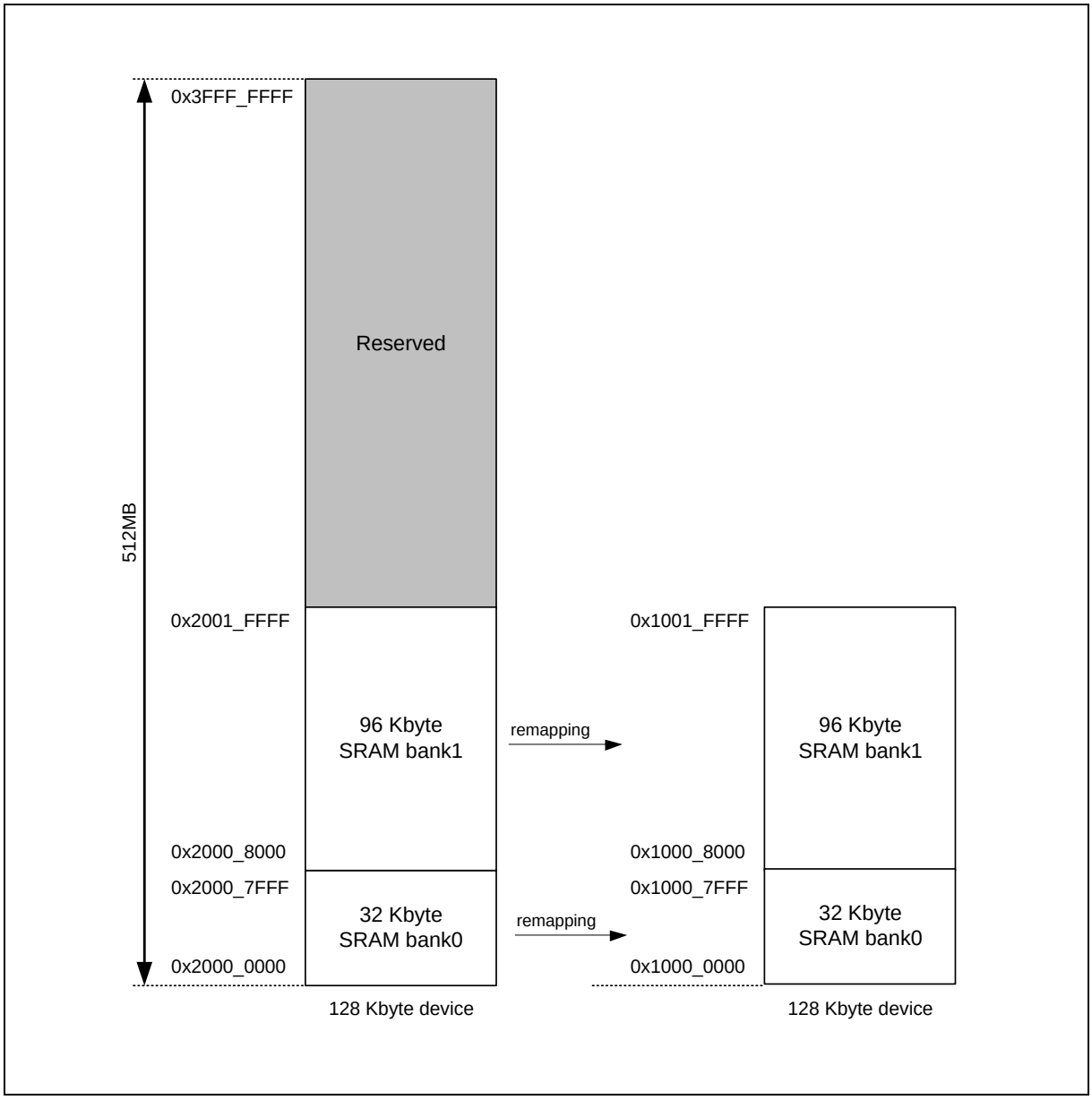


Figure 6.2-12 SRAM Memory Organization for 128 Kbyte Device

For 128 Kbytes device, SRAM bank0 all 32 Kbytes has byte parity error check function. When CPU is accessing 32 Kbytes of SRAM bank0, the parity error checking mechanism is dynamic operating. As parity error occurred, the PERRIF (SYS_SRAM_STATUS[0]) will be asserted to 1 and the SYS_SRAM_ERRADDR register will recode the address with parity error. Chip will enter interrupt when SRAM parity error occurred if PERRIEN (SYS_SRAM_INTCTL[0]) is set to 1. When SRAM parity error occurred, chip will stop detecting SRAM parity error until user writes 1 to clear the PERRIF(SYS_SRAM_STATUS[0]) bit.

6.2.8 Bus Matrix

The M463/M467 series supports Bus Matrix to manage the access arbitration between masters. The access arbitration can be selected by INTACTEN (SYS_AHBMCTL[0]) to use round-robin algorithm or set Cortex-M4 CPU as the highest bus priority.

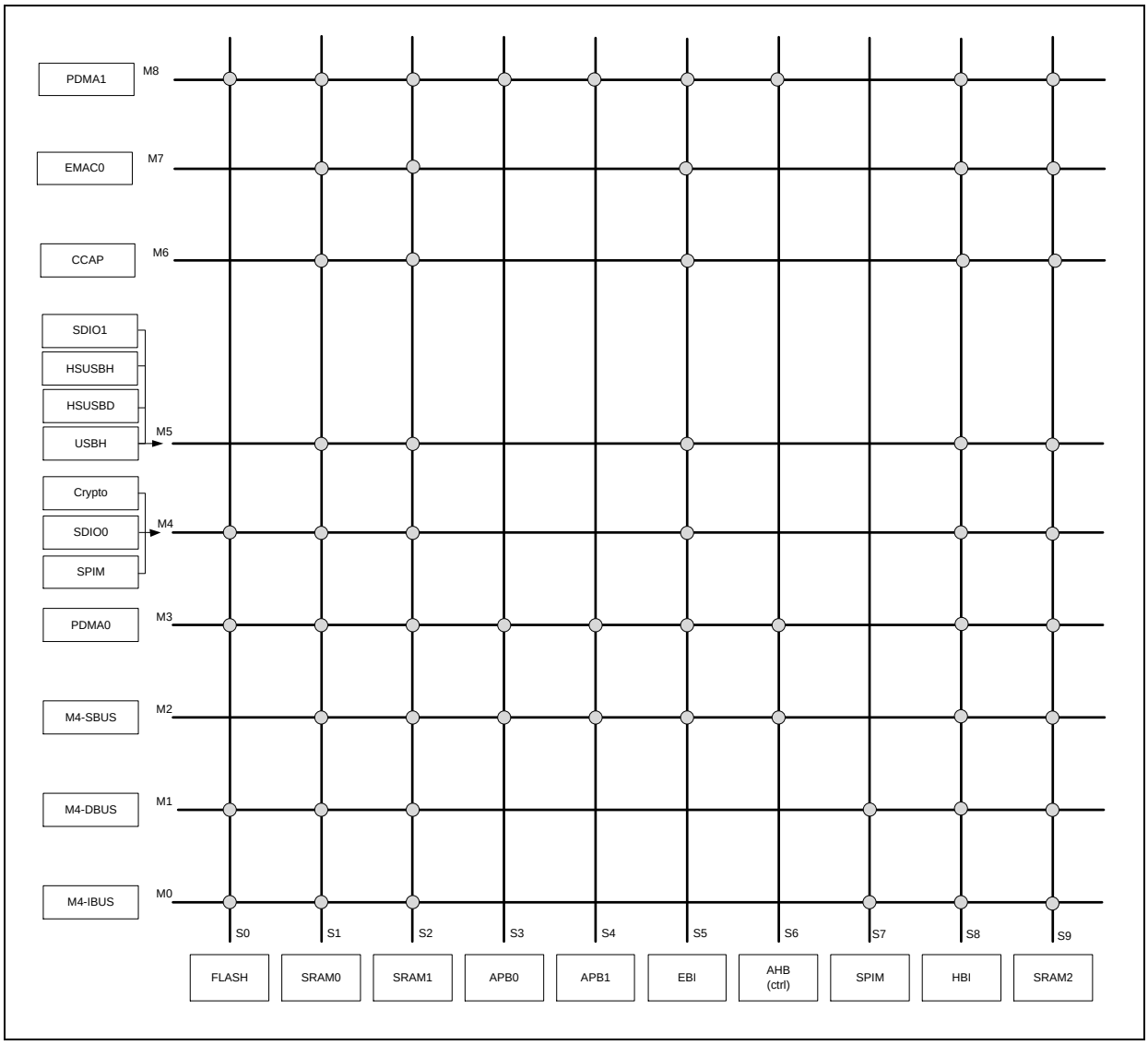


Figure 6.2-13 NuMicro M463/M467 Series Bus Matrix Diagram

6.2.9 HIRC Auto Trim

This chip supports auto-trim function: the HIRC trim (12 MHz RC oscillator) and HIRC48M trim (48 MHz RC oscillator), according to the accurate LXT (32.768 kHz crystal oscillator) or internal USB synchronous mode, automatically gets accurate HIRC output frequency, 0.25 % deviation within all temperature ranges.

For instance, the system needs an accurate 12 MHz clock. In such case, if neither using use PLL as the system clock source nor soldering 32.768 kHz crystal in system, user has to set REFCKSEL (SYS_IRCTCTL[10] reference clock selection) to “1”, set FREQSEL (SYS_IRCTCTL[1:0] trim frequency selection) to “01”, and the auto-trim function will be enabled. Interrupt status bit FREQLOCK (SYS_IRCTISTS[8] HIRC frequency lock status) “1” indicates the HIRC output frequency is accurate within 0.25% deviation.

In HIRC48M case, the system needs an accurate 48 MHz clock. In such case, if neither using use PLL as the system clock source nor soldering 32.768 kHz crystal in system, user has to set REFCKSEL (SYS_HIRCTCTL[10] reference clock selection) to “1”, set FREQSEL (SYS_HIRCTCTL[1:0] trim frequency selection) to “10”, and the auto-trim function will be enabled. Interrupt status bit FREQLOCK (SYS_HIRCTSTS[8] HIRC frequency lock status) “1” indicates the HIRC output frequency is accurate within 0.25% deviation.

6.2.10 Register Lock Control

Some of the system control registers need to be protected to avoid inadvertent write and disturb the chip operation. These system control registers are protected after the power-on reset till user to disable register protection. For user to program these protected registers, a register protection disable sequence needs to be followed by a special programming. The register protection disable sequence is writing the data “59h”, “16h” “88h” to the register SYS_REGLCTL address at 0x4000_0100 continuously. Any different data value, different sequence or any other write to other address during these three data writing will abort the whole sequence.

SYS_IPRST0	**Address 0x4000_0008**
SYS_BODCTL	address 0x4000_0018
SYS_PORCTL	address 0x4000_0024
SYS_VREFCTL	address 0x4000_0028
SYS_USBPHY	address 0x4000_002C
SYS_SRAM_BISTCTL	address 0x4000_00D0
SYS_PORDISAN	address 0x4000_01EC
SYS_PLCTL	address 0x4000_01F8
CLK_PWRCTL	address 0x4000_0200
CLK_APBCLK0	address 0x4000_0208
CLK_CLKSEL0	address 0x4000_0210
CLK_CLKSEL1	address 0x4000_0214
CLK_PLLCTL	address 0x4000_0240
CLK_PLLFNCTL0	address 0x4000_0248
CLK_PLLFNCTL1	address 0x4000_024C
CLK_CLKDSTS	address 0x4000_0274
CLK_PMUCTL	address 0x4000_0290
NMIEN	address 0x4000_0300
AHBMCTL	address 0x4000_0400
FMC_ISPCTL	address 0x4000_C000
FMC_ISPTRG	address 0x4000_C010
FMC_ISPSTS	address 0x4000_C040
FMC_CYCCTL	address 0x4000_C04C
WDT_CTL	address 0x4004_0000
WDT_ALTCTL	address 0x4004_0004

TIMER0_CTL	address 0x4005_0000
TIMER1_CTL	address 0x4005_0100
TIMER2_CTL	address 0x4005_1000
TIMER3_CTL	address 0x4005_1100
TIMER0_PWMCTL	address 0x4005_0040
TIMER1_PWMCTL	address 0x4005_0140
TIMER2_PWMCTL	address 0x4005_1040
TIMER3_PWMCTL	address 0x4005_1140
TIMER0_PWMDTCTL	address 0x4005_0058
TIMER1_PWMDTCTL	address 0x4005_0158
TIMER2_PWMDTCTL	address 0x4005_1058
TIMER3_PWMDTCTL	address 0x4005_1158
TIMER0_PWMBRKCTL	address 0x4005_0070
TIMER1_PWMBRKCTL	address 0x4005_0170
TIMER2_PWMBRKCTL	address 0x4005_1070
TIMER3_PWMBRKCTL	address 0x4005_1170
TIMER0_PWMSWBRK	address 0x4005_007C
TIMER1_PWMSWBRK	address 0x4005_017C
TIMER2_PWMSWBRK	address 0x4005_107C
TIMER3_PWMSWBRK	address 0x4005_117C
TIMER0_PWMINTEN1	address 0x4005_0084
TIMER1_PWMINTEN1	address 0x4005_0184
TIMER2_PWMINTEN1	address 0x4005_1084
TIMER3_PWMINTEN1	address 0x4005_1184
TIMER0_PWMINTSTS1	address 0x4005_008C
TIMER1_PWMINTSTS1	address 0x4005_018C
TIMER2_PWMINTSTS1	address 0x4005_108C
TIMER3_PWMINTSTS1	address 0x4005_118C
EPWM_CTL0	address 0x4005_8000/0x4005_9000
EPWM_CTL1	address 0x4005_8000/0x4005_9000
EPWM_DTCTL0_1	address 0x4005_8070/0x4005_9070
EPWM_DTCTL2_3	address 0x4005_8074/0x4005_9074
EPWM_DTCTL4_5	address 0x4005_8078/0x4005_9078
EPWM_BRKCTL0_1	address 0x4005_80C8/0x4005_90C8
EPWM_BRKCTL2_3	address 0x4005_80CC/0x4005_90CC

EPWM_BRKCTL4_5	address 0x4005_80D0/0x4005_90D0
EPWM_SWBRK	address 0x4005_80DC/0x4005_90DC
EPWM_INTEN1	address 0x4005_80E4/0x4005_90E4
EPWM_INTSTS1	address 0x4005_80EC/0x4005_90EC
BPWM_CTL0	address 0x4005_A000/0x4005_B000

6.2.11 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SYS Base Address: **SYS_BA = 0x4000_0000**				
SYS_PDID	SYS_BA+0x00	R	Part Device Identification Number Register	0x0045_1A93 [1]
SYS_RSTSTS	SYS_BA+0x04	R/W	System Reset Status Register	0x0000_0043
SYS_IPRST0	SYS_BA+0x08	R/W	Peripheral Reset Control Register 0	0x0000_0000
SYS_IPRST1	SYS_BA+0x0C	R/W	Peripheral Reset Control Register 1	0x0000_0000
SYS_IPRST2	SYS_BA+0x10	R/W	Peripheral Reset Control Register 2	0x0000_0000
SYS_BODCTL	SYS_BA+0x18	R/W	Brown-out Detector Control Register	0x000X_838X
SYS_IVSCTL	SYS_BA+0x1C	R/W	Internal Voltage Source Control Register	0x0000_0000
SYS_IPRST3	SYS_BA+0x20	R/W	Peripheral Reset Control Register 3	0x0000_0000
SYS_PORCTL	SYS_BA+0x24	R/W	Power-On-reset Controller Register	0x0000_0000
SYS_VREFCTL	SYS_BA+0x28	R/W	V_{REF} Control Register	0x0000_0200
SYS_USBPHY	SYS_BA+0x2C	R/W	USB PHY Control Register	0x0003_0007
SYS_GPA_MFOS	SYS_BA+0x80	R/W	GPIOA Multiple Function Output Select Register	0x0000_0000
SYS_GPB_MFOS	SYS_BA+0x84	R/W	GPIOB Multiple Function Output Select Register	0x0000_0000
SYS_GPC_MFOS	SYS_BA+0x88	R/W	GPIOC Multiple Function Output Select Register	0x0000_0000
SYS_GPD_MFOS	SYS_BA+0x8C	R/W	GPIOD Multiple Function Output Select Register	0x0000_0000
SYS_GPE_MFOS	SYS_BA+0x90	R/W	GPIOE Multiple Function Output Select Register	0x0000_0000
SYS_GPF_MFOS	SYS_BA+0x94	R/W	GPIOF Multiple Function Output Select Register	0x0000_0000
SYS_GPG_MFOS	SYS_BA+0x98	R/W	GPIOG Multiple Function Output Select Register	0x0000_0000
SYS_GPH_MFOS	SYS_BA+0x9C	R/W	GPIOH Multiple Function Output Select Register	0x0000_0000
SYS_GPI_MFOS	SYS_BA+0xA0	R/W	GPIOI Multiple Function Output Select Register	0x0000_0000
SYS_GPJ_MFOS	SYS_BA+0xA4	R/W	GPIOJ Multiple Function Output Select Register	0x0000_0000
SYS_SRAM_INTCTL	SYS_BA+0xC0	R/W	System SRAM Interrupt Enable Control Register	0x0000_0000
SYS_SRAM_STATUS	SYS_BA+0xC4	R/W	System SRAM Parity Error Status Register	0x0000_0000
SYS_SRAM_ERRADDR	SYS_BA+0xC8	R	System SRAM Parity Check Error Address Register	0x0000_0000
SYS_SRAM_BISTCTL	SYS_BA+0xD0	R/W	System SRAM BIST Test Control Register	0x0000_0000
SYS_SRAM_BISTSTS	SYS_BA+0xD4	R	System SRAM BIST Test Status Register	0x00XX_00XX
SYS_HIRCTCTL	SYS_BA+0xE4	R/W	HIRC48M Trim Control Register	0x0008_0000

SYS_HIRCTIEN	SYS_BA+0xE8	R/W	HIRC48M Trim Interrupt Enable Register	0x0000_0000
SYS_HIRCTISTS	SYS_BA+0xEC	R/W	HIRC48M Trim Interrupt Status Register	0x0000_0000
SYS_IRCTCTL	SYS_BA+0xF0	R/W	HIRC Trim Control Register	0x0008_0000
SYS_IRCTIEN	SYS_BA+0xF4	R/W	HIRC Trim Interrupt Enable Register	0x0000_0000
SYS_IRCTISTS	SYS_BA+0xF8	R/W	HIRC Trim Interrupt Status Register	0x0000_0000
SYS_REGLCTL	SYS_BA+0x100	R/W	Register Lock Control Register	0x0000_0000
SYS_PORDISAN	SYS_BA+0x1EC	R/W	Analog POR Disable Control Register	0x0000_0000
SYS_CSERVER	SYS_BA+0x1F4	R	Chip Series Version Register	0x0000_00XX
SYS_PLCTL	SYS_BA+0x1F8	R/W	Power Level Control Register	0x0000_0000
SYS_PLSTS	SYS_BA+0x1FC	R	Power Level Status Register	0x0000_0000
SYS_AHBMCTL	SYS_BA+0x400	R/W	AHB Bus Matrix Priority Control Register	0x0000_0001
SYS_GPA_MFP0	SYS_BA+0x500	R/W	GPIOA Multiple Function Control Register 0	0x0000_0000
SYS_GPA_MFP1	SYS_BA+0x504	R/W	GPIOA Multiple Function Control Register 1	0x0000_0000
SYS_GPA_MFP2	SYS_BA+0x508	R/W	GPIOA Multiple Function Control Register 2	0x0000_0000
SYS_GPA_MFP3	SYS_BA+0x50C	R/W	GPIOA Multiple Function Control Register 3	0x0000_0000
SYS_GPB_MFP0	SYS_BA+0x510	R/W	GPIOB Multiple Function Control Register 0	0x0000_0000
SYS_GPB_MFP1	SYS_BA+0x514	R/W	GPIOB Multiple Function Control Register 1	0x0000_0000
SYS_GPB_MFP2	SYS_BA+0x518	R/W	GPIOB Multiple Function Control Register 2	0x0000_0000
SYS_GPB_MFP3	SYS_BA+0x51C	R/W	GPIOB Multiple Function Control Register 3	0x0000_0000
SYS_GPC_MFP0	SYS_BA+0x520	R/W	GPIOC Multiple Function Control Register 0	0x0000_0000
SYS_GPC_MFP1	SYS_BA+0x524	R/W	GPIOC Multiple Function Control Register 1	0x0000_0000
SYS_GPC_MFP2	SYS_BA+0x528	R/W	GPIOC Multiple Function Control Register 2	0x0000_0000
SYS_GPC_MFP3	SYS_BA+0x52C	R/W	GPIOC Multiple Function Control Register 3	0x0000_0000
SYS_GPD_MFP0	SYS_BA+0x530	R/W	GPIOD Multiple Function Control Register 0	0x0000_0000
SYS_GPD_MFP1	SYS_BA+0x534	R/W	GPIOD Multiple Function Control Register 1	0x0000_0000
SYS_GPD_MFP2	SYS_BA+0x538	R/W	GPIOD Multiple Function Control Register 2	0x0000_0000
SYS_GPD_MFP3	SYS_BA+0x53C	R/W	GPIOD Multiple Function Control Register 3	0x0000_0000
SYS_GPE_MFP0	SYS_BA+0x540	R/W	GPIOE Multiple Function Control Register 0	0x0000_0000
SYS_GPE_MFP1	SYS_BA+0x544	R/W	GPIOE Multiple Function Control Register 1	0x0000_0000
SYS_GPE_MFP2	SYS_BA+0x548	R/W	GPIOE Multiple Function Control Register 2	0x0000_0000
SYS_GPE_MFP3	SYS_BA+0x54C	R/W	GPIOE Multiple Function Control Register 3	0x0000_0000

SYS_GPF_MFP0	SYS_BA+0x550	R/W	GPIOF Multiple Function Control Register 0	0x0000_0E0E
SYS_GPF_MFP1	SYS_BA+0x554	R/W	GPIOF Multiple Function Control Register 1	0x0000_0000
SYS_GPF_MFP2	SYS_BA+0x558	R/W	GPIOF Multiple Function Control Register 2	0x0000_0000
SYS_GPF_MFP3	SYS_BA+0x55C	R/W	GPIOF Multiple Function Control Register 3	0x0000_0000
SYS_GPG_MFP0	SYS_BA+0x560	R/W	GPIOG Multiple Function Control Register 0	0x0000_0000
SYS_GPG_MFP1	SYS_BA+0x564	R/W	GPIOG Multiple Function Control Register 1	0x0000_0000
SYS_GPG_MFP2	SYS_BA+0x568	R/W	GPIOG Multiple Function Control Register 2	0x0000_0000
SYS_GPG_MFP3	SYS_BA+0x56C	R/W	GPIOG Multiple Function Control Register 3	0x0000_0000
SYS_GPH_MFP0	SYS_BA+0x570	R/W	GPIOH Multiple Function Control Register 0	0x0000_0000
SYS_GPH_MFP1	SYS_BA+0x574	R/W	GPIOH Multiple Function Control Register 1	0x0000_0000
SYS_GPH_MFP2	SYS_BA+0x578	R/W	GPIOH Multiple Function Control Register 2	0x0000_0000
SYS_GPH_MFP3	SYS_BA+0x57C	R/W	GPIOH Multiple Function Control Register 3	0x0000_0000
SYS_GPI_MFP0	SYS_BA+0x580	R/W	GPIOI Multiple Function Control Register 0	0x0000_0000
SYS_GPI_MFP1	SYS_BA+0x584	R/W	GPIOI Multiple Function Control Register 1	0x0000_0000
SYS_GPI_MFP2	SYS_BA+0x588	R/W	GPIOI Multiple Function Control Register 2	0x0000_0000
SYS_GPI_MFP3	SYS_BA+0x58C	R/W	GPIOI Multiple Function Control Register 3	0x0000_0000
SYS_GPJ_MFP0	SYS_BA+0x590	R/W	GPIOJ Multiple Function Control Register 0	0x0000_0000
SYS_GPJ_MFP1	SYS_BA+0x594	R/W	GPIOJ Multiple Function Control Register 1	0x0000_0000
SYS_GPJ_MFP2	SYS_BA+0x598	R/W	GPIOJ Multiple Function Control Register 2	0x0000_0000
SYS_GPJ_MFP3	SYS_BA+0x59C	R/W	GPIOJ Multiple Function Control Register 3	0x0000_0000

6.2.12 Register Description

Part Device Identification Number Register (SYS_PDID)

Register	Offset	R/W	Description	Reset Value
SYS_PDID	SYS_BA+0x00	R	Part Device Identification Number Register	0x0045_1A93 [1]

[1] Every part number has a unique default reset value.

31	30	29	28	27	26	25	24
PDID							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PDID							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PDID							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDID							

Bits	Description	
[31:0]	**PDID**	**Part Device Identification Number (Read Only)** This register reflects device part number code. Software can read this register to identify which device is used.

System Reset Status Register (SYS_RSTSTS)

This register provides specific information for software to identify this chip's reset source from last operation.

Register	Offset	R/W	Description	Reset Value
SYS_RSTSTS	SYS_BA+0x04	R/W	System Reset Status Register	0x0000_0043

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							CPULKRF
7	6	5	4	3	2	1	0
CPURF	HRESETRF	MCURF	BODRF	LVRF	WDTRF	PINRF	PORF

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**CPULKRF**	**CPU Lockup Reset Flag** 0 = No reset from CPU lockup happened. 1 = The Cortex-M4 lockup happened and chip is reset. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** When CPU lockup happened under ICE is connected, this flag will set to 1 but chip will not reset.
[7]	**CPURF**	**CPU Reset Flag** The CPU reset flag is set by hardware if software writes CPURST (SYS_IPRST0[1]) 1 to reset Cortex-M4 Core and Flash Memory Controller (FMC). 0 = No reset from CPU. 1 = The Cortex-M4 Core and FMC are reset by software setting CPURST to 1. **Note:** Write 1 to clear this bit to 0.
[6]	**HRESETRF**	**HRESET Reset Flag** The HRESET reset flag is set by the "Reset Signal" from the HRESET. 0 = No reset from HRESET. 1 = Reset from HRESET. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** HRESET includes: POR, Reset Pin, LVR, BOD, WDT, WWDT, CPU lockup, CHIP and MCU reset.

Bits	Description	
[5]	**MCURF**	**MCU Reset Flag** The MCU reset flag is set by the “Reset Signal” from the Cortex-M4 Core to indicate the previous reset source. 0 = No reset from Cortex-M4. 1 = The Cortex-M4 had issued the reset signal to reset the system by writing 1 to the bit SYSRESETREQ(AIRCR[2], Application Interrupt and Reset Control Register, address = 0xE000ED0C) in system control registers of Cortex-M4 core. **Note:** Write 1 to clear this bit to 0.
[4]	**BODRF**	**BOD Reset Flag** The BOD reset flag is set by the “Reset Signal” from the Brown-Out Detector to indicate the previous reset source. 0 = No reset from BOD. 1 = The BOD had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[3]	**LVRF**	**LVR Reset Flag** The LVR reset flag is set by the “Reset Signal” from the Low Voltage Reset Controller to indicate the previous reset source. 0 = No reset from LVR. 1 = LVR controller had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[2]	**WDTRF**	**WDT Reset Flag** The WDT reset flag is set by the “Reset Signal” from the Watchdog Timer or Window Watchdog Timer to indicate the previous reset source. 0 = No reset from watchdog timer or window watchdog timer. 1 = The watchdog timer or window watchdog timer had issued the reset signal to reset the system. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** Watchdog Timer register RSTF(WDT_CTL[2]) bit is set if the system has been reset by WDT time-out reset. Window Watchdog Timer register WWDTRF(WWDT_STATUS[1]) bit is set if the system has been reset by WWDT time-out reset.
[1]	**PINRF**	**NRESET Pin Reset Flag** The nRESET pin reset flag is set by the “Reset Signal” from the nRESET pin to indicate the previous reset source. 0 = No reset from nRESET pin. 1 = Pin nRESET had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[0]	**PORF**	**POR Reset Flag** The POR reset flag is set by the “Reset Signal” from the Power-on Reset (POR) Controller or bit CHIPRST (SYS_IPRST0[0]) to indicate the previous reset source. 0 = No reset from POR or CHIPRST. 1 = Power-on Reset (POR) or CHIPRST had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.

Peripheral Reset Control Register 0 (SYS_IPRST0)

Register	Offset	R/W	Description	Reset Value
SYS_IPRST0	SYS_BA+0x08	R/W	Peripheral Reset Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CANFD3RST	CANFD2RST	CANFD1RST	CANFD0RST	Reserved	PDMA1RST	SDH1RST	HSUSBHRST
15	14	13	12	11	10	9	8
Reserved	SPIMRST	KSRST	CRPTRST	HBIRST	HSUSBDRST	Reserved	CCAPRST
7	6	5	4	3	2	1	0
CRCRST	SDH0RST	EMAC0RST	Reserved	EBIRST	PDMA0RST	CPURST	CHIPRST

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**CANFD3RST**	**CANFD3 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the CANFD3 controller. User needs to set this bit to 0 to release from reset state. 0 = CANFD3 controller normal operation. 1 = CANFD3 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[22]	**CANFD2RST**	**CANFD2 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the CANFD2 controller. User needs to set this bit to 0 to release from reset state. 0 = CANFD2 controller normal operation. 1 = CANFD2 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[21]	**CANFD1RST**	**CANFD1 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the CANFD1 controller. User needs to set this bit to 0 to release from reset state. 0 = CANFD1 controller normal operation. 1 = CANFD1 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[20]	**CANFD0RST**	**CANFD0 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the CANFD0 controller. User needs to set this bit to 0 to release from reset state. 0 = CANFD0 controller normal operation. 1 = CANFD0 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[19]	**Reserved**	Reserved.
[18]	**PDMA1RST**	**PDMA1 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the PDMA1 controller. User needs to set

		this bit to 0 to release from reset state. 0 = PDMA1 controller normal operation. 1 = PDMA1 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[17]	**SDH1RST**	**SDH1 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the SDH1 controller. User needs to set this bit to 0 to release from the reset state. 0 = SDH1 controller normal operation. 1 = SDH1 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[16]	**HSUSBHRST**	**HSUSBH Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the HSUSBH controller. User needs to set this bit to 0 to release from the reset state. 0 = HSUSBH controller normal operation. 1 = HSUSBH controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[15]	**Reserved**	Reserved.
[14]	**SPIMRST**	**SPIM Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the SPIM controller. User needs to set this bit to 0 to release from the reset state. 0 = SPIM controller normal operation. 1 = SPIM controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[13]	**KSRST**	**Key Store Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the Key Store controller. User needs to set this bit to 0 to release from the reset state. 0 = Key Store controller normal operation. 1 = Key Store controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[12]	**CRPTRST**	**CRYPTO Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the CRYPTO controller. User needs to set this bit to 0 to release from the reset state. 0 = CRYPTO controller normal operation. 1 = CRYPTO controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11]	**HBIRST**	**HBI Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the HBI controller. User needs to set this bit to 0 to release from the reset state. 0 = HBI controller normal operation. 1 = HBI controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[10]	**HSUSBDRST**	**HSUSBD Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the HSUSBD controller. User needs to set this bit to 0 to release from the reset state. 0 = HSUSBD controller normal operation. 1 = HSUSBD controller reset.
[9]	**Reserved**	Reserved.

[8]	**CCAPRST**	**CCAP Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the CCAP controller. User needs to set this bit to 0 to release from the reset state. 0 = CCAP controller normal operation. 1 = CCAP controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	**CRCRST**	**CRC Calculation Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the CRC calculation controller. User needs to set this bit to 0 to release from the reset state. 0 = CRC calculation controller normal operation. 1 = CRC calculation controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**SDH0RST**	**SDH0 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the SDH0 controller. User needs to set this bit to 0 to release from the reset state. 0 = SDH0 controller normal operation. 1 = SDH0 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**EMA0CRST**	**EMAC0 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the EMAC0 controller. User needs to set this bit to 0 to release from the reset state. 0 = EMAC0 controller normal operation. 1 = EMAC0 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**Reserved**	Reserved.
[3]	**EBIRST**	**EBI Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the EBI controller. User needs to set this bit to 0 to release from the reset state. 0 = EBI controller normal operation. 1 = EBI controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**PDMA0RST**	**PDMA0 Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the PDMA0 controller. User needs to set this bit to 0 to release from reset state. 0 = PDMA0 controller normal operation. 1 = PDMA0 controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[1]	**CPURST**	**Processor Core One-shot Reset (Write Protect)** Setting this bit will only reset the processor core and Flash Memory Controller(FMC), and this bit will automatically return to 0 after the 2 clock cycles. 0 = Processor core normal operation. 1 = Processor core one-shot reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**CHIPRST**	**Chip One-shot Reset (Write Protect)** Setting this bit will reset the whole chip, including Processor core and all peripherals, and this bit will automatically return to 0 after the 2 clock cycles. The CHIPRST is same as the POR reset, all the chip controllers are reset and the chip setting from Flash are also reload. About the difference between CHIPRST and SYSRESETREQ(AIRCR[2]), please refer to

		System Reset section. 0 = Chip normal operation. 1 = Chip one-shot reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Peripheral Reset Control Register 1 (SYS_IPRST1)

Setting these bits 1 will generate asynchronous reset signals to the corresponding module controller. Users need to set these bits to 0 to release corresponding module controller from reset state.

Register	Offset	R/W	Description	Reset Value
SYS_IPRST1	SYS_BA+0x0C	R/W	Peripheral Reset Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
TRNGRST	HSOTGRST	I2S0RST	EADC0RST	USBDRST	OTGRST	Reserved	Reserved
23	22	21	20	19	18	17	16
UART7RST	UART6RST	UART5RST	UART4RST	UART3RST	UART2RST	UART1RST	UART0RST
15	14	13	12	11	10	9	8
SPI2RST	SPI1RST	SPI0RST	QSPI0RST	I2C3RST	I2C2RST	I2C1RST	I2C0RST
7	6	5	4	3	2	1	0
ACMP01RST	Reserved	TMR3RST	TMR2RST	TMR1RST	TMR0RST	GPIORST	Reserved

Bits	Description	
[31]	**TRNGRST**	**TRNG Controller Reset** 0 = TRNG controller normal operation. 1 = TRNG controller reset.
[30]	**HSOTGRST**	**HSOTG Controller Reset** 0 = HSOTG controller normal operation. 1 = HSOTG controller reset.
[29]	**I2S0RST**	**I2S0 Controller Reset** 0 = I2S0 controller normal operation. 1 = I2S0 controller reset.
[28]	**EADC0RST**	**EADC0 Controller Reset** 0 = EADC0 controller normal operation. 1 = EADC0 controller reset.
[27]	**USBDRST**	**USBD Controller Reset** 0 = USBD controller normal operation. 1 = USBD controller reset.
[26]	**OTGRST**	**OTG Controller Reset** 0 = OTG controller normal operation. 1 = OTG controller reset.
[25:24]	**Reserved**	Reserved.
[23]	**UART7RST**	**UART7 Controller Reset** 0 = UART7 controller normal operation. 1 = UART7 controller reset.
[22]	**UART6RST**	**UART6 Controller Reset** 0 = UART6 controller normal operation.

		1 = UART6 controller reset.
[21]	**UART5RST**	**UART5 Controller Reset** 0 = UART5 controller normal operation. 1 = UART5 controller reset.
[20]	**UART4RST**	**UART4 Controller Reset** 0 = UART4 controller normal operation. 1 = UART4 controller reset.
[19]	**UART3RST**	**UART3 Controller Reset** 0 = UART3 controller normal operation. 1 = UART3 controller reset.
[18]	**UART2RST**	**UART2 Controller Reset** 0 = UART2 controller normal operation. 1 = UART2 controller reset.
[17]	**UART1RST**	**UART1 Controller Reset** 0 = UART1 controller normal operation. 1 = UART1 controller reset.
[16]	**UART0RST**	**UART0 Controller Reset** 0 = UART0 controller normal operation. 1 = UART0 controller reset.
[15]	**SPI2RST**	**SPI2 Controller Reset** 0 = SPI2 controller normal operation. 1 = SPI2 controller reset.
[14]	**SPI1RST**	**SPI1 Controller Reset** 0 = SPI1 controller normal operation. 1 = SPI1 controller reset.
[13]	**SPI0RST**	**SPI0 Controller Reset** 0 = SPI0 controller normal operation. 1 = SPI0 controller reset.
[12]	**QSPI0RST**	**QSPI0 Controller Reset** 0 = QSPI0 controller normal operation. 1 = QSPI0 controller reset.
[11]	**I2C3RST**	**I2C3 Controller Reset** 0 = I2C3 controller normal operation. 1 = I2C3 controller reset.
[10]	**I2C2RST**	**I2C2 Controller Reset** 0 = I2C2 controller normal operation. 1 = I2C2 controller reset.
[9]	**I2C1RST**	**I2C1 Controller Reset** 0 = I2C1 controller normal operation. 1 = I2C1 controller reset.
[8]	**I2C0RST**	**I2C0 Controller Reset** 0 = I2C0 controller normal operation. 1 = I2C0 controller reset.

[7]	**ACMP01RST**	**Analog Comparator 0/1 Controller Reset** 0 = Analog Comparator 0/1 controller normal operation. 1 = Analog Comparator 0/1 controller reset.
[6]	**Reserved**	Reserved.
[5]	**TMR3RST**	**Timer3 Controller Reset** 0 = Timer3 controller normal operation. 1 = Timer3 controller reset.
[4]	**TMR2RST**	**Timer2 Controller Reset** 0 = Timer2 controller normal operation. 1 = Timer2 controller reset.
[3]	**TMR1RST**	**Timer1 Controller Reset** 0 = Timer1 controller normal operation. 1 = Timer1 controller reset.
[2]	**TMR0RST**	**Timer0 Controller Reset** 0 = Timer0 controller normal operation. 1 = Timer0 controller reset.
[1]	**GPIORST**	**GPIO Controller Reset** 0 = GPIO controller normal operation. 1 = GPIO controller reset.
[0]	**Reserved**	Reserved.

Peripheral Reset Control Register 2 (SYS_IPRST2)

Setting these bits to 1 will generate asynchronous reset signals to the corresponding module controller. Users need to set these bits to 0 to release corresponding module controller from reset state.

Register	Offset	R/W	Description	Reset Value
SYS_IPRST2	SYS_BA+0x10	R/W	Peripheral Reset Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
EADC1RST	Reserved	I2S1RST	Reserved	ECAP1RST	ECAP0RST	Reserved	
23	22	21	20	19	18	17	16
EQEI1RST	EQEI0RST	EQEI3RST	EQEI2RST	BPWM1RST	BPWM0RST	EPWM1RST	EPWM0RST
15	14	13	12	11	10	9	8
Reserved	ECAP3RST	ECAP2RST	DACRST	Reserved	PSIORST	Reserved	USCI0RST
7	6	5	4	3	2	1	0
SPI4RST	SPI3RST	Reserved	QSPI1RST	I2C4RST	SC2RST	SC1RST	SC0RST

Bits	Description	
[31]	**EADC1RST**	**EADC1 Controller Reset** 0 = EADC1 controller normal operation. 1 = EADC1 controller reset.
[30]	**Reserved**	Reserved.
[29]	**I2S1RST**	**I2S1 Controller Reset** 0 = I2S1 controller normal operation. 1 = I2S1 controller reset.
[28]	**Reserved**	Reserved.
[27]	**ECAP1RST**	**ECAP1 Controller Reset** 0 = ECAP1 controller normal operation. 1 = ECAP1 controller reset.
[26]	**ECAP0RST**	**ECAP0 Controller Reset** 0 = ECAP0 controller normal operation. 1 = ECAP0 controller reset.
[25:24]	**Reserved**	Reserved.
[23]	**EQEI1RST**	**EQEI1 Controller Reset** 0 = EQEI1 controller normal operation. 1 = EQEI1 controller reset.
[22]	**EQEI0RST**	**EQEI0 Controller Reset** 0 = EQEI0 controller normal operation. 1 = EQEI0 controller reset.
[21]	**EQEI3RST**	**EQEI3 Controller Reset** 0 = EQEI3 controller normal operation.

		1 = EQEI3 controller reset.
[20]	**EQEI2RST**	**EQEI2 Controller Reset** 0 = EQEI2 controller normal operation. 1 = EQEI2 controller reset.
[19]	**BPWM1RST**	**BPWM1 Controller Reset** 0 = BPWM1 controller normal operation. 1 = BPWM1 controller reset.
[18]	**BPWM0RST**	**BPWM0 Controller Reset** 0 = BPWM0 controller normal operation. 1 = BPWM0 controller reset.
[17]	**EPWM1RST**	**EPWM1 Controller Reset** 0 = EPWM1 controller normal operation. 1 = EPWM1 controller reset.
[16]	**EPWM0RST**	**EPWM0 Controller Reset** 0 = EPWM0 controller normal operation. 1 = EPWM0 controller reset.
[15]	**Reserved**	Reserved.
[14]	**ECAP3RST**	**ECAP3 Controller Reset** 0 = ECAP3 controller normal operation. 1 = ECAP3 controller reset.
[13]	**ECAP2RST**	**ECAP2 Controller Reset** 0 = ECAP2 controller normal operation. 1 = ECAP2 controller reset.
[12]	**DACRST**	**DAC Controller Reset** 0 = DAC controller normal operation. 1 = DAC controller reset.
[11]	**Reserved**	Reserved.
[10]	**PSIORST**	**PSIO Controller Reset** 0 = PSIO controller normal operation. 1 = PSIO controller reset.
[9]	**Reserved**	Reserved.
[8]	**USCI0RST**	**USCI0 Controller Reset** 0 = USCI0 controller normal operation. 1 = USCI0 controller reset.
[7]	**SPI4RST**	**SPI4 Controller Reset** 0 = SPI4 controller normal operation. 1 = SPI4 controller reset.
[6]	**SPI3RST**	**SPI3 Controller Reset** 0 = SPI3 controller normal operation. 1 = SPI3 controller reset.
[5]	**Reserved**	Reserved.
[4]	**QSPI1RST**	**QSPI1 Controller Reset** 0 = QSPI1 controller normal operation.

		1 = QSPI1 controller reset.
[3]	**I2C4RST**	**I2C4 Controller Reset** 0 = I2C4 controller normal operation. 1 = I2C4 controller reset.
[2]	**SC2RST**	**SC2 Controller Reset** 0 = SC2 controller normal operation. 1 = SC2 controller reset.
[1]	**SC1RST**	**SC1 Controller Reset** 0 = SC1 controller normal operation. 1 = SC1 controller reset.
[0]	**SC0RST**	**SC0 Controller Reset** 0 = SC0 controller normal operation. 1 = SC0 controller reset.

Brown-out Detector Control Register (SYS_BODCTL)

Partial of the SYS_BODCTL control registers values are initiated by the Flash configuration and partial bits are write-protected bit.

Register	Offset	R/W	Description	Reset Value
SYS_BODCTL	SYS_BA+0x18	R/W	Brown-out Detector Control Register	0x000X_838X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					BODVL		
15	14	13	12	11	10	9	8
LVRRDY	LVRDGSEL			Reserved	BODDGSEL		
7	6	5	4	3	2	1	0
LVREN	BODOUT	BODLPM	BODIF	BODRSTEN	Reserved		BODEN

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:16]	**BODVL**	**Brown-out Detector Threshold Voltage Selection (Write Protect)** The default value is set by Flash controller user configuration register CBOV (CONFIG0 [23:21]). 000 = Brown-Out Detector threshold voltage is 1.6V. 001 = Brown-Out Detector threshold voltage is 1.8V. 010 = Brown-Out Detector threshold voltage is 2.0V. 011 = Brown-Out Detector threshold voltage is 2.2V. 100 = Brown-Out Detector threshold voltage is 2.4V. 101 = Brown-Out Detector threshold voltage is 2.6V. 110 = Brown-Out Detector threshold voltage is 2.8V. 111 = Brown-Out Detector threshold voltage is 3.0V. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15]	**LVRRDY**	**Low Voltage Reset Ready Flag (Read Only)** When the LVR function first enable, need more HCLK to wait LVR ready. 0 = Low Voltage Reset function not ready. 1 = Low Voltage Reset function ready.

Bits	Description	
[14:12]	**LVRDGSEL**	**LVR Output De-glitch Time Select (Write Protect)** 000 = Without de-glitch function. 001 = 4 system clock (HCLK). 010 = 8 system clock (HCLK). 011 = 16 system clock (HCLK). 100 = 32 system clock (HCLK). 101 = 64 system clock (HCLK). 110 = 128 system clock (HCLK). 111 = 256 system clock (HCLK). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[11]	**Reserved**	Reserved.
[10:8]	**BODDGSEL**	**Brown-out Detector Output De-glitch Time Select (Write Protect)** 000 = BOD output is sampled by LIRC clock. 001 = 4 system clock (HCLK). 010 = 8 system clock (HCLK). 011 = 16 system clock (HCLK). 100 = 32 system clock (HCLK). 101 = 64 system clock (HCLK). 110 = 128 system clock (HCLK). 111 = 256 system clock (HCLK). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[7]	**LVREN**	**Low Voltage Reset Enable Bit (Write Protect)** The LVR function resets the chip when the input power voltage is lower than LVR circuit setting. LVR function is enabled by default. 0 = Low Voltage Reset function Disabled. 1 = Low Voltage Reset function Enabled. **Note 1:** After enabling the bit, the LVR function will be active with 100us ~ 200us delay for LVR output stable. LVRRDY(SYS_BODCTL[15]) is used to indicate LVR ready status. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**BODOUT**	**Brown-out Detector Output Status** 0 = Brown-out Detector output status is 0. It means the detected voltage is higher than BODVL setting or BODEN is 0. 1 = Brown-out Detector output status is 1. It means the detected voltage is lower than BODVL setting. If the BODEN is 0, BOD function disabled, this bit always responds 0.
[5]	**BODLPM**	**Brown-out Detector Low Power Mode (Write Protect)** 0 = BOD operate in normal mode (default). 1 = BOD Low Power mode Enabled. **Note 1:** The BOD consumes about 100uA in normal mode, the low power mode can reduce the current to about 1/10 but slow the BOD response. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.

Bits	Description	
[4]	**BODIF**	**Brown-out Detector Interrupt Flag** 0 = Brown-out Detector does not detect any voltage draft at AV_{DD} down through or up through the voltage of BODVL setting. 1 = When Brown-out Detector detects the AV_{DD} is dropped down through the voltage of BODVL setting or the AV_{DD} is raised up through the voltage of BODVL setting, this bit is set to 1 and the brown-out interrupt is requested if brown-out interrupt is enabled. **Note:** Write 1 to clear this bit to 0.
[3]	**BODRSTEN**	**Brown-out Reset Enable Bit (Write Protect)** The default value is set by Flash controller user configuration register CBORST(CONFIG0[20]) bit. 0 = Brown-out “INTERRUPT” function Enabled. 1 = Brown-out “RESET” function Enabled. **Note 1:** While the Brown-out Detector function is enabled (BODEN high) and BOD reset function is enabled (BODRSTEN high), BOD will assert a signal to reset chip when the detected voltage is lower than the threshold (BODOUT high). While the BOD function is enabled (BODEN high) and BOD interrupt function is enabled (BODRSTEN low), BOD will assert an interrupt if BODOUT is high. BOD interrupt will keep till to the BODEN set to 0. BOD interrupt can be blocked by disabling the NVIC BOD interrupt or disabling BOD function (set BODEN low). **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2:1]	**Reserved**	Reserved.
[0]	**BODEN**	**Brown-out Detector Enable Bit (Write Protect)** The default value is set by Flash controller user configuration register CBODEN (CONFIG0 [19]). 0 = Brown-out Detector function Disabled. 1 = Brown-out Detector function Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Internal Voltage Source Control Register (SYS_IVSCTL)

Register	Offset	R/W	Description	Reset Value
SYS_IVSCTL	SYS_BA+0x1C	R/W	Internal Voltage Source Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						VBATUGEN	VTEMPEN

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**VBATUGEN**	**V_{BAT} Unity Gain Buffer Enable Bit** This bit is used to enable/disable V_{BAT} unity gain buffer function. 0 = V_{BAT} unity gain buffer function Disabled (default). 1 = V_{BAT} unity gain buffer function Enabled. **Note:** After this bit is set to 1, the value of V_{BAT} unity gain buffer output voltage can be obtained from ADC conversion result.
[0]	**VTEMPEN**	**Temperature Sensor Enable Bit** This bit is used to enable/disable temperature sensor function. 0 = Temperature sensor function Disabled (default). 1 = Temperature sensor function Enabled.

Peripheral Reset Control Register 3 (SYS_IPRST3)

Setting these bits to 1 will generate asynchronous reset signals to the corresponding module controller. Users need to set these bits to 0 to release corresponding module controller from reset state.

Register	Offset	R/W	Description	Reset Value
SYS_IPRST3	SYS_BA+0x20	R/W	Peripheral Reset Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						UART9RST	UART8RST
15	14	13	12	11	10	9	8
Reserved		SPI10RST	SPI9RST	SPI8RST	SPI7RST	SPI6RST	SPI5RST
7	6	5	4	3	2	1	0
ACMP23RST	EADC2RST	Reserved					KPIRST

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**UART9RST**	**UART9 Controller Reset** 0 = UART9 controller normal operation. 1 = UART9 controller reset.
[16]	**UART8RST**	**UART8 Controller Reset** 0 = UART8 controller normal operation. 1 = UART8 controller reset.
[15:14]	**Reserved**	Reserved.
[13]	**SPI10RST**	**SPI10 Controller Reset** 0 = SPI10 controller normal operation. 1 = SPI10 controller reset.
[12]	**SPI9RST**	**SPI9 Controller Reset** 0 = SPI9 controller normal operation. 1 = SPI9 controller reset.
[11]	**SPI8RST**	**SPI8 Controller Reset** 0 = SPI8 controller normal operation. 1 = SPI8 controller reset.
[10]	**SPI7RST**	**SPI7 Controller Reset** 0 = SPI7 controller normal operation. 1 = SPI7 controller reset.
[9]	**SPI6RST**	**SPI6 Controller Reset** 0 = SPI6 controller normal operation. 1 = SPI6 controller reset.

[8]	**SPI5RST**	**SPI5 Controller Reset** 0 = SPI5 controller normal operation. 1 = SPI5 controller reset.
[7]	**ACMP23RST**	**Analog Comparator 2/3 Controller Reset** 0 = Analog Comparator 2/3 controller normal operation. 1 = Analog Comparator 2/3 controller reset.
[6]	**EADC2RST**	**EADC2 Controller Reset** 0 = EADC2 controller normal operation. 1 = EADC2 controller reset.
[5:1]	**Reserved**	Reserved.
[0]	**KPIRST**	**KPI Controller Reset** 0 = KPI controller normal operation. 1 = KPI controller reset.

Power-on Reset Controller Register (SYS_PORCTL)

Register	Offset	R/W	Description	Reset Value
SYS_PORCTL	SYS_BA+0x24	R/W	Power-On-reset Controller Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
POROFF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POROFF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**POROFF**	**Power-on Reset Enable Bit (Write Protect)** When powered on, the POR circuit generates a reset signal to reset the whole chip function, but noise on the power may cause the POR active again. User can disable internal POR circuit to avoid unpredictable noise to cause chip reset by writing 0x5AA5 to this field. The POR function will be active again when this field is set to another value or chip is reset by other reset source, including: nRESET, Watchdog, LVR reset, BOD reset, ICE reset command and the software-chip reset function. **Note 1:** Need to disable LVR by setting LVREN(SYS_BODCTL[7]) to 0 when setting POROFF(SYS_PORCTL[15:0]) to 0x5AA5. **Note 2:** These bits are write protected. Refer to the SYS_REGLCTL register.

V_{REF} Control Register (SYS_VREFCTL)

Register	Offset	R/W	Description	Reset Value
SYS_VREFCTL	SYS_BA+0x28	R/W	V_{REF} Control Register	0x0000_0200

31	30	29	28	27	26	25	24
Reserved					VBGISEL		VBFGEN
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PRELOADSEL		Reserved	VREFCTL				

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:25]	**VBGISEL**	**Chip Internal Voltage Band-gap Current Selection Bits (Write Protect)** 00 = Band-gap voltage buffer current is 4.2uA. 01 = Band-gap voltage buffer current is 7.3uA. 10 = Band-gap voltage buffer current is 10.4uA. 11 = Band-gap voltage buffer current is 13.5uA. **Note 1:** When ADC conversion source select band-gap voltage, suggest set VBGISEL as 10. **Note 2:** These bits are write protected. Refer to the SYS_REGLCTL register.
[24]	**VBGFEN**	**Chip Internal Voltage Band-gap Force Enable Bit (Write Protect)** 0 = Chip internal voltage band-gap controlled by ADC/ACMP if source selected. 1 = Chip internal voltage band-gap force enable. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[23:8]	**Reserved**	Reserved.
[7:6]	**PRELOADSEL**	**Pre-load Timing Selection (Write Protect)** 00 = Pre-load time is 60us for 0.1uF Capacitor. 01 = Pre-load time is 310us for 1uF Capacitor. 10 = Pre-load time is 1270us for 4.7uF Capacitor. 11 = Pre-load time is 2650us for 10uF Capacitor. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[5]	**Reserved**	Reserved.
[4:0]	**VREFCTL**	**V_{REF} Control Bits (Write Protect)** 00000 = V_{REF} is from external pin. 00011 = V_{REF} is internal 1.6V. 00111 = V_{REF} is internal 2.0V. 01011 = V_{REF} is internal 2.5V. 01111 = V_{REF} is internal 3.0V.

		Others = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

USB PHY Control Register (SYS_USBPHY)

Register	Offset	R/W	Description	Reset Value
SYS_USBPHY	SYS_BA+0x2C	R/W	USB PHY Control Register	0x0003_0007

31	30	29	28	27	26	25	24
Reserved						HSUSBACT	HSUSBEN
23	22	21	20	19	18	17	16
Reserved						HSUSBROLE	
15	14	13	12	11	10	9	8
Reserved							USBEN
7	6	5	4	3	2	1	0
Reserved					SBO	USBROLE	

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**HSUSBACT**	**HSUSB PHY Active Control** This bit is used to control HSUSB PHY at reset state or active state. 0 = HSUSB PHY at reset state. 1 = HSUSB PHY at active state. **Note:** After setting HSUSBEN (SYS_USBPHY[24]) to enable HSUSB PHY, user should keep HSUSB PHY at reset mode at lease 10us before changing to active mode.
[24]	**HSUSBEN**	**HSUSB PHY Enable** This bit is used to enable/disable HSUSB PHY. 0 = HSUSB PHY Disabled. 1 = HSUSB PHY Enabled.
[23:18]	**Reserved**	Reserved.
[17:16]	**HSUSBROLE**	**HSUSB Role Option (Write Protect)** These two bits are used to select the role of HSUSB. 00 = Standard HSUSB Device mode. 01 = Standard HSUSB Host mode. 10 = ID dependent mode. 11 = On-The-Go device mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15:9]	**Reserved**	Reserved.
[8]	**USBEN**	**USB PHY Enable** This bit is used to enable/disable USB PHY. 0 = USB PHY Disabled. 1 = USB PHY Enabled.
[7:3]	**Reserved**	Reserved.
[2]	**SBO**	**Note:** This bit must always be kept 1. If set to 0, the result is unpredictable.

[1:0]	**USBROLE**	**USB Role Option (Write Protect)** These two bits are used to select the role of USB. 00 = Standard USB Device mode. 01 = Standard USB Host mode. 10 = ID dependent mode. 11 = On-The-Go device mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

GPIO A-J Multiple Function Output Select Register (SYS_GPx_MFOS)

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFOS	SYS_BA+0x80	R/W	GPIOA Multiple Function Output Select Register	0x0000_0000
SYS_GPB_MFOS	SYS_BA+0x84	R/W	GPIOB Multiple Function Output Select Register	0x0000_0000
SYS_GPC_MFOS	SYS_BA+0x88	R/W	GPIOC Multiple Function Output Select Register	0x0000_0000
SYS_GPD_MFOS	SYS_BA+0x8C	R/W	GPIOD Multiple Function Output Select Register	0x0000_0000
SYS_GPE_MFOS	SYS_BA+0x90	R/W	GPIOE Multiple Function Output Select Register	0x0000_0000
SYS_GPF_MFOS	SYS_BA+0x94	R/W	GPIOF Multiple Function Output Select Register	0x0000_0000
SYS_GPG_MFOS	SYS_BA+0x98	R/W	GPIOG Multiple Function Output Select Register	0x0000_0000
SYS_GPH_MFOS	SYS_BA+0x9C	R/W	GPIOH Multiple Function Output Select Register	0x0000_0000
SYS_GPI_MFOS	SYS_BA+0xA0	R/W	GPIOI Multiple Function Output Select Register	0x0000_0000
SYS_GPJ_MFOS	SYS_BA+0xA4	R/W	GPIOJ Multiple Function Output Select Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MFOS							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MFOS							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**MFOS**	**GPIOA-J Pin[n] Multiple Function Pin Output Mode Select** This bit used to select multiple function pin output mode type for Px.n pin. 0 = Multiple function pin output mode type is Push-pull mode. 1 = Multiple function pin output mode type is Open-drain mode. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

System SRAM Parity Error Interrupt Enable Control Register (SYS_SRAM_INTCTL)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_INTCTL	SYS_BA+0xC0	R/W	System SRAM Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							PERRIEN

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**PERRIEN**	**SRAM Parity Check Error Interrupt Enable Bit** 0 = SRAM parity check error interrupt Disabled. 1 = SRAM parity check error interrupt Enabled.

System SRAM Parity Check Status Register (SYS_SRAM_STATUS)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_STATUS	SYS_BA+0xC4	R/W	System SRAM Parity Error Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							PERRIF

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**PERRIF**	**SRAM Parity Check Error Flag** This bit indicates the System SRAM parity error occurred. Write 1 to clear this to 0. 0 = No System SRAM parity error. 1 = System SRAM parity error occur.

System SRAM Parity Error Address Register (SYS_SRAM_ERRADDR)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_ERRADDR	SYS_BA+0xC8	R	System SRAM Parity Check Error Address Register	0x0000_0000

31	30	29	28	27	26	25	24
ERRADDR							
23	22	21	20	19	18	17	16
ERRADDR							
15	14	13	12	11	10	9	8
ERRADDR							
7	6	5	4	3	2	1	0
ERRADDR							

Bits	Description	
[31:0]	ERRADDR	**System SRAM Parity Error Address (Read Only)** This register shows system SRAM parity error byte address.

System SRAM BIST Test Control Register (SYS_SRAM_BISTCTL)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_BISTCTL	SYS_BA+0xD0	R/W	System SRAM BIST Test Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		RSABIST	CCAPBIST	KSBIST	SRBIST2	HSUSBHBIST	HSUSBDBIST
7	6	5	4	3	2	1	0
Reserved	EMAC0BIST	SPIMBIST	USBBIST	CANBIST	CRBIST	SRBIST1	SRBIST0

Bits	Description	
[31:11]	**Reserved**	Reserved.
[13]	**RSABIST**	**RSA SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for RSA SRAM. 0 = RSA SRAM BIST Disabled. 1 = RSA SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[12]	**CCAPBIST**	**CCAP SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for CCAP SRAM. 0 = CCAP SRAM BIST Disabled. 1 = CCAP SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11]	**KSBIST**	**Key Store SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for Key Store SRAM. 0 = Key Store SRAM BIST Disabled. 1 = Key Store SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[10]	**SRBIST2**	**System SRAM Bank2 BIST Enable Bit (Write Protect)** This bit enables BIST test for system SRAM bank2. 0 = System SRAM bank2 BIST Disabled. 1 = System SRAM bank2 BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[9]	**HSUSBHBIST**	**HSUSBH SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for HSUSBH SRAM. 0 = HSUSBH SRAM BIST Disabled. 1 = HSUSBH SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[8]	**HSUSBDBIST**	**HSUSBD SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for HSUSBD SRAM. 0 = HSUSBD SRAM BIST Disabled. 1 = HSUSBD SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	**Reserved**	Reserved.
[6]	**EMAC0BIST**	**EMAC0 SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for EMAC0 SRAM. 0 = EMAC0 SRAM BIST Disabled. 1 = EMAC0 SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**SPIMBIST**	**SPIM SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for SPIM SRAM. 0 = SPIM SRAM BIST Disabled. 1 = SPIM SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**USBBIST**	**USB SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for USB SRAM. 0 = USB SRAM BIST Disabled. 1 = USB SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**CANBIST**	**CAN SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for CAN SRAM. 0 = CAN SRAM BIST Disabled. 1 = CAN SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**CRBIST**	**CACHE SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for CACHE SRAM. 0 = CACHE SRAM BIST Disabled. 1 = CACHE SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[1]	**SRBIST1**	**System SRAM Bank1 BIST Enable Bit (Write Protect)** This bit enables BIST test for system SRAM bank1. 0 = System SRAM bank1 BIST Disabled. 1 = System SRAM bank1 BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**SRBIST0**	**System SRAM Bank0 BIST Enable Bit (Write Protect)** This bit enables BIST test for system SRAM bank0. 0 = System SRAM bank0 BIST Disabled. 1 = System SRAM bank0 BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

System SRAM BIST Test Status Register (SYS_SRAM_BISTSTS)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_BISTSTS	SYS_BA+0xD4	R	System SRAM BIST Test Status Register	0x00XX_00XX

31	30	29	28	27	26	25	24
Reserved		RSABEND	CCAPBEND	KSBEND	SRBEND2	HSUSBHBEND	HSUSBDBEND
23	22	21	20	19	18	17	16
Reserved	EMAC0BEND	SPIMBEND	USBBEND	CANBEND	CRBEND	SRBEND1	SRBEND0
15	14	13	12	11	10	9	8
Reserved		RSABISTEF	CCAPBISTEF	KSBISTEF	SRBISTEF2	HSUSBHBEF	HSUSBDBEF
7	6	5	4	3	2	1	0
Reserved	EAMC0BEF	SPIMBEF	USBBEF	CANBEF	CRBISTEF	SRBISTEF1	SRBISTEF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**RSABEND**	**RSA SRAM BIST Test Finish (Read Only)** 0 = RSA SRAM BIST is active. 1 = RSA SRAM BIST test finish.
[28]	**CCAPBEND**	**CCAP BIST Test Finish (Read Only)** 0 = CCAP SRAM BIST is active. 1 = CCAP SRAM BIST test finish.
[27]	**KSBEND**	**Key Store SRAM BIST Test Finish (Read Only)** 0 = Key Store SRAM BIST is active. 1 = Key Store SRAM BIST test finish.
[26]	**SRBEND2**	**System SRAM Bnak2 BIST Test Finish (Read Only)** 0 = System SRAM bank2 BIST is active. 1 = System SRAM bank2 BIST finish.
[25]	**HSUSBHBEND**	**HSUSBH SRAM BIST Test Finish (Read Only)** 0 = HSUSBH SRAM BIST is active. 1 = HSUSBH SRAM BIST test finish.
[24]	**HSUSBDBEND**	**HSUSBD SRAM BIST Test Finish (Read Only)** 0 = HSUSBD SRAM BIST is active. 1 = HSUSBD SRAM BIST test finish.
[23]	**Reserved**	Reserved.
[22]	**EMAC0BEND**	**EMAC0 SRAM BIST Test Finish (Read Only)** 0 = EMAC0 SRAM BIST is active. 1 = EMAC0 SRAM BIST test finish.
[21]	**SPIMBEND**	**SPIM SRAM BIST Test Finish (Read Only)** 0 = SPIM SRAM BIST is active.

		1 = SPIM SRAM BIST test finish.
[20]	**USBBEND**	**USB SRAM BIST Test Finish (Read Only)** 0 = USB SRAM BIST is active. 1 = USB SRAM BIST test finish.
[19]	**CANBEND**	**CAN SRAM BIST Test Finish (Read Only)** 0 = CAN SRAM BIST is active. 1 = CAN SRAM BIST test finish. **Note:** All of the CAN SRAM macros BIST finish, this flag is 1.
[18]	**CRBEND**	**CACHE SRAM BIST Test Finish (Read Only)** 0 = CACHE SRAM BIST is active. 1 = CACHE SRAM BIST test finish.
[17]	**SRBEND1**	**System SRAM Bnak1 BIST Test Finish (Read Only)** 0 = System SRAM bank1 BIST is active. 1 = System SRAM bank1 BIST finish.
[16]	**SRBEND0**	**System SRAM Bank0 BIST Test Finish (Read Only)** 0 = System SRAM bank0 BIST active. 1 = System SRAM bank0 BIST finish.
[15:14]	**Reserved**	Reserved.
[13]	**RSABISTE**	**RSA SRAM BIST Fail Flag (Read Only)** 0 = RSA SRAM BIST test pass. 1 = RSA SRAM BIST test fail.
[12]	**CCAPBISTEF**	**CCAP SRAM BIST Fail Flag (Read Only)** 0 = CCAP SRAM BIST test pass. 1 = CCAP SRAM BIST test fail.
[11]	**KSBISTEF**	**Key Store SRAM BIST Fail Flag (Read Only)** 0 = Key Store SRAM BIST test pass. 1 = Key Store SRAM BIST test fail.
[10]	**SRBISTEF2**	**System Bnak2 SRAM BIST Fail Flag (Read Only)** 0 = System SRAM bank2 BIST test pass. 1 = System SRAM bank2 BIST test fail.
[9]	**HSUSBHBEF**	**HSUSBH SRAM BIST Fail Flag (Read Only)** 0 = HSUSBH SRAM BIST test pass. 1 = HSUSBH SRAM BIST test fail.
[8]	**HSUSBDBEF**	**HSUSBD SRAM BIST Fail Flag (Read Only)** 0 = HSUSBD SRAM BIST test pass. 1 = HSUSBD SRAM BIST test fail.
[7]	**Reserved**	Reserved.
[6]	**EMAC0BEF**	**EMAC0 SRAM BIST Fail Flag (Read Only)** 0 = EMAC0 SRAM BIST test pass. 1 = EMAC0 SRAM BIST test fail.
[5]	**SPIMBEF**	**SPIM SRAM BIST Fail Flag (Read Only)** 0 = SPIM SRAM BIST test pass. 1 = SPIM SRAM BIST test fail.

[4]	**USBBEF**	**USB SRAM BIST Fail Flag (Read Only)** 0 = USB SRAM BIST test pass. 1 = USB SRAM BIST test fail.
[3]	**CANBEF**	**CAN SRAM BIST Fail Flag (Read Only)** 0 = CAN SRAM BIST test pass. 1 = CAN SRAM BIST test fail. **Note:** Any of the CAN SRAM macros BIST fail, this flag is 1.
[2]	**CRBISTEF**	**CACHE SRAM BIST Fail Flag (Read Only)** 0 = CACHE RAM BIST test pass. 1 = CACHE RAM BIST test fail.
[1]	**SRBISTEF1**	**System SRAM Bank1 BIST Fail Flag (Read Only)** 0 = System SRAM bank1 BIST test pass. 1 = System SRAM bank1 BIST test fail.
[0]	**SRBISTEF0**	**System SRAM Bank0 BIST Fail Flag (Read Only)** 0 = System SRAM bank0 BIST test pass. 1 = System SRAM bank0 BIST test fail.

HIRC48M Trim Control Register (SYS_HIRCTCTL)

Register	Offset	R/W	Description	Reset Value
SYS_HIRCTCTL	SYS_BA+0xE4	R/W	HIRC48M Trim Control Register	0x0008_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			BOUNDARY				
15	14	13	12	11	10	9	8
Reserved					REFCKSEL	BOUNDEN	CESTOPEN
7	6	5	4	3	2	1	0
RETRYCNT		LOOPSEL		Reserved		FREQSEL	

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:16]	**BOUNDARY**	**Boundary Selection** Fill the boundary range from 0x1 to 0x31, 0x0 is reserved. **Note:** This field is effective only when the BOUNDEN(SYS_HIRCTCTL[9]) is enabled.
[15:11]	**Reserved**	Reserved.
[10]	**REFCKSEL**	**Reference Clock Selection** 0 = HIRC trim reference clock is from LXT (32.768 kHz). 1 = HIRC trim reference clock is from internal USB synchronous mode. **Note:** HIRC trim reference clock is 20 kHz in test mode.
[9]	**BOUNDEN**	**Boundary Enable Bit** 0 = Boundary function Disabled. 1 = Boundary function Enabled.
[8]	**CESTOPEN**	**Clock Error Stop Enable Bit** 0 = The trim operation is keep going if clock is inaccuracy. 1 = The trim operation is stopped if clock is inaccuracy.
[7:6]	**RETRYCNT**	**Trim Value Update Limitation Count** This field defines that how many times the auto trim circuit will try to update the HIRC trim value before the frequency of HIRC locked. Once the HIRC locked, the internal trim value update counter will be reset. If the trim value update counter reached this limitation value and frequency of HIRC still doesn't lock, the auto trim operation will be disabled and FREQSEL will be cleared to 00. 00 = Trim retry count limitation is 64 loops. 01 = Trim retry count limitation is 128 loops. 10 = Trim retry count limitation is 256 loops. 11 = Trim retry count limitation is 512 loops.
[5:4]	**LOOPSEL**	**Trim Calculation Loop Selection** This field defines that trim value calculation is based on how many reference clocks.

		00 = Trim value calculation is based on average difference in 4 clocks of reference clock. 01 = Trim value calculation is based on average difference in 8 clocks of reference clock. 10 = Trim value calculation is based on average difference in 16 clocks of reference clock. 11 = Trim value calculation is based on average difference in 32 clocks of reference clock. **Note:** For example, if LOOPSEL is set as 00, auto trim circuit will calculate trim value based on the average frequency difference in 4 clocks of reference clock.
[3:2]	**Reserved**	Reserved.
[1:0]	**FREQSEL**	**Trim Frequency Selection** This field indicates the target frequency of 48 MHz internal high speed RC oscillator (HIRC48M) auto trim. During auto trim operation, if clock error detected with CESTOPEN is set to 1 or trim retry limitation count reached, this field will be cleared to 00 automatically. 00 = Disable HIRC auto trim function. 01 = Enable HIRC auto trim function and trim HIRC to 48 MHz. 10 = Reserved. 11 = Reserved.

HIRC48M Trim Interrupt Enable Register (SYS_HIRCTIEN)

Register	Offset	R/W	Description	Reset Value
SYS_HIRCTIEN	SYS_BA+0xE8	R/W	HIRC48M Trim Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CLKEIEN	TFAILIEN	Reserved

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**CLKEIEN**	**Clock Error Interrupt Enable Bit** This bit controls if CPU would get an interrupt while clock is inaccuracy during auto trim operation. If this bit is set to1, and CLKERRIF(SYS_HIRCTISTS[2]) is set during auto trim operation, an interrupt will be triggered to notify the clock frequency is inaccuracy. 0 = Disable CLKERRIF(SYS_HIRCTISTS[2]) status to trigger an interrupt to CPU. 1 = Enable CLKERRIF(SYS_HIRCTISTS[2]) status to trigger an interrupt to CPU.
[1]	**TFAILIEN**	**Trim Failure Interrupt Enable Bit** This bit controls if an interrupt will be triggered while HIRC trim value update limitation count reached and HIRC frequency still not locked on target frequency set by FREQSEL(SYS_HIRCTCTL[1:0]). If this bit is high and TFAILIF(SYS_HIRCTISTS[1]) is set during auto trim operation, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. 0 = Disable TFAILIF(SYS_HIRCTISTS[1]) status to trigger an interrupt to CPU. 1 = Enable TFAILIF(SYS_HIRCTISTS[1]) status to trigger an interrupt to CPU.
[0]	**Reserved**	Reserved.

HIRC48M Trim Interrupt Status Register (SYS_HIRCTISTS)

Register	Offset	R/W	Description	Reset Value
SYS_HIRCTISTS	SYS_BA+0xEC	R/W	HIRC48M Trim Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				OVBDIF	CLKERRIF	TFAILIF	FREQLOCK

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**OVBDIF**	**Over Boundary Status** When the over boundary function is set, if there occurs the over boundary condition, this flag will be set. 0 = Over boundary condition did not occur. 1 = Over boundary condition occurred. **Note:** Write 1 to clear this flag.
[2]	**CLKERRIF**	**Clock Error Interrupt Status** When the frequency of 32.768 kHz external low speed crystal oscillator (LXT) or 48 MHz internal high speed RC oscillator (HIRC48M) is shift larger to unreasonable value, this bit will be set and to be an indicate that clock frequency is inaccuracy. Once this bit is set to 1, the auto trim operation stopped and FREQSEL(SYS_HIRCTCTL[1:0]) will be cleared to 00 by hardware automatically if CESTOPEN(SYS_HIRCTCTL[8]) is set to 1. If this bit is set and CLKEIEN(SYS_HIRCTIEN[2]) is high, an interrupt will be triggered to notify the clock frequency is inaccuracy. Write 1 to clear this to 0. 0 = Clock frequency is accuracy. 1 = Clock frequency is inaccuracy.
[1]	**TFAILIF**	**Trim Failure Interrupt Status** This bit indicates that HIRC trim value update limitation count reached and the HIRC clock frequency still doesn't be locked. Once this bit is set, the auto trim operation stopped and FREQSEL(SYS_HIRCTCTL[1:0]) will be cleared to 00 by hardware automatically. If this bit is set and TFAILIEN(SYS_HIRCTIEN[1]) is high, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. Write 1 to clear this to 0. 0 = Trim value update limitation count does not reach. 1 = Trim value update limitation count reached and HIRC frequency still not locked.
[0]	**FREQLOCK**	**HIRC Frequency Lock Status** This bit indicates the HIRC frequency is locked. This is a status bit and doesn't trigger any interrupt. Write 1 to clear this to 0. This bit will be set automatically, if the frequency is lock and the

		RC_TRIM is enabled. 0 = The internal high-speed oscillator frequency doesn't lock at 48 MHz yet. 1 = The internal high-speed oscillator frequency locked at 48 MHz.

HIRC Trim Control Register (SYS_IRCTCTL)

Register	Offset	R/W	Description	Reset Value
SYS_IRCTCTL	SYS_BA+0xF0	R/W	HIRC Trim Control Register	0x0008_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			BOUNDARY				
15	14	13	12	11	10	9	8
Reserved					REFCKSEL	BOUNDEN	CESTOPEN
7	6	5	4	3	2	1	0
RETRYCNT		LOOPSEL		Reserved		FREQSEL	

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:16]	**BOUNDARY**	**Boundary Selection** Fill the boundary range from 0x1 to 0x31, 0x0 is reserved. **Note:** This field is effective only when the BOUNDEN(SYS_IRCTCTL[9]) is enabled.
[15:11]	**Reserved**	Reserved.
[10]	**REFCKSEL**	**Reference Clock Selection** 0 = HIRC trim reference clock is from LXT (32.768 kHz). 1 = HIRC trim reference clock is from internal USB synchronous mode. **Note:** HIRC trim reference clock is 20 kHz in test mode.
[9]	**BOUNDEN**	**Boundary Enable Bit** 0 = Boundary function Disabled. 1 = Boundary function Enabled.
[8]	**CESTOPEN**	**Clock Error Stop Enable Bit** 0 = The trim operation is keep going if clock is inaccuracy. 1 = The trim operation is stopped if clock is inaccuracy.
[7:6]	**RETRYCNT**	**Trim Value Update Limitation Count** This field defines that how many times the auto trim circuit will try to update the HIRC trim value before the frequency of HIRC locked. Once the HIRC locked, the internal trim value update counter will be reset. If the trim value update counter reached this limitation value and frequency of HIRC still doesn't lock, the auto trim operation will be disabled and FREQSEL will be cleared to 00. 00 = Trim retry count limitation is 64 loops. 01 = Trim retry count limitation is 128 loops. 10 = Trim retry count limitation is 256 loops. 11 = Trim retry count limitation is 512 loops.
[5:4]	**LOOPSEL**	**Trim Calculation Loop Selection** This field defines that trim value calculation is based on how many reference clocks.

		00 = Trim value calculation is based on average difference in 4 clocks of reference clock. 01 = Trim value calculation is based on average difference in 8 clocks of reference clock. 10 = Trim value calculation is based on average difference in 16 clocks of reference clock. 11 = Trim value calculation is based on average difference in 32 clocks of reference clock. **Note:** For example, if LOOPSEL is set as 00, auto trim circuit will calculate trim value based on the average frequency difference in 4 clocks of reference clock.
[3:2]	**Reserved**	Reserved.
[1:0]	**FREQSEL**	**Trim Frequency Selection** This field indicates the target frequency of 12 MHz internal high speed RC oscillator (HIRC) auto trim. During auto trim operation, if clock error detected with CESTOPEN is set to 1 or trim retry limitation count reached, this field will be cleared to 00 automatically. 00 = Disable HIRC auto trim function. 01 = Enable HIRC auto trim function and trim HIRC to 12 MHz. 10 = Reserved. 11 = Reserved.

HIRC Trim Interrupt Enable Register (SYS_IRCTIEN)

Register	Offset	R/W	Description	Reset Value
SYS_IRCTIEN	SYS_BA+0xF4	R/W	HIRC Trim Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CLKEIEN	TFAILIEN	Reserved

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**CLKEIEN**	**Clock Error Interrupt Enable Bit** This bit controls if CPU would get an interrupt while clock is inaccuracy during auto trim operation. If this bit is set to1, and CLKERRIF(SYS_IRCTISTS[2]) is set during auto trim operation, an interrupt will be triggered to notify the clock frequency is inaccuracy. 0 = Disable CLKERRIF(SYS_IRCTISTS[2]) status to trigger an interrupt to CPU. 1 = Enable CLKERRIF(SYS_IRCTISTS[2]) status to trigger an interrupt to CPU.
[1]	**TFAILIEN**	**Trim Failure Interrupt Enable Bit** This bit controls if an interrupt will be triggered while HIRC trim value update limitation count reached and HIRC frequency still not locked on target frequency set by FREQSEL(SYS_IRCTCTL[1:0]). If this bit is high and TFAILIF(SYS_IRCTISTS[1]) is set during auto trim operation, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. 0 = Disable TFAILIF(SYS_IRCTISTS[1]) status to trigger an interrupt to CPU. 1 = Enable TFAILIF(SYS_IRCTISTS[1]) status to trigger an interrupt to CPU.
[0]	**Reserved**	Reserved.

HIRC Trim Interrupt Status Register (SYS_IRCTISTS)

Register	Offset	R/W	Description	Reset Value
SYS_IRCTISTS	SYS_BA+0xF8	R/W	HIRC Trim Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				OVBDIF	CLKERRIF	TFAILIF	FREQLOCK

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**OVBDIF**	**Over Boundary Status** When the over boundary function is set, if there occurs the over boundary condition, this flag will be set. 0 = Over boundary condition did not occur. 1 = Over boundary condition occurred. **Note:** Write 1 to clear this flag.
[2]	**CLKERRIF**	**Clock Error Interrupt Status** When the frequency of 32.768 kHz external low speed crystal oscillator (LXT) or 12 MHz internal high speed RC oscillator (HIRC) is shift larger to unreasonable value, this bit will be set and to be an indicate that clock frequency is inaccuracy. Once this bit is set to 1, the auto trim operation stopped and FREQSEL(SYS_IRCTCTL[1:0]) will be cleared to 00 by hardware automatically if CESTOPEN(SYS_IRCTCTL[8]) is set to 1. If this bit is set and CLKEIEN(SYS_IRCTIEN[2]) is high, an interrupt will be triggered to notify the clock frequency is inaccuracy. Write 1 to clear this to 0. 0 = Clock frequency is accuracy. 1 = Clock frequency is inaccuracy.
[1]	**TFAILIF**	**Trim Failure Interrupt Status** This bit indicates that HIRC trim value update limitation count reached and the HIRC clock frequency still doesn't be locked. Once this bit is set, the auto trim operation stopped and FREQSEL(SYS_IRCTCTL[1:0]) will be cleared to 00 by hardware automatically. If this bit is set and TFAILIEN(SYS_IRCTIEN[1]) is high, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. Write 1 to clear this to 0. 0 = Trim value update limitation count does not reach. 1 = Trim value update limitation count reached and HIRC frequency still not locked.
[0]	**FREQLOCK**	**HIRC Frequency Lock Status** This bit indicates the HIRC frequency is locked. This is a status bit and doesn't trigger any interrupt. Write 1 to clear this to 0. This bit will be set automatically, if the frequency is lock and the

		RC_TRIM is enabled. 0 = The internal high-speed oscillator frequency doesn't lock at 12 MHz yet. 1 = The internal high-speed oscillator frequency locked at 12 MHz.

Register Lock Control Register (SYS_REGLCTL)

Some of the system control registers need to be protected to avoid inadvertent write and disturb the chip operation. These system control registers are protected after the power-on reset till user to disable register protection. For user to program these protected registers, a register protection disable sequence needs to be followed by a special programming. The register protection disable sequence is writing the data “59h”, “16h” “88h” to the register SYS_REGLCTL address at 0x4000_0100 continuously. Any different data value, different sequence or any other write to other address during these three data writing will abort the whole sequence.

After the protection is disabled, user can check the protection disable bit at address 0x4000_0100 bit0, 1 is protection disable, and 0 is protection enable. Then user can update the target protected register value and then write any data to the address “0x4000_0100” to enable register protection.

This register is written to disable/enable register protection and read for the REGLCTL status.

Register	Offset	R/W	Description	Reset Value
SYS_REGLCTL	SYS_BA+0x100	R/W	Register Lock Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REGLCTL							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**REGLCTL**	**Register Lock Control Code (Write Only)** Some registers have write-protection function. Writing these registers have to disable the protected function by writing the sequence value “59h”, “16h”, “88h” to this field. After this sequence is completed, the REGLCTL bit will be set to 1 and write-protection registers can be normal write. **REGLCTL[0]** **Register Lock Control Disable Index (Read Only)** 0 = Write-protection Enabled for writing protected registers. Any write to the protected register is ignored. 1 = Write-protection Disabled for writing protected registers.

Analog POR Disable Control Register (SYS_PORDISAN)

Register	Offset	R/W	Description	Reset Value
SYS_PORDISAN	SYS_BA+0x1EC	R/W	Analog POR Disable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
POROFFAN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POROFFAN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**POROFFAN**	**Power-on Reset Enable Bit (Write Protect)** After powered on, User can turn off internal analog POR circuit to save power by writing 0x5AA5 to this field. The analog POR circuit will be active again when this field is set to another value or chip is reset by other reset source, including: nRESET, Watchdog, LVR reset, BOD reset, ICE reset command and the software-chip reset function. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Chip Series Version Register (SYS_CSERVER)

Register	Offset	R/W	Description	Reset Value
SYS_CSERVER	SYS_BA+0x1F4	R	Chip Series Version Register	0x0000_00XX

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
VERSION							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**VERSION**	**Chip Series Version (Read Only)** These bits indicate the series version of chip. 0x02 = M467. 0x03 = M463. Others = Reserved.

Power Level Control Register (SYS_PLCTL)

Register	Offset	R/W	Description	Reset Value
SYS_PLCTL	SYS_BA+0x1F8	R/W	Power Level Control Register	0x0000_0000

31	30	29	28	27	26	25	24
LVSPRD							
23	22	21	20	19	18	17	16
Reserved		LVSSTEP					
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						PLSEL	

Bits	Description	
[31:24]	**LVSPRD**	**LDO Voltage Scaling Period (Write Protect)** The LVSPRD value is the period of each LDO voltage rising step. LDO voltage scaling period = (LVSPRD + 1) * 1us. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[23:22]	**Reserved**	Reserved.
[21:16]	**LVSSTEP**	**LDO Voltage Scaling Step (Write Protect)** The LVSSTEP value is LDO voltage rising step. LDO voltage scaling step = (LVSSTEP + 1) * 10mV. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15:2]	**Reserved**	Reserved.
[1:0]	**PLSEL**	**Power Level Select (Write Protect)** These bits indicate the status of power level. 00 = Power level is PL0. 01 = Power level is PL1. Others = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Power Level Status Register (SYS_PLSTS)

Register	Offset	R/W	Description	Reset Value
SYS_PLSTS	SYS_BA+0x1FC	R	Power Level Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						PLSTATUS	
7	6	5	4	3	2	1	0
Reserved							PLCBUSY

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:8]	**PLSTATUS**	**Power Level Status (Read Only)** This bit indicates the status of power level. 00 = Power level is PL0. 01 = Power level is PL1. Others = Reserved.
[7:1]	**Reserved**	Reserved.
[0]	**PLCBUSY**	**Power Level Change Busy Bit (Read Only)** This bit is set by hardware when power level is changing. After power level change is completed, this bit will be cleared automatically by hardware. 0 = Core voltage change is completed. 1 = Core voltage change is ongoing.

AHB Bus Matrix Priority Control Register (SYS_AHBMCTL)

Register	Offset	R/W	Description	Reset Value
SYS_AHBMCTL	SYS_BA+0x400	R/W	AHB Bus Matrix Priority Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**INTACTEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**INTACTEN**	**Highest AHB Bus Priority of Cortex-M4 Core Enable Bit (Write Protect)** Enable Cortex-M4 Core with Highest AHB Bus Priority in AHB Bus Matrix. 0 = Round-robin mode. 1 = Cortex-M4 CPU with highest bus priority when interrupt occurred. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

GPIOA Multiple Function Control Register 0 (SYS_GPA_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFP0	SYS_BA+0x500	R/W	GPIOA Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PA3MFP				
23	22	21	20	19	18	17	16
Reserved			PA2MFP				
15	14	13	12	11	10	9	8
Reserved			PA1MFP				
7	6	5	4	3	2	1	0
Reserved			PA0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PA3MFP**	**PA.3 Multi-function Pin Selection** 02 = SPIM_SS. 03 = QSPI0_SS. 04 = SPI0_SS. 05 = SD1_DAT3. 06 = SC0_PWR. 07 = UART4_TXD. 08 = UART1_TXD. 09 = I2C1_SCL. 10 = I2C0_SMBAL. 12 = BPWM0_CH3. 13 = EPWM0_CH2. 14 = QEI0_B. 15 = EPWM1_BRAKE1. 17 = PSIO0_CH4.
[23:21]	**Reserved**	Reserved.
[20:16]	**PA2MFP**	**PA.2 Multi-function Pin Selection** 02 = SPIM_CLK. 03 = QSPI0_CLK. 04 = SPI0_CLK. 05 = SD1_DAT2. 06 = SC0_RST. 07 = UART4_RXD. 08 = UART1_RXD. 09 = I2C1_SDA. 10 = I2C0_SMBSUS. 12 = BPWM0_CH2.

Bits	Description	
		13 = EPWM0_CH3. 14 = QEI3_INDEX. 17 = PSIO0_CH5.
[15:13]	**Reserved**	Reserved.
[12:8]	**PA1MFP**	**PA.1 Multi-function Pin Selection** 02 = SPIM_MISO. 03 = QSPI0_MISO0. 04 = SPI0_MISO. 05 = SD1_DAT1. 06 = SC0_DAT. 07 = UART0_TXD. 08 = UART1_nCTS. 09 = I2C2_SCL. 10 = CCAP_DATA7. 12 = BPWM0_CH1. 13 = EPWM0_CH4. 14 = QEI3_A. 15 = DAC1_ST. 17 = PSIO0_CH6.
[7:5]	**Reserved**	Reserved.
[4:0]	**PA0MFP**	**PA.0 Multi-function Pin Selection** 02 = SPIM_MOSI. 03 = QSPI0_MOSI0. 04 = SPI0_MOSI. 05 = SD1_DAT0. 06 = SC0_CLK. 07 = UART0_RXD. 08 = UART1_nRTS. 09 = I2C2_SDA. 10 = CCAP_DATA6. 12 = BPWM0_CH0. 13 = EPWM0_CH5. 14 = QEI3_B. 15 = DAC0_ST. 17 = PSIO0_CH7.

GPIOA Multiple Function Control Register 1 (SYS_GPA_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFP1	SYS_BA+0x504	R/W	GPIOA Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PA7MFP				
23	22	21	20	19	18	17	16
Reserved			PA6MFP				
15	14	13	12	11	10	9	8
Reserved			PA5MFP				
7	6	5	4	3	2	1	0
Reserved			PA4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PA7MFP**	**PA.7 Multi-function Pin Selection** 02 = EBI_AD7. 03 = EMAC_RMII_CRSDV. 04 = SPI1_CLK. 06 = SC2_DAT. 07 = UART0_TXD. 08 = I2C1_SCL. 09 = QSPI1_MISO1. 11 = EPWM1_CH4. 12 = BPWM1_CH2. 13 = ACMP0_WLAT. 14 = TM2. 15 = INT1. 17 = SPI5_SS. 18 = KPI_COL1. 19 = SPI6_SS.
[23:21]	**Reserved**	Reserved.
[20:16]	**PA6MFP**	**PA.6 Multi-function Pin Selection** 02 = EBI_AD6. 03 = EMAC_RMII_RXERR. 04 = SPI1_SS. 05 = SD1_nCD. 06 = SC2_CLK. 07 = UART0_RXD. 08 = I2C1_SDA. 09 = QSPI1_MOSI1. 11 = EPWM1_CH5.

Bits	Description	
		12 = BPWM1_CH3. 13 = ACMP1_WLAT. 14 = TM3. 15 = INT0. 17 = SPI5_CLK. 18 = KPI_COL0. 19 = SPI6_CLK.
[15:13]	**Reserved**	Reserved.
[12:8]	**PA5MFP**	**PA.5 Multi-function Pin Selection** 02 = SPIM_D2. 03 = QSPI0_MISO1. 04 = SPI1_I2SMCLK. 05 = SD1_CMD. 06 = SC2_nCD. 07 = UART0_nCTS. 08 = UART5_TXD. 09 = I2C0_SCL. 10 = CAN0_TXD. 11 = UART0_TXD. 12 = BPWM0_CH5. 13 = EPWM0_CH0. 14 = QEI0_INDEX.
[7:5]	**Reserved**	Reserved.
[4:0]	**PA4MFP**	**PA.4 Multi-function Pin Selection** 02 = SPIM_D3. 03 = QSPI0_MOSI1. 04 = SPI0_I2SMCLK. 05 = SD1_CLK. 06 = SC0_nCD. 07 = UART0_nRTS. 08 = UART5_RXD. 09 = I2C0_SDA. 10 = CAN0_RXD. 11 = UART0_RXD. 12 = BPWM0_CH4. 13 = EPWM0_CH1. 14 = QEI0_A.

GPIOA Multiple Function Control Register 2 (SYS_GPA_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFP2	SYS_BA+0x508	R/W	GPIOA Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PA11MFP				
23	22	21	20	19	18	17	16
Reserved			PA10MFP				
15	14	13	12	11	10	9	8
Reserved			PA9MFP				
7	6	5	4	3	2	1	0
Reserved			PA8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PA11MFP**	**PA.11 Multi-function Pin Selection** 01 = EADC1_CH7, EADC2_CH7, ACMP0_P0. 02 = EBI_nRD. 03 = SC2_PWR. 04 = SPI2_SS. 05 = SD1_DAT3. 06 = USCI0_CLK. 07 = I2C2_SCL. 08 = UART6_TXD. 09 = BPWM0_CH0. 10 = EPWM0_SYNC_OUT. 12 = I2S1_BCLK. 13 = TM0_EXT. 14 = DAC1_ST. 18 = KPI_ROW4.
[23:21]	**Reserved**	Reserved.
[20:16]	**PA10MFP**	**PA.10 Multi-function Pin Selection** 01 = EADC1_CH6, EADC2_CH6, ACMP1_P0. 02 = EBI_nWR. 03 = SC2_RST. 04 = SPI2_CLK. 05 = SD1_DAT2. 06 = USCI0_DAT0. 07 = I2C2_SDA. 08 = UART6_RXD. 09 = BPWM0_CH1. 10 = QEI1_INDEX.

Bits	Description	
		11 = ECAP0_IC0. 12 = I2S1_MCLK. 13 = TM1_EXT. 14 = DAC0_ST. 15 = SWDH_CLK. 18 = KPI_ROW5.
[15:13]	**Reserved**	Reserved.
[12:8]	**PA9MFP**	**PA.9 Multi-function Pin Selection** 01 = EADC1_CH5, EADC2_CH5. 02 = EBI_MCLK. 03 = SC2_DAT. 04 = SPI2_MISO. 05 = SD1_DAT1. 06 = USCI0_DAT1. 07 = UART1_TXD. 08 = UART7_TXD. 09 = BPWM0_CH2. 10 = QEI1_A. 11 = ECAP0_IC1. 12 = I2S1_DI. 13 = TM2_EXT. 15 = SWDH_DAT.
[7:5]	**Reserved**	Reserved.
[4:0]	**PA8MFP**	**PA.8 Multi-function Pin Selection** 01 = EADC1_CH4, EADC2_CH4. 02 = EBI_ALE. 03 = SC2_CLK. 04 = SPI2_MOSI. 05 = SD1_DAT0. 06 = USCI0_CTL1. 07 = UART1_RXD. 08 = UART7_RXD. 09 = BPWM0_CH3. 10 = QEI1_B. 11 = ECAP0_IC2. 12 = I2S1_DO. 13 = TM3_EXT. 15 = INT4.

GPIOA Multiple Function Control Register 3 (SYS_GPA_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFP3	SYS_BA+0x50C	R/W	GPIOA Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PA15MFP				
23	22	21	20	19	18	17	16
Reserved			PA14MFP				
15	14	13	12	11	10	9	8
Reserved			PA13MFP				
7	6	5	4	3	2	1	0
Reserved			PA12MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PA15MFP**	**PA.15 Multi-function Pin Selection** 02 = I2S0_DO. 03 = UART0_RXD. 04 = SPIM_MOSI. 05 = SPI2_MOSI. 06 = I2C2_SDA. 07 = SC2_CLK. 09 = SPI0_MOSI. 11 = BPWM1_CH5. 12 = EPWM0_SYNC_IN. 13 = QEI3_INDEX. 14 = USB_OTG_ID. 16 = I2C0_SDA. 17 = PSIO0_CH7. 19 = SPI10_MOSI.
[23:21]	**Reserved**	Reserved.
[20:16]	**PA14MFP**	**PA.14 Multi-function Pin Selection** 02 = I2S0_DI. 03 = UART0_TXD. 04 = EBI_AD5. 05 = SPI2_MISO. 06 = I2C2_SCL. 07 = SC2_DAT. 09 = SPI0_MISO. 11 = BPWM1_CH4. 12 = QEI1_B. 13 = ECAP3_IC2.

Bits	Description	
		14 = USB_D+. 16 = I2C0_SCL. 17 = PSIO0_CH6. 19 = SPI10_MISO.
[15:13]	**Reserved**	Reserved.
[12:8]	**PA13MFP**	**PA.13 Multi-function Pin Selection** 02 = I2S0_MCLK. 03 = UART4_RXD. 04 = I2C1_SDA. 05 = SPI2_CLK. 06 = CAN0_RXD. 07 = SC2_RST. 09 = SPI0_CLK. 10 = QSPI1_MOSI0. 11 = BPWM1_CH3. 12 = QEI1_A. 13 = ECAP3_IC1. 14 = USB_D-. 17 = PSIO0_CH5. 19 = SPI10_CLK.
[7:5]	**Reserved**	Reserved.
[4:0]	**PA12MFP**	**PA.12 Multi-function Pin Selection** 02 = I2S0_BCLK. 03 = UART4_TXD. 04 = I2C1_SCL. 05 = SPI2_SS. 06 = CAN0_TXD. 07 = SC2_PWR. 08 = SD1_nCD. 09 = SPI0_SS. 10 = QSPI1_MISO0. 11 = BPWM1_CH2. 12 = QEI1_INDEX. 13 = ECAP3_IC0. 14 = USB_VBUS. 17 = PSIO0_CH4. 19 = SPI10_SS.

GPIOB Multiple Function Control Register 0 (SYS_GPB_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPB_MFP0	SYS_BA+0x510	R/W	GPIOB Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PB3MFP				
23	22	21	20	19	18	17	16
Reserved			PB2MFP				
15	14	13	12	11	10	9	8
Reserved			PB1MFP				
7	6	5	4	3	2	1	0
Reserved			PB0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PB3MFP**	**PB.3 Multi-function Pin Selection** 01 = EADC0_CH3, EADC1_CH11, ACMP0_N. 02 = EBI_ADR2. 03 = SD0_DAT1. 04 = EMAC_RMII_RXD1. 05 = SPI1_CLK. 06 = UART1_TXD. 07 = UART5_nRTS. 09 = SC0_RST. 10 = I2S0_DI. 11 = EPWM0_CH2. 12 = I2C1_SCL. 14 = TM2. 15 = INT2. 17 = PSIO0_CH6. 18 = KPI_ROW0. 19 = SPI4_SS.
[23:21]	**Reserved**	Reserved.
[20:16]	**PB2MFP**	**PB.2 Multi-function Pin Selection** 01 = EADC0_CH2, EADC1_CH10, ACMP0_P1. 02 = EBI_ADR3. 03 = SD0_DAT0. 04 = EMAC_RMII_CRSDV. 05 = SPI1_SS. 06 = UART1_RXD. 07 = UART5_nCTS. 09 = SC0_PWR.

Bits	Description	
		10 = I2S0_DO. 11 = EPWM0_CH3. 12 = I2C1_SDA. 14 = TM3. 15 = INT3. 17 = PSIO0_CH7. 18 = KPI_ROW1. 19 = SPI4_CLK.
[15:13]	**Reserved**	Reserved.
[12:8]	**PB1MFP**	**PB.1 Multi-function Pin Selection** 01 = EADC0_CH1, EADC1_CH9, EADC2_CH9, ACMP3_P0. 02 = EBI_ADR8. 03 = SD0_CLK. 04 = EMAC_RMII_RXERR. 05 = SPI1_I2SMCLK. 06 = SPI3_I2SMCLK. 07 = UART2_TXD. 09 = I2C1_SCL. 10 = I2S0_LRCK. 11 = EPWM0_CH4. 12 = EPWM1_CH4. 13 = EPWM0_BRAKE0. 14 = ACMP2_O. 15 = QSPI0_MISO1. 18 = KPI_ROW2. 19 = SPI4_MISO.
[7:5]	**Reserved**	Reserved.
[4:0]	**PB0MFP**	**PB.0 Multi-function Pin Selection** 01 = EADC0_CH0, EADC1_CH8, EADC2_CH8, ACMP3_N. 02 = EBI_ADR9. 03 = SD0_CMD. 04 = SPI2_I2SMCLK. 06 = USCI0_CTL0. 07 = UART2_RXD. 08 = SPI0_I2SMCLK. 09 = I2C1_SDA. 10 = I2S1_LRCK. 11 = EPWM0_CH5. 12 = EPWM1_CH5. 13 = EPWM0_BRAKE1. 14 = ACMP3_O. 15 = QSPI0_MOSI1. 18 = KPI_ROW3. 19 = SPI4_MOSI.

GPIOB Multiple Function Control Register 1 (SYS_GPB_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPB_MFP1	SYS_BA+0x514	R/W	GPIOB Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PB7MFP				
23	22	21	20	19	18	17	16
Reserved			PB6MFP				
15	14	13	12	11	10	9	8
Reserved			PB5MFP				
7	6	5	4	3	2	1	0
Reserved			PB4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PB7MFP**	**PB.7 Multi-function Pin Selection** 01 = EADC0_CH7, EADC2_CH15, ACMP2_P0. 02 = EBI_nWRL. 03 = EMAC_RMII_TXEN. 05 = CAN1_TXD. 06 = UART1_TXD. 07 = SD1_CMD. 08 = EBI_nCS0. 10 = BPWM1_CH4. 11 = EPWM1_BRAKE0. 12 = EPWM1_CH4. 13 = INT5. 14 = USB_VBUS_ST. 15 = ACMP0_O. 16 = SPI3_MISO. 18 = KPI_COL4. 19 = SPI1_CLK.
[23:21]	**Reserved**	Reserved.
[20:16]	**PB6MFP**	**PB.6 Multi-function Pin Selection** 01 = EADC0_CH6, EADC2_CH14, ACMP2_N. 02 = EBI_nWRH. 03 = EMAC_PPS. 05 = CAN1_RXD. 06 = UART1_RXD. 07 = SD1_CLK. 08 = EBI_nCS1. 10 = BPWM1_CH5.

Bits	Description	
		11 = EPWM1_BRAKE1. 12 = EPWM1_CH5. 13 = INT4. 14 = USB_VBUS_EN. 15 = ACMP1_O. 16 = SPI3_MOSI 18 = KPI_COL5. 19 = SPI1_SS.
[15:13]	**Reserved**	Reserved.
[12:8]	**PB5MFP**	**PB.5 Multi-function Pin Selection** 01 = EADC0_CH5, ACMP1_N. 02 = EBI_ADR0. 03 = SD0_DAT3. 04 = EMAC_RMII_REFCLK. 05 = SPI1_MISO. 06 = I2C0_SCL. 07 = UART5_TXD. 09 = SC0_CLK. 10 = I2S0_BCLK. 11 = EPWM0_CH0. 12 = UART2_TXD. 14 = TM0. 15 = INT0. 17 = PSIO0_CH4. 18 = KPI_COL6.
[7:5]	**Reserved**	Reserved.
[4:0]	**PB4MFP**	**PB.4 Multi-function Pin Selection** 01 = EADC0_CH4, ACMP1_P1. 02 = EBI_ADR1. 03 = SD0_DAT2. 04 = EMAC_RMII_RXD0. 05 = SPI1_MOSI. 06 = I2C0_SDA. 07 = UART5_RXD. 09 = SC0_DAT. 10 = I2S0_MCLK. 11 = EPWM0_CH1. 12 = UART2_RXD. 14 = TM1. 15 = INT1. 17 = PSIO0_CH5. 18 = KPI_COL7.

GPIOB Multiple Function Control Register 2 (SYS_GPB_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPB_MFP2	SYS_BA+0x518	R/W	GPIOB Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PB11MFP				
23	22	21	20	19	18	17	16
Reserved			PB10MFP				
15	14	13	12	11	10	9	8
Reserved			PB9MFP				
7	6	5	4	3	2	1	0
Reserved			PB8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PB11MFP**	**PB.11 Multi-function Pin Selection** 01 = EADC0_CH11. 02 = EBI_ADR16. 03 = EMAC_RMII_MDC. 05 = UART0_nCTS. 06 = UART4_TXD. 07 = I2C1_SCL. 08 = CAN0_TXD. 09 = SPI0_I2SMCLK. 10 = BPWM1_CH0. 11 = SPI3_CLK. 12 = CCAP_SFIELD. 14 = HSUSB_VBUS_ST.
[23:21]	**Reserved**	Reserved.
[20:16]	**PB10MFP**	**PB.10 Multi-function Pin Selection** 01 = EADC0_CH10, ACMP2_P3. 02 = EBI_ADR17. 03 = EMAC_RMII_MDIO. 05 = UART0_nRTS. 06 = UART4_RXD. 07 = I2C1_SDA. 08 = CAN0_RXD. 10 = BPWM1_CH1. 11 = SPI3_SS. 12 = CCAP_VSYNC. 14 = HSUSB_VBUS_EN.

Bits	Description	
[15:13]	**Reserved**	Reserved.
[12:8]	**PB9MFP**	**PB.9 Multi-function Pin Selection** 01 = EADC0_CH9, ACMP2_P2. 02 = EBI_ADR18. 03 = EMAC_RMII_TXD0. 05 = UART0_TXD. 06 = UART1_nCTS. 07 = I2C1_SMBAL. 08 = UART7_TXD. 09 = I2C0_SCL. 10 = BPWM1_CH2. 11 = SPI3_MISO. 12 = CAN2_TXD. 13 = INT7. 14 = CCAP_HSYNC.
[7:5]	**Reserved**	Reserved.
[4:0]	**PB8MFP**	**PB.8 Multi-function Pin Selection** 01 = EADC0_CH8, ACMP2_P1. 02 = EBI_ADR19. 03 = EMAC_RMII_TXD1. 05 = UART0_RXD. 06 = UART1_nRTS. 07 = I2C1_SMBSUS. 08 = UART7_RXD. 09 = I2C0_SDA. 10 = BPWM1_CH3. 11 = SPI3_MOSI. 12 = CAN2_RXD. 13 = INT6. 14 = EADC2_ST.

GPIOB Multiple Function Control Register 3 (SYS_GPB_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPB_MFP3	SYS_BA+0x51C	R/W	GPIOB Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PB15MFP				
23	22	21	20	19	18	17	16
Reserved			PB14MFP				
15	14	13	12	11	10	9	8
Reserved			PB13MFP				
7	6	5	4	3	2	1	0
Reserved			PB12MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PB15MFP**	**PB.15 Multi-function Pin Selection** 01 = EADC0_CH15, EADC1_CH15. 02 = EBI_AD12. 03 = SC1_PWR. 04 = SPI0_SS. 05 = USCI0_CTL1. 06 = UART0_nCTS. 07 = UART3_TXD. 08 = I2C2_SMBAL. 09 = CCAP_DATA1. 10 = EPWM0_BRAKE1. 11 = EPWM1_CH0. 12 = ETMC_TRACE_DATA0. 13 = TM0_EXT. 14 = USB_VBUS_EN. 15 = HSUSB_VBUS_EN. 17 = PSIO0_CH0. 18 = KPI_COL0. 19 = SPI9_CLK.
[23:21]	**Reserved**	Reserved.
[20:16]	**PB14MFP**	**PB.14 Multi-function Pin Selection** 01 = EADC0_CH14, EADC1_CH14. 02 = EBI_AD13. 03 = SC1_RST. 04 = SPI0_CLK. 05 = USCI0_DAT1. 06 = UART0_nRTS.

Bits	Description	
		07 = UART3_RXD. 08 = I2C2_SMBSUS. 09 = CCAP_DATA0. 11 = EPWM1_CH1. 12 = ETMC_TRACE_DATA1. 13 = TM1_EXT. 14 = CLKO. 15 = USB_VBUS_ST. 17 = PSIO0_CH1. 18 = KPI_COL1. 19 = SPI9_SS.
[15:13]	**Reserved**	Reserved.
[12:8]	**PB13MFP**	**PB.13 Multi-function Pin Selection** 01 = EADC0_CH13, EADC1_CH13, DAC1_OUT, ACMP0_P3, ACMP1_P3. 02 = EBI_AD14. 03 = SC1_DAT. 04 = SPI0_MISO. 05 = USCI0_DAT0. 06 = UART0_TXD. 07 = UART3_nRTS. 08 = I2C2_SCL. 10 = CCAP_PIXCLK. 11 = EPWM1_CH2. 12 = ETMC_TRACE_DATA2. 13 = TM2_EXT. 14 = CAN3_TXD. 16 = SPI3_CLK. 17 = PSIO0_CH2. 18 = KPI_COL2. 19 = SPI9_MISO.
[7:5]	**Reserved**	Reserved.
[4:0]	**PB12MFP**	**PB.12 Multi-function Pin Selection** 01 = EADC0_CH12, EADC1_CH12, DAC0_OUT, ACMP0_P2, ACMP1_P2. 02 = EBI_AD15. 03 = SC1_CLK. 04 = SPI0_MOSI. 05 = USCI0_CLK. 06 = UART0_RXD. 07 = UART3_nCTS. 08 = I2C2_SDA. 09 = SD0_nCD. 10 = CCAP_SCLK. 11 = EPWM1_CH3. 12 = ETMC_TRACE_DATA3. 13 = TM3_EXT. 14 = CAN3_RXD.

Bits	Description	
		16 = SPI3_SS. 17 = PSIO0_CH3. 18 = KPI_COL3.

GPIOC Multiple Function Control Register 0 (SYS_GPC_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPC_MFP0	SYS_BA+0x520	R/W	GPIOC Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PC3MFP				
23	22	21	20	19	18	17	16
Reserved			PC2MFP				
15	14	13	12	11	10	9	8
Reserved			PC1MFP				
7	6	5	4	3	2	1	0
Reserved			PC0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PC3MFP**	**PC.3 Multi-function Pin Selection** 02 = EBI_AD3. 03 = SPIM_SS. 04 = QSPI0_SS. 05 = SC1_PWR. 06 = I2S0_MCLK. 07 = SPI1_MISO. 08 = UART2_nRTS. 09 = I2C0_SMBAL. 10 = CAN1_TXD. 11 = UART3_TXD. 12 = EPWM1_CH2. 13 = CCAP_DATA3. 14 = QSPI1_MISO0. 15 = I2C3_SCL. 16 = HBI_nCS. 17 = PSIO0_CH2. 18 = KPI_ROW2. 19 = SPI7_SS.
[23:21]	**Reserved**	Reserved.
[20:16]	**PC2MFP**	**PC.2 Multi-function Pin Selection** 02 = EBI_AD2. 03 = SPIM_CLK. 04 = QSPI0_CLK. 05 = SC1_RST. 06 = I2S0_DI. 07 = SPI1_MOSI.

Bits	Description	
		08 = UART2_nCTS. 09 = I2C0_SMBSUS. 10 = CAN1_RXD. 11 = UART3_RXD. 12 = EPWM1_CH3. 13 = CCAP_DATA2. 14 = QSPI1_MOSI0. 15 = I2C3_SDA. 16 = HBI_nRESET. 17 = PSIO0_CH3. 18 = KPI_ROW3. 19 = SPI7_CLK.
[15:13]	**Reserved**	Reserved.
[12:8]	**PC1MFP**	**PC.1 Multi-function Pin Selection** 02 = EBI_AD1. 03 = SPIM_MISO. 04 = QSPI0_MISO0. 05 = SC1_DAT. 06 = I2S0_DO. 07 = SPI1_CLK. 08 = UART2_TXD. 09 = I2C0_SCL. 10 = CAN2_TXD. 12 = EPWM1_CH4. 13 = CCAP_DATA1. 14 = ACMP0_O. 15 = EADC0_ST. 16 = HBI_RWDS. 17 = QSPI1_SS. 18 = KPI_ROW4. 19 = SPI7_MISO.
[7:5]	**Reserved**	Reserved.
[4:0]	**PC0MFP**	**PC.0 Multi-function Pin Selection** 02 = EBI_AD0. 03 = SPIM_MOSI. 04 = QSPI0_MOSI0. 05 = SC1_CLK. 06 = I2S0_LRCK. 07 = SPI1_SS. 08 = UART2_RXD. 09 = I2C0_SDA. 10 = CAN2_RXD. 12 = EPWM1_CH5. 13 = CCAP_DATA0. 14 = ACMP1_O. 15 = EADC1_ST.

Bits	Description	
		16 = HBI_D2. 17 = QSPI1_CLK. 18 = KPI_ROW5. 19 = SPI7_MOSI.

GPIOC Multiple Function Control Register 1 (SYS_GPC_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPC_MFP1	SYS_BA+0x524	R/W	GPIOC Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PC7MFP				
23	22	21	20	19	18	17	16
Reserved			PC6MFP				
15	14	13	12	11	10	9	8
Reserved			PC5MFP				
7	6	5	4	3	2	1	0
Reserved			PC4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PC7MFP**	**PC.7 Multi-function Pin Selection** 02 = EBI_AD9. 03 = EMAC_RMII_RXD0. 04 = SPI1_MISO. 05 = UART4_TXD. 06 = SC2_PWR. 07 = UART0_nCTS. 08 = I2C1_SMBAL. 09 = UART6_TXD. 10 = ACMP2_WLAT. 11 = EPWM1_CH2. 12 = BPWM1_CH0. 13 = CAN3_TXD. 14 = TM0. 15 = INT3. 18 = KPI_COL3. 19 = SPI6_MISO.
[23:21]	**Reserved**	Reserved.
[20:16]	**PC6MFP**	**PC.6 Multi-function Pin Selection** 02 = EBI_AD8. 03 = EMAC_RMII_RXD1. 04 = SPI1_MOSI. 05 = UART4_RXD. 06 = SC2_RST. 07 = UART0_nRTS. 08 = I2C1_SMBSUS. 09 = UART6_RXD.

Bits	Description	
		10 = ACMP3_WLAT. 11 = EPWM1_CH3. 12 = BPWM1_CH1. 13 = CAN3_RXD. 14 = TM1. 15 = INT2. 18 = KPI_COL2. 19 = SPI6_MOSI.
[15:13]	**Reserved**	Reserved.
[12:8]	**PC5MFP**	**PC.5 Multi-function Pin Selection** 02 = EBI_AD5. 03 = SPIM_D2. 04 = QSPI0_MISO1. 08 = UART2_TXD. 09 = I2C1_SCL. 10 = CAN0_TXD. 11 = UART4_TXD. 12 = EPWM1_CH0. 13 = CCAP_DATA5. 14 = QSPI1_SS. 15 = I2C3_SMBAL. 16 = HBI_nCK. 17 = PSIO0_CH0. 18 = KPI_ROW0.
[7:5]	**Reserved**	Reserved.
[4:0]	**PC4MFP**	**PC.4 Multi-function Pin Selection** 02 = EBI_AD4. 03 = SPIM_D3. 04 = QSPI0_MOSI1. 05 = SC1_nCD. 06 = I2S0_BCLK. 07 = SPI1_I2SMCLK. 08 = UART2_RXD. 09 = I2C1_SDA. 10 = CAN0_RXD. 11 = UART4_RXD. 12 = EPWM1_CH1. 13 = CCAP_DATA4. 14 = QSPI1_CLK. 15 = I2C3_SMBSUS. 16 = HBI_CK. 17 = PSIO0_CH1. 18 = KPI_ROW1.

GPIOC Multiple Function Control Register 2 (SYS_GPC_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPC_MFP2	SYS_BA+0x528	R/W	GPIOC Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PC11MFP				
23	22	21	20	19	18	17	16
Reserved			PC10MFP				
15	14	13	12	11	10	9	8
Reserved			PC9MFP				
7	6	5	4	3	2	1	0
Reserved			PC8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PC11MFP**	**PC.11 Multi-function Pin Selection** 01 = EADC2_CH12, ACMP3_P3. 02 = EBI_ADR5. 03 = UART0_RXD. 04 = I2C0_SDA. 05 = UART6_RXD. 06 = SPI3_MOSI. 10 = I2C4_SDA. 11 = ECAP1_IC1. 12 = EPWM1_CH1. 14 = ACMP1_O.
[23:21]	**Reserved**	Reserved.
[20:16]	**PC10MFP**	**PC.10 Multi-function Pin Selection** 01 = EADC2_CH11, ACMP3_P2. 02 = EBI_ADR6. 05 = UART6_nRTS. 06 = SPI3_CLK. 07 = UART3_TXD. 09 = CAN1_TXD. 10 = I2C4_SMBAL. 11 = ECAP1_IC0. 12 = EPWM1_CH2. 14 = EADC1_ST.
[15:13]	**Reserved**	Reserved.
[12:8]	**PC9MFP**	**PC.9 Multi-function Pin Selection** 01 = EADC2_CH10, ACMP3_P1.

Bits	Description	
		02 = EBI_ADR7. 05 = UART6_nCTS. 06 = SPI3_SS. 07 = UART3_RXD. 09 = CAN1_RXD. 10 = I2C4_SMBSUS. 12 = EPWM1_CH3. 14 = EADC1_ST.
[7:5]	**Reserved**	Reserved.
[4:0]	**PC8MFP**	**PC.8 Multi-function Pin Selection** 02 = EBI_ADR16. 03 = EMAC_RMII_REFCLK. 04 = I2C0_SDA. 05 = UART4_nCTS. 08 = UART1_RXD. 11 = EPWM1_CH1. 12 = BPWM1_CH4. 18 = KPI_COL4.

GPIOC Multiple Function Control Register 3 (SYS_GPC_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPC_MFP3	SYS_BA+0x52C	R/W	GPIOC Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PC15MFP				
23	22	21	20	19	18	17	16
Reserved			PC14MFP				
15	14	13	12	11	10	9	8
Reserved			PC13MFP				
7	6	5	4	3	2	1	0
Reserved			PC12MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PC15MFP**	PC.15 Multi-function Pin Selection
[20:16]	**PC14MFP**	**PC.14 Multi-function Pin Selection** 02 = EBI_AD11. 03 = SC1_nCD. 04 = SPI0_I2SMCLK. 05 = USCI0_CTL0. 06 = QSPI0_CLK. 10 = ITM_TRACE_SWO. 11 = EPWM0_SYNC_IN. 12 = ETMC_TRACE_CLK. 13 = TM1. 14 = USB_VBUS_ST. 15 = HSUSB_VBUS_ST. 19 = SPI9_MOSI.
[15:13]	**Reserved**	Reserved.
[12:8]	**PC13MFP**	**PC.13 Multi-function Pin Selection** 01 = EADC1_CH3, EADC2_CH3. 02 = EBI_ADR10. 03 = SC2_nCD. 04 = SPI2_I2SMCLK. 05 = CAN1_TXD. 06 = USCI0_CTL0. 07 = UART2_TXD. 08 = UART8_nCTS. 09 = BPWM0_CH4. 13 = CLKO.

Bits	Description	
		14 = EADC0_ST. 19 = SPI9_SS.
[7:5]	**Reserved**	Reserved.
[4:0]	**PC12MFP**	**PC.12 Multi-function Pin Selection** 01 = EADC2_CH13. 02 = EBI_ADR4. 03 = UART0_TXD. 04 = I2C0_SCL. 05 = UART6_TXD. 06 = SPI3_MISO. 09 = SC0_nCD. 10 = I2C4_SCL. 11 = ECAP1_IC2. 12 = EPWM1_CH0. 14 = ACMP0_O.

GPIOD Multiple Function Control Register 0 (SYS_GPD_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPD_MFP0	SYS_BA+0x530	R/W	GPIOD Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PD3MFP				
23	22	21	20	19	18	17	16
Reserved			PD2MFP				
15	14	13	12	11	10	9	8
Reserved			PD1MFP				
7	6	5	4	3	2	1	0
Reserved			PD0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PD3MFP**	**PD.3 Multi-function Pin Selection** 02 = EBI_AD10. 03 = USCI0_CTL1. 04 = SPI0_SS. 05 = UART3_nRTS. 07 = SC2_PWR. 08 = SC1_nCD. 09 = UART0_TXD. 10 = I2S1_BCLK. 13 = QEI3_A.
[23:21]	**Reserved**	Reserved.
[20:16]	**PD2MFP**	**PD.2 Multi-function Pin Selection** 02 = EBI_AD11. 03 = USCI0_DAT1. 04 = SPI0_CLK. 05 = UART3_nCTS. 07 = SC2_RST. 09 = UART0_RXD. 10 = I2S1_MCLK. 13 = QEI3_B.
[15:13]	**Reserved**	Reserved.
[12:8]	**PD1MFP**	**PD.1 Multi-function Pin Selection** 02 = EBI_AD12. 03 = USCI0_DAT0. 04 = SPI0_MISO. 05 = UART3_TXD.

Bits	Description	
		06 = I2C2_SCL. 07 = SC2_DAT. 10 = I2S1_DI. 12 = QEI2_INDEX. 13 = ECAP2_IC0.
[7:5]	**Reserved**	Reserved.
[4:0]	**PD0MFP**	**PD.0 Multi-function Pin Selection** 02 = EBI_AD13. 03 = USCI0_CLK. 04 = SPI0_MOSI. 05 = UART3_RXD. 06 = I2C2_SDA. 07 = SC2_CLK. 10 = I2S1_DO. 12 = QEI2_A. 13 = ECAP2_IC1. 14 = TM2.

GPIOD Multiple Function Control Register 1 (SYS_GPD_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPD_MFP1	SYS_BA+0x534	R/W	GPIOD Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PD7MFP				
23	22	21	20	19	18	17	16
Reserved			PD6MFP				
15	14	13	12	11	10	9	8
Reserved			PD5MFP				
7	6	5	4	3	2	1	0
Reserved			PD4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PD7MFP**	**PD.7 Multi-function Pin Selection** 02 = EBI_AD4. 03 = UART1_TXD. 04 = I2C0_SCL. 05 = SPI1_MISO. 06 = QSPI1_MISO0. 07 = CCAP_HSYNC. 08 = SC1_PWR. 16 = HBI_D5. 17 = PSIO0_CH4.
[23:21]	**Reserved**	Reserved.
[20:16]	**PD6MFP**	**PD.6 Multi-function Pin Selection** 02 = EBI_AD5. 03 = UART1_RXD. 04 = I2C0_SDA. 05 = SPI1_MOSI. 06 = QSPI1_MOSI0. 08 = SC1_RST. 14 = ACMP0_O. 15 = EADC0_ST. 16 = HBI_D6. 17 = PSIO0_CH5.
[15:13]	**Reserved**	Reserved.
[12:8]	**PD5MFP**	**PD.5 Multi-function Pin Selection** 04 = I2C1_SCL. 05 = SPI1_CLK.

Bits	Description	
		08 = SC1_DAT. 14 = ACMP1_O. 15 = EADC1_ST. 16 = HBI_D7. 17 = PSIO0_CH6.
[7:5]	**Reserved**	Reserved.
[4:0]	**PD4MFP**	**PD.4 Multi-function Pin Selection** 03 = USCI0_CTL0. 04 = I2C1_SDA. 05 = SPI1_SS. 08 = SC1_CLK. 14 = USB_VBUS_ST. 17 = PSIO0_CH7.

GPIOD Multiple Function Control Register 2 (SYS_GPD_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPD_MFP2	SYS_BA+0x538	R/W	GPIOD Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PD11MFP				
23	22	21	20	19	18	17	16
Reserved			PD10MFP				
15	14	13	12	11	10	9	8
Reserved			PD9MFP				
7	6	5	4	3	2	1	0
Reserved			PD8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PD11MFP**	**PD.11 Multi-function Pin Selection** 01 = EADC1_CH1, EADC2_CH1. 02 = EBI_nCS1. 03 = UART1_TXD. 04 = CAN0_TXD. 08 = UART8_TXD. 10 = QEI0_A. 11 = ECAP3_IC1. 15 = INT6. 19 = SPI9_MISO.
[23:21]	**Reserved**	Reserved.
[20:16]	**PD10MFP**	**PD.10 Multi-function Pin Selection** 01 = EADC1_CH0, EADC2_CH0. 02 = EBI_nCS2. 03 = UART1_RXD. 04 = CAN0_RXD. 08 = UART8_RXD. 10 = QEI0_B. 11 = ECAP3_IC2. 15 = INT7. 19 = SPI9_MOSI.
[15:13]	**Reserved**	Reserved.
[12:8]	**PD9MFP**	**PD.9 Multi-function Pin Selection** 02 = EBI_AD7. 03 = I2C2_SCL. 04 = UART2_nCTS.

Bits	Description	
		05 = UART7_TXD. 06 = CAN2_TXD. 17 = PSIO0_CH2.
[7:5]	**Reserved**	Reserved.
[4:0]	**PD8MFP**	**PD.8 Multi-function Pin Selection** 02 = EBI_AD6. 03 = I2C2_SDA. 04 = UART2_nRTS. 05 = UART7_RXD. 06 = CAN2_RXD. 17 = PSIO0_CH3.

GPIOD Multiple Function Control Register 3 (SYS_GPD_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPD_MFP3	SYS_BA+0x53C	R/W	GPIOD Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PD15MFP				
23	22	21	20	19	18	17	16
Reserved			PD14MFP				
15	14	13	12	11	10	9	8
Reserved			PD13MFP				
7	6	5	4	3	2	1	0
Reserved			PD12MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PD15MFP**	PD.15 Multi-function Pin Selection
[20:16]	**PD14MFP**	**PD.14 Multi-function Pin Selection** 02 = EBI_nCS0. 03 = SPI3_I2SMCLK. 04 = SC1_nCD. 05 = SPI0_I2SMCLK. 10 = I2S1_BCLK. 11 = EPWM0_CH4.
[15:13]	**Reserved**	Reserved.
[12:8]	**PD13MFP**	**PD.13 Multi-function Pin Selection** 02 = EBI_AD10. 03 = SD0_nCD. 04 = SPI0_I2SMCLK. 05 = SPI1_I2SMCLK. 06 = QSPI1_MOSI0. 07 = SC2_nCD. 08 = SD1_CLK. 09 = UART6_RXD. 10 = I2S1_LRCK. 11 = BPWM0_CH0. 12 = QEI2_B. 13 = ECAP2_IC2. 14 = CLKO. 15 = EADC0_ST. 19 = QSPI1_MOSI1.
[7:5]	**Reserved**	Reserved.

Bits	Description	
[4:0]	**PD12MFP**	**PD.12 Multi-function Pin Selection** 01 = EADC1_CH2, EADC2_CH2. 02 = EBI_nCS0. 05 = CAN1_RXD. 07 = UART2_RXD. 08 = UART8_nRTS. 09 = BPWM0_CH5. 10 = QEI0_INDEX. 11 = ECAP3_IC0. 13 = CLKO. 14 = EADC0_ST. 15 = INT5. 19 = SPI9_CLK.

GPIOE Multiple Function Control Register 0 (SYS_GPE_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPE_MFP0	SYS_BA+0x540	R/W	GPIOE Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PE3MFP				
23	22	21	20	19	18	17	16
Reserved			PE2MFP				
15	14	13	12	11	10	9	8
Reserved			PE1MFP				
7	6	5	4	3	2	1	0
Reserved			PE0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PE3MFP**	**PE.3 Multi-function Pin Selection** 02 = EBI_MCLK. 03 = SD0_DAT1. 04 = SPIM_MISO. 05 = SPI3_MISO. 06 = SC0_DAT. 07 = USCI0_DAT0. 08 = UART6_nRTS. 09 = UART7_TXD. 10 = UART8_nCTS. 11 = QEI0_A. 12 = EPWM0_CH4. 13 = BPWM0_CH1.
[23:21]	**Reserved**	Reserved.
[20:16]	**PE2MFP**	**PE.2 Multi-function Pin Selection** 02 = EBI_ALE. 03 = SD0_DAT0. 04 = SPIM_MOSI. 05 = SPI3_MOSI. 06 = SC0_CLK. 07 = USCI0_CLK. 08 = UART6_nCTS. 09 = UART7_RXD. 10 = UART8_nRTS. 11 = QEI0_B. 12 = EPWM0_CH5. 13 = BPWM0_CH0.

Bits	Description	
[15:13]	**Reserved**	Reserved.
[12:8]	**PE1MFP**	**PE.1 Multi-function Pin Selection** 02 = EBI_AD10. 03 = QSPI0_MISO0. 04 = SC2_DAT. 05 = I2S0_BCLK. 06 = SPI1_MISO. 07 = UART3_TXD. 08 = I2C1_SCL. 09 = UART4_nCTS. 10 = UART8_TXD.
[7:5]	**Reserved**	Reserved.
[4:0]	**PE0MFP**	**PE.0 Multi-function Pin Selection** 02 = EBI_AD11. 03 = QSPI0_MOSI0. 04 = SC2_CLK. 05 = I2S0_MCLK. 06 = SPI1_MOSI. 07 = UART3_RXD. 08 = I2C1_SDA. 09 = UART4_nRTS. 10 = UART8_RXD.

GPIOE Multiple Function Control Register 1 (SYS_GPE_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPE_MFP1	SYS_BA+0x544	R/W	GPIOE Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PE7MFP				
23	22	21	20	19	18	17	16
Reserved			PE6MFP				
15	14	13	12	11	10	9	8
Reserved			PE5MFP				
7	6	5	4	3	2	1	0
Reserved			PE4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PE7MFP**	**PE.7 Multi-function Pin Selection** 03 = SD0_CMD. 04 = SPIM_D2. 08 = UART5_TXD. 09 = CAN1_TXD. 10 = UART9_nCTS. 11 = QEI1_INDEX. 12 = EPWM0_CH0. 13 = BPWM0_CH5. 14 = ACMP2_O. 17 = PSIO0_CH0.
[23:21]	**Reserved**	Reserved.
[20:16]	**PE6MFP**	**PE.6 Multi-function Pin Selection** 03 = SD0_CLK. 04 = SPIM_D3. 05 = SPI3_I2SMCLK. 06 = SC0_nCD. 07 = USCI0_CTL0. 08 = UART5_RXD. 09 = CAN1_RXD. 10 = UART9_nRTS. 11 = QEI1_A. 12 = EPWM0_CH1. 13 = BPWM0_CH4. 14 = ACMP3_O. 17 = PSIO0_CH1.

Bits	Description	
[15:13]	**Reserved**	Reserved.
[12:8]	**PE5MFP**	**PE.5 Multi-function Pin Selection** 02 = EBI_nRD. 03 = SD0_DAT3. 04 = SPIM_SS. 05 = SPI3_SS. 06 = SC0_PWR. 07 = USCI0_CTL1. 08 = UART6_TXD. 09 = UART7_nRTS. 10 = UART9_TXD. 11 = QEI1_B. 12 = EPWM0_CH2. 13 = BPWM0_CH3. 17 = PSIO0_CH2.
[7:5]	**Reserved**	Reserved.
[4:0]	**PE4MFP**	**PE.4 Multi-function Pin Selection** 02 = EBI_nWR. 03 = SD0_DAT2. 04 = SPIM_CLK. 05 = SPI3_CLK. 06 = SC0_RST. 07 = USCI0_DAT1. 08 = UART6_RXD. 09 = UART7_nCTS. 10 = UART9_RXD. 11 = QEI0_INDEX. 12 = EPWM0_CH3. 13 = BPWM0_CH2. 17 = PSIO0_CH3.

GPIOE Multiple Function Control Register 2 (SYS_GPE_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPE_MFP2	SYS_BA+0x548	R/W	GPIOE Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PE11MFP				
23	22	21	20	19	18	17	16
Reserved			PE10MFP				
15	14	13	12	11	10	9	8
Reserved			PE9MFP				
7	6	5	4	3	2	1	0
Reserved			PE8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PE11MFP**	**PE.11 Multi-function Pin Selection** 02 = EBI_ADR13. 03 = EMAC_RMII_TXD1. 04 = I2S0_DO. 05 = SPI2_SS. 07 = UART3_RXD. 08 = UART1_nCTS. 10 = EPWM0_CH3. 11 = EPWM1_BRAKE1. 13 = ECAP1_IC2. 14 = TRACE_DATA0. 18 = KPI_COL7.
[23:21]	**Reserved**	Reserved.
[20:16]	**PE10MFP**	**PE.10 Multi-function Pin Selection** 02 = EBI_ADR12. 03 = EMAC_RMII_TXD0. 04 = I2S0_DI. 05 = SPI2_MOSI. 07 = UART3_TXD. 10 = EPWM0_CH2. 11 = EPWM1_BRAKE0. 12 = ECAP0_IC2. 13 = QEI2_B. 14 = TRACE_DATA1. 15 = ECAP3_IC2.
[15:13]	**Reserved**	Reserved.

Bits	Description	
[12:8]	**PE9MFP**	**PE.9 Multi-function Pin Selection** 02 = EBI_ADR11. 03 = EMAC_RMII_MDIO. 04 = I2S0_MCLK. 05 = SPI2_MISO. 07 = UART2_RXD. 10 = EPWM0_CH1. 11 = EPWM0_BRAKE1. 12 = ECAP0_IC1. 13 = QEI2_A. 14 = TRACE_DATA2. 15 = ECAP3_IC1.
[7:5]	**Reserved**	Reserved.
[4:0]	**PE8MFP**	**PE.8 Multi-function Pin Selection** 02 = EBI_ADR10. 03 = EMAC_RMII_MDC. 04 = I2S0_BCLK. 05 = SPI2_CLK. 07 = UART2_TXD. 10 = EPWM0_CH0. 11 = EPWM0_BRAKE0. 12 = ECAP0_IC0. 13 = QEI2_INDEX. 14 = TRACE_DATA3. 15 = ECAP3_IC0.

GPIOE Multiple Function Control Register 3 (SYS_GPE_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPE_MFP3	SYS_BA+0x54C	R/W	GPIOE Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PE15MFP				
23	22	21	20	19	18	17	16
Reserved			PE14MFP				
15	14	13	12	11	10	9	8
Reserved			PE13MFP				
7	6	5	4	3	2	1	0
Reserved			PE12MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PE15MFP**	**PE.15 Multi-function Pin Selection** 02 = EBI_AD9. 03 = UART2_RXD. 04 = CAN0_RXD. 06 = UART6_RXD. 17 = PSIO0_CH1.
[23:21]	**Reserved**	Reserved.
[20:16]	**PE14MFP**	**PE.14 Multi-function Pin Selection** 02 = EBI_AD8. 03 = UART2_TXD. 04 = CAN0_TXD. 05 = SD1_nCD. 06 = UART6_TXD. 17 = PSIO0_CH0.
[15:13]	**Reserved**	Reserved.
[12:8]	**PE13MFP**	**PE.13 Multi-function Pin Selection** 02 = EBI_ADR15. 03 = EMAC_PPS. 04 = I2C0_SCL. 05 = UART4_nRTS. 08 = UART1_TXD. 10 = EPWM0_CH5. 11 = EPWM1_CH0. 12 = BPWM1_CH5. 13 = ECAP1_IC0. 14 = ITM_TRACE_SWO.

Bits	Description	
		18 = KPI_COL5.
[7:5]	**Reserved**	Reserved.
[4:0]	**PE12MFP**	**PE.12 Multi-function Pin Selection** 02 = EBI_ADR14. 03 = EMAC_RMII_TXEN. 04 = I2S0_LRCK. 05 = SPI2_I2SMCLK. 08 = UART1_nRTS. 10 = EPWM0_CH4. 13 = ECAP1_IC1. 14 = TRACE_CLK. 18 = KPI_COL6.

GPIOF Multiple Function Control Register 0 (SYS_GPF_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPF_MFP0	SYS_BA+0x550	R/W	GPIOF Multiple Function Control Register 0	0x0000_0E0E

31	30	29	28	27	26	25	24
Reserved			PF3MFP				
23	22	21	20	19	18	17	16
Reserved			PF2MFP				
15	14	13	12	11	10	9	8
Reserved			PF1MFP				
7	6	5	4	3	2	1	0
Reserved			PF0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PF3MFP**	**PF.3 Multi-function Pin Selection** 02 = EBI_nCS0. 03 = UART0_TXD. 04 = I2C0_SCL. 07 = UART9_TXD. 10 = XT1_IN. 11 = BPWM1_CH0. 12 = I2C4_SMBAL. 13 = ACMP2_O. 15 = EADC2_ST.
[23:21]	**Reserved**	Reserved.
[20:16]	**PF2MFP**	**PF.2 Multi-function Pin Selection** 02 = EBI_nCS1. 03 = UART0_RXD. 04 = I2C0_SDA. 05 = QSPI0_CLK. 07 = UART9_RXD. 10 = XT1_OUT. 11 = BPWM1_CH1. 12 = I2C4_SMBSUS. 13 = ACMP3_O.
[15:13]	**Reserved**	Reserved.
[12:8]	**PF1MFP**	**PF.1 Multi-function Pin Selection** 02 = UART1_RXD. 03 = I2C1_SDA. 04 = UART0_RXD.

Bits	Description	
		05 = SC1_CLK. 06 = I2S0_LRCK. 08 = UART2_RXD. 09 = I2C0_SDA. 10 = CAN2_RXD. 11 = EPWM1_CH5. 12 = BPWM1_CH1. 13 = ACMP1_O. 14 = ICE_CLK. 15 = EADC1_ST. 19 = QSPI1_MOSI0.
[7:5]	**Reserved**	Reserved.
[4:0]	**PF0MFP**	**PF.0 Multi-function Pin Selection** 02 = UART1_TXD. 03 = I2C1_SCL. 04 = UART0_TXD. 05 = SC1_DAT. 06 = I2S0_DO. 08 = UART2_TXD. 09 = I2C0_SCL. 10 = CAN2_TXD. 11 = EPWM1_CH4. 12 = BPWM1_CH0. 13 = ACMP0_O. 14 = ICE_DAT. 15 = EADC0_ST. 19 = QSPI1_MISO0.

GPIOF Multiple Function Control Register 1 (SYS_GPF_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPF_MFP1	SYS_BA+0x554	R/W	GPIOF Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PF7MFP				
23	22	21	20	19	18	17	16
Reserved			PF6MFP				
15	14	13	12	11	10	9	8
Reserved			PF5MFP				
7	6	5	4	3	2	1	0
Reserved			PF4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PF7MFP**	**PF.7 Multi-function Pin Selection** 02 = EBI_ADR18. 03 = SC0_DAT. 04 = I2S0_DO. 05 = SPI0_MISO. 06 = UART4_TXD. 07 = CCAP_DATA0. 08 = CAN2_TXD. 10 = TAMPER1. 19 = SPI5_SS.
[23:21]	**Reserved**	Reserved.
[20:16]	**PF6MFP**	**PF.6 Multi-function Pin Selection** 02 = EBI_ADR19. 03 = SC0_CLK. 04 = I2S0_LRCK. 05 = SPI0_MOSI. 06 = UART4_RXD. 07 = EBI_nCS0. 08 = CAN2_RXD. 09 = SPI3_I2SMCLK. 10 = TAMPER0. 13 = QEI2_INDEX. 14 = ITM_TRACE_SWO. 19 = SPI5_CLK.
[15:13]	**Reserved**	Reserved.
[12:8]	**PF5MFP**	**PF.5 Multi-function Pin Selection**

Bits	Description	
		02 = UART2_RXD. 03 = EBI_AD1. 04 = UART2_nCTS. 07 = EPWM0_CH0. 08 = BPWM0_CH4. 09 = EPWM0_SYNC_OUT. 10 = X32_IN. 11 = EADC0_ST. 12 = I2C4_SCL. 13 = QEI2_A. 19 = SPI5_MOSI.
[7:5]	**Reserved**	Reserved.
[4:0]	**PF4MFP**	**PF.4 Multi-function Pin Selection** 02 = UART2_TXD. 03 = EBI_AD0. 04 = UART2_nRTS. 07 = EPWM0_CH1. 08 = BPWM0_CH5. 10 = X32_OUT. 11 = EADC1_ST. 12 = I2C4_SDA. 13 = QEI2_B. 19 = SPI5_MISO.

GPIOF Multiple Function Control Register 2 (SYS_GPF_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPF_MFP2	SYS_BA+0x558	R/W	GPIOF Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PF11MFP				
23	22	21	20	19	18	17	16
Reserved			PF10MFP				
15	14	13	12	11	10	9	8
Reserved			PF9MFP				
7	6	5	4	3	2	1	0
Reserved			PF8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PF11MFP**	**PF.11 Multi-function Pin Selection** 02 = EBI_ADR14. 03 = SPI2_MOSI. 06 = UART5_TXD. 07 = CCAP_DATA4. 08 = CAN3_TXD. 10 = TAMPER5. 11 = UART9_nCTS. 13 = TM3.
[23:21]	**Reserved**	Reserved.
[20:16]	**PF10MFP**	**PF.10 Multi-function Pin Selection** 02 = EBI_ADR15. 03 = SC0_nCD. 04 = I2S0_BCLK. 05 = SPI0_I2SMCLK. 06 = UART5_RXD. 07 = CCAP_DATA3. 08 = CAN3_RXD. 10 = TAMPER4. 11 = UART9_nRTS.
[15:13]	**Reserved**	Reserved.
[12:8]	**PF9MFP**	**PF.9 Multi-function Pin Selection** 02 = EBI_ADR16. 03 = SC0_PWR. 04 = I2S0_MCLK. 05 = SPI0_SS.

Bits	Description	
		06 = UART5_nRTS. 07 = CCAP_DATA2. 08 = CAN1_TXD. 10 = TAMPER3. 11 = UART9_TXD.
[7:5]	**Reserved**	Reserved.
[4:0]	**PF8MFP**	**PF.8 Multi-function Pin Selection** 02 = EBI_ADR17. 03 = SC0_RST. 04 = I2S0_DI. 05 = SPI0_CLK. 06 = UART5_nCTS. 07 = CCAP_DATA1. 08 = CAN1_RXD. 10 = TAMPER2. 11 = UART9_RXD.

GPIOF Multiple Function Control Register 3 (SYS_GPF_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPF_MFP3	SYS_BA+0x55C	R/W	GPIOF Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PF15MFP				
23	22	21	20	19	18	17	16
Reserved			PF14MFP				
15	14	13	12	11	10	9	8
Reserved			PF13MFP				
7	6	5	4	3	2	1	0
Reserved			PF12MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PF15MFP**	PF.15 Multi-function Pin Selection
[23:21]	**Reserved**	Reserved.
[20:16]	**PF14MFP**	PF.14 Multi-function Pin Selection
[15:13]	**Reserved**	Reserved.
[12:8]	**PF13MFP**	PF.13 Multi-function Pin Selection
[7:5]	**Reserved**	Reserved.
[4:0]	**PF12MFP**	PF.12 Multi-function Pin Selection

GPIOG Multiple Function Control Register 0 (SYS_GPG_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPG_MFP0	SYS_BA+0x560	R/W	GPIOG Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PG3MFP				
23	22	21	20	19	18	17	16
Reserved			PG2MFP				
15	14	13	12	11	10	9	8
Reserved			PG1MFP				
7	6	5	4	3	2	1	0
Reserved			PG0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PG3MFP**	**PG.3 Multi-function Pin Selection** 02 = EBI_ADR12. 03 = SPI2_CLK. 04 = I2C0_SMBSUS. 05 = I2C1_SDA. 07 = CCAP_DATA6. 09 = I2C3_SMBSUS. 13 = TM1.
[23:21]	**Reserved**	Reserved.
[20:16]	**PG2MFP**	**PG.2 Multi-function Pin Selection** 02 = EBI_ADR11. 03 = SPI2_SS. 04 = I2C0_SMBAL. 05 = I2C1_SCL. 07 = CCAP_DATA7. 09 = I2C3_SMBAL. 13 = TM0.
[15:13]	**Reserved**	Reserved.
[12:8]	**PG1MFP**	**PG.1 Multi-function Pin Selection** 02 = EBI_ADR9. 03 = SPI2_I2SMCLK. 04 = I2C0_SDA. 05 = I2C1_SMBSUS. 06 = UART2_TXD. 07 = CAN1_RXD. 08 = UART1_RXD.

Bits	Description	
		09 = I2C3_SDA.
[7:5]	**Reserved**	Reserved.
[4:0]	**PG0MFP**	**PG.0 Multi-function Pin Selection** 02 = EBI_ADR8. 04 = I2C0_SCL. 05 = I2C1_SMBAL. 06 = UART2_RXD. 07 = CAN1_TXD. 08 = UART1_TXD. 09 = I2C3_SCL.

GPIOG Multiple Function Control Register 1 (SYS_GPG_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPG_MFP1	SYS_BA+0x564	R/W	GPIOG Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PG7MFP				
23	22	21	20	19	18	17	16
Reserved			PG6MFP				
15	14	13	12	11	10	9	8
Reserved			PG5MFP				
7	6	5	4	3	2	1	0
Reserved			PG4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PG7MFP**	**PG.7 Multi-function Pin Selection** 02 = EBI_nWRL. 03 = SPI3_MISO. 04 = SC1_DAT. 08 = I2C3_SCL. 10 = I2S1_DO. 11 = EPWM0_CH1.
[23:21]	**Reserved**	Reserved.
[20:16]	**PG6MFP**	**PG.6 Multi-function Pin Selection** 02 = EBI_nCS2. 03 = SPI3_CLK. 04 = SC1_RST. 08 = I2C3_SMBSUS. 10 = I2S1_DI. 11 = EPWM0_CH2.
[15:13]	**Reserved**	Reserved.
[12:8]	**PG5MFP**	**PG.5 Multi-function Pin Selection** 02 = EBI_nCS1. 03 = SPI3_SS. 04 = SC1_PWR. 08 = I2C3_SMBAL. 10 = I2S1_MCLK. 11 = EPWM0_CH3.
[7:5]	**Reserved**	Reserved.
[4:0]	**PG4MFP**	**PG.4 Multi-function Pin Selection**

Bits	Description	
		02 = EBI_ADR13. 03 = SPI2_MISO. 07 = CCAP_DATA5. 13 = TM2.

GPIOG Multiple Function Control Register 2 (SYS_GPG_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPG_MFP2	SYS_BA+0x568	R/W	GPIOG Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PG11MFP				
23	22	21	20	19	18	17	16
Reserved			PG10MFP				
15	14	13	12	11	10	9	8
Reserved			PG9MFP				
7	6	5	4	3	2	1	0
Reserved			PG8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PG11MFP**	**PG.11 Multi-function Pin Selection** 02 = EBI_AD2. 03 = SD1_DAT1. 04 = SPIM_SS. 05 = QSPI1_SS. 06 = UART7_TXD. 07 = CCAP_SFIELD. 08 = I2C4_SMBAL. 09 = ECAP2_IC2. 12 = BPWM0_CH3. 16 = HBI_D0. 19 = SPI8_MOSI.
[23:21]	**Reserved**	Reserved.
[20:16]	**PG10MFP**	**PG.10 Multi-function Pin Selection** 02 = EBI_AD1. 03 = SD1_DAT2. 04 = SPIM_D3. 05 = QSPI1_MOSI1. 07 = CCAP_SCLK. 08 = I2C4_SDA. 09 = ECAP2_IC1. 12 = BPWM0_CH4. 16 = HBI_D3. 19 = SPI8_CLK.
[15:13]	**Reserved**	Reserved.
[12:8]	**PG9MFP**	**PG.9 Multi-function Pin Selection**

Bits	Description	
		02 = EBI_AD0. 03 = SD1_DAT3. 04 = SPIM_D2. 05 = QSPI1_MISO1. 07 = CCAP_PIXCLK. 08 = I2C4_SCL. 09 = ECAP2_IC0. 12 = BPWM0_CH5. 16 = HBI_D4. 19 = SPI8_SS.
[7:5]	**Reserved**	Reserved.
[4:0]	**PG8MFP**	**PG.8 Multi-function Pin Selection** 02 = EBI_nWRH. 03 = SPI3_MOSI. 04 = SC1_CLK. 08 = I2C3_SDA. 10 = I2S1_LRCK. 11 = EPWM0_CH0.

GPIOG Multiple Function Control Register 3 (SYS_GPG_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPG_MFP3	SYS_BA+0x56C	R/W	GPIOG Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PG15MFP				
23	22	21	20	19	18	17	16
Reserved			PG14MFP				
15	14	13	12	11	10	9	8
Reserved			PG13MFP				
7	6	5	4	3	2	1	0
Reserved			PG12MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PG15MFP**	**PG.15 Multi-function Pin Selection** 03 = SD1_nCD. 14 = CLKO. 15 = EADC0_ST. 16 = HBI_D7. 19 = QSPI1_MISO1.
[23:21]	**Reserved**	Reserved.
[20:16]	**PG14MFP**	**PG.14 Multi-function Pin Selection** 02 = EBI_AD5. 03 = SD1_CLK. 04 = SPIM_MOSI. 05 = QSPI1_MOSI0. 06 = UART6_RXD. 12 = BPWM0_CH0. 16 = HBI_D6.
[15:13]	**Reserved**	Reserved.
[12:8]	**PG13MFP**	**PG.13 Multi-function Pin Selection** 02 = EBI_AD4. 03 = SD1_CMD. 04 = SPIM_MISO. 05 = QSPI1_MISO0. 06 = UART6_TXD. 07 = CCAP_HSYNC. 12 = BPWM0_CH1. 16 = HBI_D5.

Bits	Description	
[7:5]	**Reserved**	Reserved.
[4:0]	**PG12MFP**	**PG.12 Multi-function Pin Selection** 02 = EBI_AD3. 03 = SD1_DAT0. 04 = SPIM_CLK. 05 = QSPI1_CLK. 06 = UART7_RXD. 07 = CCAP_VSYNC. 08 = I2C4_SMBSUS. 12 = BPWM0_CH2. 16 = HBI_D1. 19 = SPI8_MISO.

GPIOH Multiple Function Control Register 0 (SYS_GPH_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPH_MFP0	SYS_BA+0x570	R/W	GPIOH Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PH3MFP				
23	22	21	20	19	18	17	16
Reserved			PH2MFP				
15	14	13	12	11	10	9	8
Reserved			PH1MFP				
7	6	5	4	3	2	1	0
Reserved			PH0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PH3MFP**	**PH.3 Multi-function Pin Selection** 02 = EBI_ADR4. 03 = SPI1_I2SMCLK. 04 = UART5_nCTS. 05 = UART4_RXD. 06 = I2C0_SDA. 13 = TM3_EXT.
[23:21]	**Reserved**	Reserved.
[20:16]	**PH2MFP**	**PH.2 Multi-function Pin Selection** 02 = EBI_ADR5. 04 = UART5_nRTS. 05 = UART4_TXD. 06 = I2C0_SCL. 13 = TM2_EXT.
[15:13]	**Reserved**	Reserved.
[12:8]	**PH1MFP**	**PH.1 Multi-function Pin Selection** 02 = EBI_ADR6. 04 = UART5_RXD. 13 = TM1_EXT.
[7:5]	**Reserved**	Reserved.
[4:0]	**PH0MFP**	**PH.0 Multi-function Pin Selection** 02 = EBI_ADR7. 04 = UART5_TXD. 13 = TM0_EXT.

GPIOH Multiple Function Control Register 1 (SYS_GPH_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPH_MFP1	SYS_BA+0x574	R/W	GPIOH Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PH7MFP				
23	22	21	20	19	18	17	16
Reserved			PH6MFP				
15	14	13	12	11	10	9	8
Reserved			PH5MFP				
7	6	5	4	3	2	1	0
Reserved			PH4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PH7MFP**	**PH.7 Multi-function Pin Selection** 02 = EBI_ADR0. 03 = SPI1_SS. 04 = UART7_RXD. 07 = UART9_nRTS.
[23:21]	**Reserved**	Reserved.
[20:16]	**PH6MFP**	**PH.6 Multi-function Pin Selection** 02 = EBI_ADR1. 03 = SPI1_CLK. 04 = UART7_TXD. 07 = UART9_nCTS.
[15:13]	**Reserved**	Reserved.
[12:8]	**PH5MFP**	**PH.5 Multi-function Pin Selection** 02 = EBI_ADR2. 03 = SPI1_MOSI. 04 = UART7_nCTS. 05 = UART6_RXD.
[7:5]	**Reserved**	Reserved.
[4:0]	**PH4MFP**	**PH.4 Multi-function Pin Selection** 02 = EBI_ADR3. 03 = SPI1_MISO. 04 = UART7_nRTS. 05 = UART6_TXD.

GPIOH Multiple Function Control Register 2 (SYS_GPH_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPH_MFP2	SYS_BA+0x578	R/W	GPIOH Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PH11MFP				
23	22	21	20	19	18	17	16
Reserved			PH10MFP				
15	14	13	12	11	10	9	8
Reserved			PH9MFP				
7	6	5	4	3	2	1	0
Reserved			PH8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PH11MFP**	**PH.11 Multi-function Pin Selection** 02 = EBI_AD15. 03 = QSPI0_MOSI1. 07 = UART4_RXD. 08 = UART0_RXD. 11 = EPWM0_CH5. 13 = UART9_RXD.
[23:21]	**Reserved**	Reserved.
[20:16]	**PH10MFP**	**PH.10 Multi-function Pin Selection** 02 = EBI_AD14. 03 = QSPI0_MISO1. 04 = SC2_nCD. 05 = I2S0_LRCK. 06 = SPI1_I2SMCLK. 07 = UART4_TXD. 08 = UART0_TXD. 13 = UART9_TXD.
[15:13]	**Reserved**	Reserved.
[12:8]	**PH9MFP**	**PH.9 Multi-function Pin Selection** 02 = EBI_AD13. 03 = QSPI0_SS. 04 = SC2_RST. 05 = I2S0_DO. 06 = SPI1_SS. 07 = UART3_nCTS. 08 = I2C1_SMBSUS.

Bits	Description	
		09 = I2C2_SDA. 10 = UART1_RXD. 13 = UART9_nRTS.
[7:5]	**Reserved**	Reserved.
[4:0]	**PH8MFP**	**PH.8 Multi-function Pin Selection** 02 = EBI_AD12. 03 = QSPI0_CLK. 04 = SC2_PWR. 05 = I2S0_DI. 06 = SPI1_CLK. 07 = UART3_nRTS. 08 = I2C1_SMBAL. 09 = I2C2_SCL. 10 = UART1_TXD. 13 = UART9_nCTS.

GPIOH Multiple Function Control Register 3 (SYS_GPH_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPH_MFP3	SYS_BA+0x57C	R/W	GPIOH Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			**PH15MFP**				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved			**PH14MFP**				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**PH13MFP**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**PH12MFP**				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PH15MFP**	**PH.15 Multi-function Pin Selection** 02 = EBI_AD3. 06 = QSPI1_CLK. 07 = CCAP_VSYNC. 16 = HBI_D4.
[23:21]	**Reserved**	Reserved.
[20:16]	**PH14MFP**	**PH.14 Multi-function Pin Selection** 02 = EBI_AD2. 06 = QSPI1_SS. 07 = CCAP_SFIELD. 16 = HBI_RWDS.
[15:13]	**Reserved**	Reserved.
[12:8]	**PH13MFP**	**PH.13 Multi-function Pin Selection** 02 = EBI_AD1. 03 = UART9_RXD. 06 = QSPI1_MOSI1. 07 = CCAP_SCLK. 10 = CAN3_RXD. 16 = HBI_CK.
[7:5]	**Reserved**	Reserved.
[4:0]	**PH12MFP**	**PH.12 Multi-function Pin Selection** 02 = EBI_AD0. 03 = UART9_TXD. 06 = QSPI1_MISO1. 07 = CCAP_PIXCLK.

Bits	Description	
		10 = CAN3_TXD. 16 = HBI_nCK.

GPIOI Multiple Function Control Register 0 (SYS_GPI_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPI_MFP0	SYS_BA+0x580	R/W	GPIOI Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PI3MFP				
23	22	21	20	19	18	17	16
Reserved			PI2MFP				
15	14	13	12	11	10	9	8
Reserved			PI1MFP				
7	6	5	4	3	2	1	0
Reserved			PI0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PI3MFP**	PI.3 Multi-function Pin Selection
[23:21]	**Reserved**	Reserved.
[20:16]	**PI2MFP**	PI.2 Multi-function Pin Selection
[15:13]	**Reserved**	Reserved.
[12:8]	**PI1MFP**	PI.1 Multi-function Pin Selection
[7:5]	**Reserved**	Reserved.
[4:0]	**PI0MFP**	PI.0 Multi-function Pin Selection

GPIOI Multiple Function Control Register 1 (SYS_GPI_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPI_MFP1	SYS_BA+0x584	R/W	GPIOI Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PI7MFP				
23	22	21	20	19	18	17	16
Reserved			PI6MFP				
15	14	13	12	11	10	9	8
Reserved			PI5MFP				
7	6	5	4	3	2	1	0
Reserved			PI4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PI7MFP**	**PI.7 Multi-function Pin Selection** 05 = SC1_PWR. 06 = I2S0_MCLK. 07 = SPI1_MISO. 08 = UART2_RXD. 09 = I2C1_SDA. 13 = CAN3_RXD. 15 = USB_VBUS_EN.
[23:21]	**Reserved**	Reserved.
[20:16]	**PI6MFP**	**PI.6 Multi-function Pin Selection** 05 = SC1_nCD. 06 = I2S0_BCLK. 07 = SPI1_I2SMCLK. 08 = UART2_TXD. 09 = I2C1_SCL. 13 = CAN3_TXD. 15 = USB_VBUS_ST.
[15:13]	**Reserved**	Reserved.
[12:8]	**PI5MFP**	PI.5 Multi-function Pin Selection
[7:5]	**Reserved**	Reserved.
[4:0]	**PI4MFP**	PI.4 Multi-function Pin Selection

GPIOI Multiple Function Control Register 2 (SYS_GPI_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPI_MFP2	SYS_BA+0x588	R/W	GPIOI Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PI11MFP				
23	22	21	20	19	18	17	16
Reserved			PI10MFP				
15	14	13	12	11	10	9	8
Reserved			PI9MFP				
7	6	5	4	3	2	1	0
Reserved			PI8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PI11MFP**	**PI.11 Multi-function Pin Selection** 08 = UART2_RXD. 09 = I2C0_SDA. 13 = CAN3_RXD.
[23:21]	**Reserved**	Reserved.
[20:16]	**PI10MFP**	**PI.10 Multi-function Pin Selection** 05 = SC1_CLK. 06 = I2S0_LRCK. 07 = SPI1_SS. 08 = UART2_TXD. 09 = I2C0_SCL. 13 = CAN3_TXD.
[15:13]	**Reserved**	Reserved.
[12:8]	**PI9MFP**	**PI.9 Multi-function Pin Selection** 05 = SC1_DAT. 06 = I2S0_DO. 07 = SPI1_CLK. 08 = UART2_nCTS. 09 = I2C0_SMBSUS. 13 = CAN2_RXD.
[7:5]	**Reserved**	Reserved.
[4:0]	**PI8MFP**	**PI.8 Multi-function Pin Selection** 05 = SC1_RST. 06 = I2S0_DI. 07 = SPI1_MOSI.

Bits	Description	
		08 = UART2_nRTS. 09 = I2C0_SMBAL. 13 = CAN2_TXD.

GPIOI Multiple Function Control Register 3 (SYS_GPI_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPI_MFP3	SYS_BA+0x58C	R/W	GPIOI Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PI15MFP				
23	22	21	20	19	18	17	16
Reserved			PI14MFP				
15	14	13	12	11	10	9	8
Reserved			PI13MFP				
7	6	5	4	3	2	1	0
Reserved			PI12MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PI15MFP**	**PI.15 Multi-function Pin Selection** 03 = SPIM_D3. 04 = QSPI0_CLK. 07 = UART8_nRTS. 10 = CAN1_RXD. 11 = UART3_RXD. 12 = EPWM1_CH3. 15 = I2C3_SDA.
[23:21]	**Reserved**	Reserved.
[20:16]	**PI14MFP**	**PI.14 Multi-function Pin Selection** 03 = SPIM_D2. 04 = QSPI0_SS. 07 = UART8_nCTS. 10 = CAN1_TXD. 11 = UART3_TXD. 12 = EPWM1_CH2. 15 = I2C3_SCL.
[15:13]	**Reserved**	Reserved.
[12:8]	**PI13MFP**	**PI.13 Multi-function Pin Selection** 03 = SPIM_MISO. 04 = QSPI0_MOSI1. 10 = CAN0_RXD. 11 = UART4_RXD. 12 = EPWM1_CH1. 15 = I2C3_SMBSUS.

Bits	Description	
[7:5]	**Reserved**	Reserved.
[4:0]	**PI12MFP**	**PI.12 Multi-function Pin Selection** 03 = SPIM_SS. 04 = QSPI0_MISO1. 10 = CAN0_TXD. 11 = UART4_TXD. 12 = EPWM1_CH0. 15 = I2C3_SMBAL.

GPIOJ Multiple Function Control Register 0 (SYS_GPJ_MFP0)

Register	Offset	R/W	Description	Reset Value
SYS_GPJ_MFP0	SYS_BA+0x590	R/W	GPIOJ Multiple Function Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PJ3MFP				
23	22	21	20	19	18	17	16
Reserved			PJ2MFP				
15	14	13	12	11	10	9	8
Reserved			PJ1MFP				
7	6	5	4	3	2	1	0
Reserved			PJ0MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PJ3MFP**	**PJ.3 Multi-function Pin Selection** 02 = EBI_AD4. 03 = UART8_nRTS. 06 = QSPI1_CLK. 07 = CCAP_DATA4. 10 = CAN0_RXD. 16 = HBI_D3.
[23:21]	**Reserved**	Reserved.
[20:16]	**PJ2MFP**	**PJ.2 Multi-function Pin Selection** 02 = EBI_AD5. 03 = UART8_nCTS. 06 = QSPI1_SS. 07 = CCAP_DATA5. 10 = CAN0_TXD. 16 = HBI_nRESET.
[15:13]	**Reserved**	Reserved.
[12:8]	**PJ1MFP**	**PJ.1 Multi-function Pin Selection** 03 = SPIM_MOSI. 04 = QSPI0_MOSI0. 07 = UART8_RXD. 10 = CAN2_RXD. 12 = EPWM1_CH5.
[7:5]	**Reserved**	Reserved.
[4:0]	**PJ0MFP**	**PJ.0 Multi-function Pin Selection** 03 = SPIM_CLK.

Bits	Description	
		04 = QSPI0_MISO0. 07 = UART8_TXD. 10 = CAN2_TXD. 12 = EPWM1_CH4.

GPIOJ Multiple Function Control Register 1 (SYS_GPJ_MFP1)

Register	Offset	R/W	Description	Reset Value
SYS_GPJ_MFP1	SYS_BA+0x594	R/W	GPIOJ Multiple Function Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PJ7MFP				
23	22	21	20	19	18	17	16
Reserved			PJ6MFP				
15	14	13	12	11	10	9	8
Reserved			PJ5MFP				
7	6	5	4	3	2	1	0
Reserved			PJ4MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PJ7MFP**	**PJ.7 Multi-function Pin Selection** 02 = EBI_AD0. 03 = UART9_nRTS. 07 = CCAP_DATA0. 10 = CAN2_RXD. 16 = HBI_nCS.
[23:21]	**Reserved**	Reserved.
[20:16]	**PJ6MFP**	**PJ.6 Multi-function Pin Selection** 02 = EBI_AD1. 03 = UART9_nCTS. 07 = CCAP_DATA1. 10 = CAN2_TXD. 16 = HBI_D0.
[15:13]	**Reserved**	Reserved.
[12:8]	**PJ5MFP**	**PJ.5 Multi-function Pin Selection** 02 = EBI_AD2. 03 = UART8_RXD. 06 = QSPI1_MOSI0. 07 = CCAP_DATA2. 10 = CAN1_RXD. 16 = HBI_D1.
[7:5]	**Reserved**	Reserved.
[4:0]	**PJ4MFP**	**PJ.4 Multi-function Pin Selection** 02 = EBI_AD3. 03 = UART8_TXD.

Bits	Description	
		06 = QSPI1_MISO0. 07 = CCAP_DATA3. 10 = CAN1_TXD. 16 = HBI_D2.

GPIOJ Multiple Function Control Register 2 (SYS_GPJ_MFP2)

Register	Offset	R/W	Description	Reset Value
SYS_GPJ_MFP2	SYS_BA+0x598	R/W	GPIOJ Multiple Function Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			PJ11MFP				
23	22	21	20	19	18	17	16
Reserved			PJ10MFP				
15	14	13	12	11	10	9	8
Reserved			PJ9MFP				
7	6	5	4	3	2	1	0
Reserved			PJ8MFP				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PJ11MFP**	**PJ.11 Multi-function Pin Selection** 02 = EBI_ALE. 03 = SD1_DAT0. 04 = SPIM_D3. 06 = UART6_RXD. 08 = I2C4_SDA. 09 = ECAP2_IC1. 11 = CAN0_RXD. 12 = BPWM0_CH2.
[23:21]	**Reserved**	Reserved.
[20:16]	**PJ10MFP**	**PJ.10 Multi-function Pin Selection** 02 = EBI_MCLK. 03 = SD1_DAT1. 04 = SPIM_D2. 06 = UART6_TXD. 08 = I2C4_SCL. 09 = ECAP2_IC0. 11 = CAN0_TXD. 12 = BPWM0_CH3.
[15:13]	**Reserved**	Reserved.
[12:8]	**PJ9MFP**	**PJ.9 Multi-function Pin Selection** 02 = EBI_nWR. 03 = SD1_DAT2. 04 = SPIM_MISO. 06 = UART7_RXD. 11 = CAN2_RXD.

Bits	Description	
		12 = BPWM0_CH4.
[7:5]	**Reserved**	Reserved.
[4:0]	**PJ8MFP**	**PJ.8 Multi-function Pin Selection** 02 = EBI_nRD. 03 = SD1_DAT3. 04 = SPIM_SS. 06 = UART7_TXD. 11 = CAN2_TXD. 12 = BPWM0_CH5.

GPIOJ Multiple Function Control Register 3 (SYS_GPJ_MFP3)

Register	Offset	R/W	Description	Reset Value
SYS_GPJ_MFP3	SYS_BA+0x59C	R/W	GPIOJ Multiple Function Control Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			**PJ15MFP**				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved			**PJ14MFP**				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**PJ13MFP**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**PJ12MFP**				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**PJ15MFP**	PJ.15 Multi-function Pin Selection
[23:21]	**Reserved**	Reserved.
[20:16]	**PJ14MFP**	PJ.14 Multi-function Pin Selection
[15:13]	**Reserved**	Reserved.
[12:8]	**PJ13MFP**	**PJ.13 Multi-function Pin Selection** 03 = SD1_CLK. 04 = SPIM_MOSI. 08 = I2C4_SMBSUS. 11 = CAN1_RXD. 12 = BPWM0_CH0. 15 = HSUSB_VBUS_EN.
[7:5]	**Reserved**	Reserved.
[4:0]	**PJ12MFP**	**PJ.12 Multi-function Pin Selection** 02 = EBI_nCS0. 03 = SD1_CMD. 04 = SPIM_CLK. 08 = I2C4_SMBAL. 09 = ECAP2_IC2. 11 = CAN1_TXD. 12 = BPWM0_CH1. 15 = HSUSB_VBUS_ST.

6.2.13 System Timer (SysTick)

The Cortex-M4 includes an integrated system timer, SysTick, which provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used as a Real Time Operating System (RTOS) tick timer or as a simple counter.

When system timer is enabled, it will count down from the value in the SysTick Current Value Register (SYST_VAL) to zero, and reload (wrap) to the value in the SysTick Reload Value Register (SYST_LOAD) on the next clock cycle, and then decrement on subsequent clocks. When the counter transitions to zero, the COUNTFLAG status bit is set. The COUNTFLAG bit clears on reads.

The SYST_VAL value is UNKNOWN on reset. Software should write to the register to clear it to zero before enabling the feature. This ensures the timer will count from the SYST_LOAD value rather than an arbitrary value when it is enabled.

If the SYST_LOAD is zero, the timer will be maintained with a current value of zero after it is reloaded with this value. This mechanism can be used to disable the feature independently from the timer enable bit.

For more detailed information, please refer to the "*Arm Cortex-M4 Technical Reference Manual*" and "*Arm v7-M Architecture Reference Manual*".

6.2.14 System Timer Control Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SYST Base Address: **SCS_BA = 0xE000_E000**				
SYST_CTRL	SCS_BA+0x10	R/W	SysTick Control and Status Register	0x0000_0000
SYST_LOAD	SCS_BA+0x14	R/W	SysTick Reload Value Register	0xXXXX_XXXX
SYST_VAL	SCS_BA+0x18	R/W	SysTick Current Value Register	0xXXXX_XXXX

6.2.14.1 System Timer Control Register Description

SysTick Control and Status Register (SYST_CTRL)

Register	Offset	R/W	Description	Reset Value
SYST_CTRL	SCS_BA+0x10	R/W	SysTick Control and Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							COUNTFLAG
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CLKSRC	TICKINT	ENABLE

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**COUNTFLAG**	**System Tick Counter Flag** Returns 1 if timer counted to 0 since last time this register was read. COUNTFLAG is set by a count transition from 1 to 0. COUNTFLAG is cleared on read or by a write to the Current Value register.
[15:3]	**Reserved**	Reserved.
[2]	**CLKSRC**	**System Tick Clock Source Selection** 0 = Clock source is the (optional) external reference clock. 1 = Core clock used for SysTick.
[1]	**TICKINT**	**System Tick Interrupt Enabled** 0 = Counting down to 0 does not cause the SysTick exception to be pended. Software can use COUNTFLAG to determine if a count to zero has occurred. 1 = Counting down to 0 will cause the SysTick exception to be pended. Clearing the SysTick current value register by a register write in software will not cause SysTick to be pended.
[0]	**ENABLE**	**System Tick Counter Enabled** 0 = Counter Disabled. 1 = Counter will operate in a multi-shot manner.

SysTick Reload Value Register (SYST_LOAD)

Register	Offset	R/W	Description	Reset Value
SYST_LOAD	SCS_BA+0x14	R/W	SysTick Reload Value Register	0xXXXX_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RELOAD							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RELOAD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RELOAD							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**RELOAD**	**System Tick Reload Value** Value to load into the Current Value register when the counter reaches 0.

SysTick Current Value Register (SYST_VAL)

Register	Offset	R/W	Description	Reset Value
SYST_VAL	SCS_BA+0x18	R/W	SysTick Current Value Register	0xXXXX_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CURRENT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CURRENT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURRENT							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CURRENT**	**System Tick Current Value** Current counter value. This is the value of the counter at the time it is sampled. The counter does not provide read-modify-Write Protect. The register is write-clear. A software write of any value will clear the register to 0.

6.2.15 Nested Vectored Interrupt Controller (NVIC)

The NVIC and the processor core interface are closely coupled to enable low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked, or nested, interrupts to enable tail-chaining of interrupts. You can only fully access the NVIC from privileged mode, but you can cause interrupts to enter a pending state in user mode if you enable the Configuration and Control Register. Any other user mode access causes a bus fault. You can access all NVIC registers using byte, halfword, and word accesses unless otherwise stated. NVIC registers are located within the SCS (System Control Space). All NVIC registers and system debug registers are little-endian regardless of the endianness state of the processor.

The NVIC supports:

- An implementation-defined number of interrupts, in the range 1-240 interrupts.
- A programmable priority level of 0-15 for each interrupt; a higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non Maskable Interrupt (NMI)
- WIC with Ultra-low Power Sleep mode support

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling.

6.2.15.1 Exception Model and System Interrupt Map

Table 6.2-8 lists the exception model supported by M463/M467 Series. Software can set 16 levels of priority on some of these exceptions as well as on all interrupts. The highest user-configurable priority is denoted as “0x00” and the lowest priority is denoted as “0xF0” (The 4-LSB always 0). The default priority of all the user-configurable interrupts is “0x00”. Note that priority “0” is treated as the fourth priority on the system, after three system exceptions “Reset”, “NMI” and “Hard Fault”.

When any interrupts is accepted, the processor will automatically fetch the starting address of the interrupt service routine (ISR) from a vector table in memory. On system reset, the vector table is fixed at address 0x00000000. Privileged software can write to the VTOR to relocate the vector table start address to a different memory location, in the range 0x00000080 to 0x3FFFFF80,

The vector table contains the initialization value for the stack pointer on reset, and the entry point addresses for all exception handlers. The vector number on previous page defines the order of entries in the vector table associated with exception handler entry as illustrated in previous section.

Exception Type	Vector Number	Vector Address	Priority
Reset	1	0x00000004	-3
NMI	2	0x00000008	-2
Hard Fault	3	0x0000000C	-1
Memory Manager Fault	4	0x00000010	Configurable
Bus Fault	5	0x00000014	Configurable
Usage Fault	6	0x00000018	Configurable
Reserved	7 ~ 10		Reserved

SVCall	11	0x0000002C	Configurable
Debug Monitor	12	0x00000030	Configurable
Reserved	13		Reserved
PendSV	14	0x00000038	Configurable
SysTick	15	0x0000003C	Configurable
Interrupt (IRQ0 ~ IRQ127)	16 ~ 143	0x00000000 + (Vector Number)*4	Configurable

Table 6.2-8 Exception Model

Vector Number	Interrupt Number (Bit In Interrupt Registers)	Interrupt Name	Interrupt Description
0 ~ 15	-	-	System exceptions
16	0	BODOUT	Brown-out low voltage detected interrupt
17	1	IRC_INT	IRC TRIM interrupt
18	2	PWRWU_INT	Clock controller interrupt for chip wake-up from power-down state
19	3	SRAM_PERR	SRAM parity check error interrupt
20	4	CLKFAIL	Clock fail detected interrupt
21	5	FMC_ISP_INT	FMC ISP interrupt
22	6	RTC_INT	Real time clock interrupt
23	7	TAMPER_INT	Backup register tamper interrupt
24	8	WDT_INT	Watchdog Timer interrupt
25	9	WWDT_INT	Window Watchdog Timer interrupt
26	10	EINT0	External interrupt from INT0 pin
27	11	EINT1	External interrupt from INT1 pin
28	12	EINT2	External interrupt from INT2 pin
29	13	EINT3	External interrupt from INT3 pin
30	14	EINT4	External interrupt from INT4 pin
31	15	EINT5	External interrupt from INT5 pin
32	16	GPA_INT	External interrupt from PA[15:0] pin
33	17	GPB_INT	External interrupt from PB[15:0] pin
34	18	GPC_INT	External interrupt from PC[15:0] pin
35	19	GPD_INT	External interrupt from PD[15:0] pin
36	20	GPE_INT	External interrupt from PE[15:0] pin
37	21	GPF_INT	External interrupt from PF[15:0] pin
38	22	QSPI0_INT	QSPI0 interrupt

39	23	SPI0_INT	SPI0 interrupt
40	24	BRAKE0_INT	EPWM0 brake interrupt
41	25	EPWM0_P0_INT	EPWM0 pair 0 interrupt
42	26	EPWM0_P1_INT	EPWM0 pair 1 interrupt
43	27	EPWM0_P2_INT	EPWM0 pair 2 interrupt
44	28	BRAKE1_INT	EPWM1 brake interrupt
45	29	EPWM1_P0_INT	EPWM1 pair 0 interrupt
46	30	EPWM1_P1_INT	EPWM1 pair 1 interrupt
47	31	EPWM1_P2_INT	EPWM1 pair 2 interrupt
48	32	TMR0_INT	Timer 0 interrupt
49	33	TMR1_INT	Timer 1 interrupt
50	34	TMR2_INT	Timer 2 interrupt
51	35	TMR3_INT	Timer 3 interrupt
52	36	UART0_INT	UART0 interrupt
53	37	UART1_INT	UART1 interrupt
54	38	I2C0_INT	I2C0 interrupt
55	39	I2C1_INT	I2C1 interrupt
56	40	PDMA0_INT	PDMA0 interrupt
57	41	DAC_INT	DAC interrupt
58	42	EADC0_INT0	EADC0 interrupt source 0
59	43	EADC0_INT1	EADC0 interrupt source 1
60	44	ACMP01_INT	ACMP0 and ACMP1 interrupt
61	45	ACMP23_INT	ACMP2 and ACMP3 interrupt
62	46	EADC0_INT2	EADC0 interrupt source 2
63	47	EADC0_INT3	EADC0 interrupt source 3
64	48	UART2_INT	UART2 interrupt
65	49	UART3_INT	UART3 interrupt
66	50	QSPI1_INT	QSPI1 interrupt
67	51	SPI1_INT	SPI1 interrupt
68	52	SPI2_INT	SPI2 interrupt
69	53	USBD_INT	USB device interrupt
70	54	USBH_INT	USB host interrupt
71	55	USBOTG_INT	USB OTG interrupt
73	57	SPI5_INT	SPI5 interrupt
74	58	SC0_INT	Smart card host 0 interrupt

75	59	SC1_INT	Smart card host 1 interrupt
76	60	SC2_INT	Smart card host 2 interrupt
77	61	GPJ_INT	External interrupt from PJ[15:0] pin
78	62	SPI3_INT	SPI3 interrupt
79	63	SPI4_INT	SPI4 interrupt
80	64	SDHOST0_INT	SD host 0 interrupt
81	65	HSUSBD_INT	HSUSBD interrupt
82	66	EMAC0_INT	Ethernet MAC 0 interrupt
83	67	Reserved	Reserved
84	68	I2S0_INT	I2S0 interrupt
85	69	I2S1_INT	I2S1 interrupt
86	70	SPI6_INT	SPI6 interrupt
87	71	CRYPTO	Crypto interrupt
88	72	GPG_INT	External interrupt from PG[15:0] pin
89	73	EINT6	External interrupt from INT6 pin
90	74	UART4_INT	UART4 interrupt
91	75	UART5_INT	UART5 interrupt
92	76	USCI0_INT	USCI0 interrupt
93	77	SPI7_INT	SPI7 interrupt
94	78	BPWM0_INT	BPWM0 interrupt
95	79	BPWM1_INT	BPWM1 interrupt
96	80	SPIM_INT	SPI Master interrupt
97	81	CCAP_INT	CCAP interrupt
98	82	I2C2_INT	I2C2 interrupt
99	83	I2C3_INT	I2C3 interrupt
100	84	EQEI0_INT	EQEI0 interrupt
101	85	EQEI1_INT	EQEI1 interrupt
102	86	ECAP0_INT	ECAP0 interrupt
103	87	ECAP1_INT	ECAP1 interrupt
105	88	GPH_INT	External interrupt from PH[15:0] pin
105	89	EINT7	External interrupt from INT7 pin
106	90	SDHOST1_INT	SD host 1 interrupt
107	91	PSIO_INT	PSIO interrupt
108	92	HSUSBH_INT	HSUSBH interrupt
109	93	HSOTG_INT	HSOTG interrupt

110	94	ECAP2_INT	ECAP2 interrupt
111	95	ECAP3_INT	ECAP3 interrupt
112	96	KPI_INT	KPI interrupt
113	97	HBI_INT	HyperBus interface interrupt
114	98	PDMA1_INT	PDMA1 interrupt
115	99	UART8_INT	UART8 interrupt
116	100	UART9_INT	UART9 interrupt
117	101	TRNG_INT	TRNG interrupt
118	102	UART6_INT	UART6 interrupt
119	103	UART7_INT	UART7 interrupt
120	104	EADC1_INT0	EADC1 interrupt source 0
121	105	EADC1_INT1	EADC1 interrupt source 1
122	106	EADC1_INT2	EADC1 interrupt source 2
123	107	EADC1_INT3	EADC1 interrupt source 3
124	108	SPI8_INT	SPI8 interrupt
125	109	KS_INT	Key store interrupt
126	110	GPI_INT	External interrupt from PI[15:0] pin
127	111	SPI9_INT	SPI9 interrupt
128	112	CANFD0_INT0	CANFD0 interrupt source 0
129	113	CANFD0_INT1	CANFD0 interrupt source 1
130	114	CANFD1_INT0	CANFD1 interrupt source 0
131	115	CANFD1_INT1	CANFD1 interrupt source 1
132	116	EQEI2_INT	EQEI2 interrupt
133	117	EQEI3_INT	EQEI3 interrupt
134	118	I2C4_INT	I2C4 interrupt
135	119	SPI10_INT	SPI10 interrupt
136	120	CANFD2_INT0	CANFD2 interrupt source 0
137	121	CANFD2_INT1	CANFD2 interrupt source 1
138	122	CANFD3_INT0	CANFD3 interrupt source 0
139	123	CANFD3_INT1	CANFD3 interrupt source 1
140	124	EADC2_INT0	EADC2 interrupt source 0
141	125	EADC2_INT1	EADC2 interrupt source 1
142	126	EADC2_INT2	EADC2 interrupt source 2
143	127	EADC2_INT3	EADC2 interrupt source 3

Table 6.2-9 Interrupt Number Table

6.2.15.2 Operation Description

NVIC interrupts can be enabled and disabled by writing to their corresponding Interrupt Set-Enable or Interrupt Clear-Enable register bit-field. The registers use a write-1-to-enable and write-1-to-clear policy, both registers reading back the current enabled state of the corresponding interrupts. When an interrupt is disabled, interrupt assertion will cause the interrupt to become Pending, however, the interrupt will not activate. If an interrupt is Active when it is disabled, it remains in its Active state until cleared by reset or an exception return. Clearing the enable bit prevents new activations of the associated interrupt.

NVIC interrupts can be pended/un-pended using a complementary pair of registers to those used to enable/disable the interrupts, named the Set-Pending Register and Clear-Pending Register respectively. The registers use a write-1-to-enable and write-1-to-clear policy, both registers reading back the current pended state of the corresponding interrupts. The Clear-Pending Register has no effect on the execution status of an Active interrupt.

NVIC interrupts are prioritized by updating an 8-bit field within a 32-bit register (each register supporting four interrupts).

The general registers associated with the NVIC are all accessible from a block of memory in the System Control Space and will be described in next section.

6.2.15.3 NVIC Control Registers

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
NVIC Base Address: **NVIC_BA = 0xE000_E100**				
NVIC_ISER0	NVIC_BA+0x00	R/W	IRQ0 ~ IRQ127 Set-enable Control Register	0x0000_0000
NVIC_ISER1	NVIC_BA+0x04	R/W	IRQ0 ~ IRQ127 Set-enable Control Register	0x0000_0000
NVIC_ISER2	NVIC_BA+0x08	R/W	IRQ0 ~ IRQ127 Set-enable Control Register	0x0000_0000
NVIC_ISER3	NVIC_BA+0x0C	R/W	IRQ0 ~ IRQ127 Set-enable Control Register	0x0000_0000
NVIC_ICER0	NVIC_BA+0x80	R/W	IRQ0 ~ IRQ127 Clear-enable Control Register	0x0000_0000
NVIC_ICER1	NVIC_BA+0x84	R/W	IRQ0 ~ IRQ127 Clear-enable Control Register	0x0000_0000
NVIC_ICER2	NVIC_BA+0x88	R/W	IRQ0 ~ IRQ127 Clear-enable Control Register	0x0000_0000
NVIC_ICER3	NVIC_BA+0x8C	R/W	IRQ0 ~ IRQ127 Clear-enable Control Register	0x0000_0000
NVIC_ISPR0	NVIC_BA+0x100	R/W	IRQ0 ~ IRQ127 Set-pending Control Register	0x0000_0000
NVIC_ISPR1	NVIC_BA+0x104	R/W	IRQ0 ~ IRQ127 Set-pending Control Register	0x0000_0000
NVIC_ISPR2	NVIC_BA+0x108	R/W	IRQ0 ~ IRQ127 Set-pending Control Register	0x0000_0000
NVIC_ISPR3	NVIC_BA+0x10C	R/W	IRQ0 ~ IRQ127 Set-pending Control Register	0x0000_0000
NVIC_ICPR0	NVIC_BA+0x180	R/W	IRQ0 ~ IRQ127 Clear-pending Control Register	0x0000_0000
NVIC_ICPR1	NVIC_BA+0x184	R/W	IRQ0 ~ IRQ127 Clear-pending Control Register	0x0000_0000
NVIC_ICPR2	NVIC_BA+0x188	R/W	IRQ0 ~ IRQ127 Clear-pending Control Register	0x0000_0000
NVIC_ICPR3	NVIC_BA+0x18C	R/W	IRQ0 ~ IRQ127 Clear-pending Control Register	0x0000_0000
NVIC_IABR0	NVIC_BA+0x200	R/W	IRQ0 ~ IRQ127 Active Bit Register	0x0000_0000
NVIC_IABR1	NVIC_BA+0x204	R/W	IRQ0 ~ IRQ127 Active Bit Register	0x0000_0000
NVIC_IABR2	NVIC_BA+0x208	R/W	IRQ0 ~ IRQ127 Active Bit Register	0x0000_0000
NVIC_IABR3	NVIC_BA+0x20C	R/W	IRQ0 ~ IRQ127 Active Bit Register	0x0000_0000
NVIC_IPRn **n=0,1..31**	NVIC_BA+0x300 +0x4*n	R/W	IRQ0 ~ IRQ127 Priority Control Register	0x0000_0000
STIR	NVIC_BA+0xE00	W	Software Trigger Interrupt Registers	0x0000_0000

IRQ0 ~ IRQ127 Set-enable Control Register (NVIC_ISER0)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER0	NVIC_BA+0x00	R/W	IRQ0 ~ IRQ127 Set-enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETENA							

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ127 Set-enable Control Register (NVIC_ISER1)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER1	NVIC_BA+0x04	R/W	IRQ0 ~ IRQ127 Set-enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETENA							

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ127 Set-enable Control Register (NVIC_ISER2)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER2	NVIC_BA+0x08	R/W	IRQ0 ~ IRQ127 Set-enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETENA							

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ127 Set-enable Control Register (NVIC_ISER3)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER3	NVIC_BA+0x0C	R/W	IRQ0 ~ IRQ127 Set-enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETENA							

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ127 Clear-enable Control Register (NVIC_ICER0)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER0	NVIC_BA+0x80	R/W	IRQ0 ~ IRQ127 Clear-enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALENA							

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ127 Clear-enable Control Register (NVIC_ICER1)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER1	NVIC_BA+0x84	R/W	IRQ0 ~ IRQ127 Clear-enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALENA							

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ127 Clear-enable Control Register (NVIC_ICER2)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER2	NVIC_BA+0x88	R/W	IRQ0 ~ IRQ127 Clear-enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALENA							

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ127 Clear-enable Control Register (NVIC_ICER3)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER3	NVIC_BA+0x8C	R/W	IRQ0 ~ IRQ127 Clear-enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALENA							

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ127 Set-pending Control Register (NVIC_ISPR0)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR0	NVIC_BA+0x100	R/W	IRQ0 ~ IRQ127 Set-pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETPEND							

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ127 Set-pending Control Register (NVIC_ISPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR1	NVIC_BA+0x104	R/W	IRQ0 ~ IRQ127 Set-pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETPEND							

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ127 Set-pending Control Register (NVIC_ISPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR2	NVIC_BA+0x108	R/W	IRQ0 ~ IRQ127 Set-pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETPEND							

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ127 Set-pending Control Register (NVIC_ISPR3)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR3	NVIC_BA+0x10C	R/W	IRQ0 ~ IRQ127 Set-pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETPEND							

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ127 Clear-pending Control Register (NVIC_ICPR0)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR0	NVIC_BA+0x180	R/W	IRQ0 ~ IRQ127 Clear-pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALPEND							

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Removes pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ127 Clear-pending Control Register (NVIC_ICPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR1	NVIC_BA+0x184	R/W	IRQ0 ~ IRQ127 Clear-pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALPEND							

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Removes pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ127 Clear-pending Control Register (NVIC_ICPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR2	NVIC_BA+0x188	R/W	IRQ0 ~ IRQ127 Clear-pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALPEND							

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Removes pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ127 Clear-pending Control Register (NVIC_ICPR3)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR3	NVIC_BA+0x18C	R/W	IRQ0 ~ IRQ127 Clear-pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALPEND							

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR3 registers remove the pending state from interrupts, and show which interrupts are pending. Write Operation: 0 = No effect. 1 = Removes pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ127 Active Bit Register (NVIC_IABR0)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR0	NVIC_BA+0x200	R/W	IRQ0 ~ IRQ127 Active Bit Register	0x0000_0000

31	30	29	28	27	26	25	24
ACTIVE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ACTIVE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ACTIVE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACTIVE							

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = Interrupt not active. 1 = Interrupt active.

IRQ0 ~ IRQ127 Active Bit Register (NVIC_IABR1)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR1	NVIC_BA+0x204	R/W	IRQ0 ~ IRQ127 Active Bit Register	0x0000_0000

31	30	29	28	27	26	25	24
ACTIVE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ACTIVE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ACTIVE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACTIVE							

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = Interrupt not active. 1 = Interrupt active.

IRQ0 ~ IRQ127 Active Bit Register (NVIC_IABR2)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR2	NVIC_BA+0x208	R/W	IRQ0 ~ IRQ127 Active Bit Register	0x0000_0000

31	30	29	28	27	26	25	24
ACTIVE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ACTIVE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ACTIVE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACTIVE							

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = Interrupt not active. 1 = Interrupt active.

IRQ0 ~ IRQ127 Active Bit Register (NVIC_IABR3)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR3	NVIC_BA+0x20C	R/W	IRQ0 ~ IRQ127 Active Bit Register	0x0000_0000

31	30	29	28	27	26	25	24
ACTIVE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ACTIVE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ACTIVE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACTIVE							

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = Interrupt not active. 1 = Interrupt active.

IRQ0 ~ IRQ127 Interrupt Priority Register (NVIC_IPRn)

Register	Offset	R/W	Description	Reset Value
NVIC_IPRn **n=0,1..31**	NVIC_BA+0x300 +0x4*n	R/W	IRQ0 ~ IRQ127 Priority Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PRI_4n_3				Reserved			
23	22	21	20	19	18	17	16
PRI_4n_2				Reserved			
15	14	13	12	11	10	9	8
PRI_4n_1				Reserved			
7	6	5	4	3	2	1	0
PRI_4n_0				Reserved			

Bits	Description	
[31:28]	**PRI_4n_3**	**Priority of IRQ_4n+3** “0” denotes the highest priority and “15” denotes the lowest priority
[27:24]	**Reserved**	Reserved.
[23:20]	**PRI_4n_2**	**Priority of IRQ_4n+2** “0” denotes the highest priority and “15” denotes the lowest priority
[19:16]	**Reserved**	Reserved.
[15:12]	**PRI_4n_1**	**Priority of IRQ_4n+1** “0” denotes the highest priority and “15” denotes the lowest priority
[11:8]	**Reserved**	Reserved.
[7:4]	**PRI_4n_0**	**Priority of IRQ_4n+0** “0” denotes the highest priority and “15” denotes the lowest priority
[3:0]	**Reserved**	Reserved.

Software Trigger Interrupt Register (STIR)

Register	Offset	R/W	Description	Reset Value
STIR	NVIC_BA+0xE00	W	Software Trigger Interrupt Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**INTID**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTID							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**INTID**	**Interrupt ID (Write Only)** Write to the STIR To Generate An Interrupt from Software When the USERSETMPEND bit in the CCR is set to 1, unprivileged software can access the STIR. Number of valid Interrupt ID refers to Table 6.2-9. For example, a value of 0x03 specifies interrupt IRQ3.

6.2.15.4 NMI Control Registers

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
NMI Base Address: **NMI_BA = 0x4000_0300**				
NMIEN	NMI_BA+0x00	R/W	NMI Source Interrupt Enable Register	0x0000_0000
NMISTS	NMI_BA+0x04	R	NMI Source Interrupt Status Register	0x0000_0000

NMI Source Interrupt Enable Register (NMIEN)

Register	Offset	R/W	Description	Reset Value
NMIEN	NMI_BA+0x00	R/W	NMI Source Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
UART1INT	**UART0INT**	**EINT5**	**EINT4**	**EINT3**	**EINT2**	**EINT1**	**EINT0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TAMPERINT	**RTCINT**	**Reserved**	**CLKFAIL**	**SRAMPERR**	**PWRWUINT**	**IRCINT**	**BODOUT**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**UART1INT**	**UART1 NMI Source Enable (Write Protect)** 0 = UART1 NMI source Disabled. 1 = UART1 NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[14]	**UART0INT**	**UART0 NMI Source Enable (Write Protect)** 0 = UART0 NMI source Disabled. 1 = UART0 NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[13]	**EINT5**	**External Interrupt from INT5 Pins NMI Source Enable (Write Protect)** 0 = External interrupt from INT5 pins NMI source Disabled. 1 = External interrupt from INT5 pins NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[12]	**EINT4**	**External Interrupt from INT4 Pins NMI Source Enable (Write Protect)** 0 = External interrupt from INT4 pins NMI source Disabled. 1 = External interrupt from INT4 pins NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11]	**EINT3**	**External Interrupt from INT3 Pins NMI Source Enable (Write Protect)** 0 = External interrupt from INT3 pins NMI source Disabled. 1 = External interrupt from INT3 pins NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[10]	**EINT2**	**External Interrupt from INT2 Pins NMI Source Enable (Write Protect)** 0 = External interrupt from INT2 pins NMI source Disabled. 1 = External interrupt from INT2 pins NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[9]	**EINT1**	**External Interrupt from INT1 Pins NMI Source Enable (Write Protect)**

		0 = External interrupt from INT1 pins NMI source Disabled. 1 = External interrupt from INT1 pins NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[8]	**EINT0**	**External Interrupt from INT0 Pins NMI Source Enable (Write Protect)** 0 = External interrupt from INT0 pins NMI source Disabled. 1 = External interrupt from INT0 pins NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	**TAMPERINT**	**TAMPER NMI Source Enable (Write Protect)** 0 = Backup register tamper detected interrupt.NMI source Disabled. 1 = Backup register tamper detected interrupt.NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**RTCINT**	**RTC NMI Source Enable (Write Protect)** 0 = RTC NMI source Disabled. 1 = RTC NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**Reserved**	Reserved.
[4]	**CLKFAIL**	**Clock Fail Detected NMI Source Enable (Write Protect)** 0 = Clock fail detected interrupt NMI source Disabled. 1 = Clock fail detected interrupt NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**SRAMPERR**	**SRAM Parity Check Error NMI Source Enable (Write Protect)** 0 = SRAM parity check error NMI source Disabled. 1 = SRAM parity check error NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**PWRWUINT**	**Power-down Mode Wake-up NMI Source Enable (Write Protect)** 0 = Power-down mode wake-up NMI source Disabled. 1 = Power-down mode wake-up NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[1]	**IRCINT**	**IRC TRIM NMI Source Enable (Write Protect)** 0 = IRC TRIM NMI source Disabled. 1 = IRC TRIM NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**BODOUT**	**BOD NMI Source Enable (Write Protect)** 0 = BOD NMI source Disabled. 1 = BOD NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

NMI Source Interrupt Status Register (NMISTS)

Register	Offset	R/W	Description	Reset Value
NMISTS	NMI_BA+0x04	R	NMI Source Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
UART1INT	UART0INT	EINT5	EINT4	EINT3	EINT2	EINT1	EINT0
7	6	5	4	3	2	1	0
TAMPERINT	RTCINT	Reserved	CLKFAIL	SRAMPERR	PWRWUINT	IRCINT	BODOUT

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**UART1_INT**	**UART1 Interrupt Flag (Read Only)** 0 = UART1 interrupt is deasserted. 1 = UART1 interrupt is asserted.
[14]	**UART0_INT**	**UART0 Interrupt Flag (Read Only)** 0 = UART1 interrupt is deasserted. 1 = UART1 interrupt is asserted.
[13]	**EINT5**	**External Interrupt from INT5 Pins Interrupt Flag (Read Only)** 0 = External Interrupt from INT5 interrupt is deasserted. 1 = External Interrupt from INT5 interrupt is asserted.
[12]	**EINT4**	**External Interrupt from INT4 Pins Interrupt Flag (Read Only)** 0 = External Interrupt from INT4 interrupt is deasserted. 1 = External Interrupt from INT4 interrupt is asserted.
[11]	**EINT3**	**External Interrupt from INT3 Pins Interrupt Flag (Read Only)** 0 = External Interrupt from PD.0 interrupt is deasserted. 1 = External Interrupt from PD.0 interrupt is asserted.
[10]	**EINT2**	**External Interrupt from INT2 Pins Interrupt Flag (Read Only)** 0 = External Interrupt from INT2 interrupt is deasserted. 1 = External Interrupt from INT2 interrupt is asserted.
[9]	**EINT1**	**External Interrupt from INT1 Pins Interrupt Flag (Read Only)** 0 = External Interrupt from INT1 interrupt is deasserted. 1 = External Interrupt from INT1 interrupt is asserted.
[8]	**EINT0**	**External Interrupt from INT0 Pins Interrupt Flag (Read Only)** 0 = External Interrupt from INT0 interrupt is deasserted. 1 = External Interrupt from INT0 interrupt is asserted.
[7]	**TAMPERINT**	**TAMPER Interrupt Flag (Read Only)**

		0 = Backup register tamper detected interrupt is deasserted. 1 = Backup register tamper detected interrupt is asserted.
[6]	**RTCINT**	**RTC Interrupt Flag (Read Only)** 0 = RTC interrupt is deasserted. 1 = RTC interrupt is asserted.
[5]	**Reserved**	Reserved.
[4]	**CLKFAIL**	**Clock Fail Detected Interrupt Flag (Read Only)** 0 = Clock fail detected interrupt is deasserted. 1 = Clock fail detected interrupt is asserted.
[3]	**SRAMPERR**	**SRAM Parity Check Error Interrupt Flag (Read Only)** 0 = SRAM parity check error interrupt is deasserted. 1 = SRAM parity check error interrupt is asserted.
[2]	**PWRWUINT**	**Power-down Mode Wake-up Interrupt Flag (Read Only)** 0 = Power-down mode wake-up interrupt is deasserted. 1 = Power-down mode wake-up interrupt is asserted.
[1]	**IRCINT**	**IRC TRIM Interrupt Flag (Read Only)** 0 = IRC TRIM interrupt is deasserted. 1 = IRC TRIM interrupt is asserted.
[0]	**BODOUT**	**BOD Interrupt Flag (Read Only)** 0 = BOD interrupt is deasserted. 1 = BOD interrupt is asserted.

6.2.16 System Control Register

The Cortex-M4 status and operation mode control are managed by System Control Registers. Including CPUID, Cortex-M4 interrupt priority and Cortex-M4 power management can be controlled through these system control registers.

For more detailed information, please refer to the "*Arm Cortex-M4 Technical Reference Manual*" and "*Arm v7-M Architecture Reference Manual*".

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
SCR Base Address: **SCS_BA = 0xE000_E000**				
ICSR	SCS_BA+0xD04	R/W	Interrupt Control and State Register	0x0000_0000
AIRCR	SCS_BA+0xD0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000
SCR	SCS_BA+0xD10	R/W	System Control Register	0x0000_0000
SHPR1	SCS_BA+0xD18	R/W	System Handler Priority Register 1	0x0000_0000
SHPR2	SCS_BA+0xD1C	R/W	System Handler Priority Register 2	0x0000_0000
SHPR3	SCS_BA+0xD20	R/W	System Handler Priority Register 3	0x0000_0000

Interrupt Control State Register (ICSR)

Register	Offset	R/W	Description	Reset Value
ICSR	SCS_BA+0xD04	R/W	Interrupt Control and State Register	0x0000_0000

31	30	29	28	27	26	25	24
NMIPENDSET	Reserved		PENDSVSET	PENDSVRTC_CAL	PENDSTSET	PENDSTRTC_CAL	Reserved
23	22	21	20	19	18	17	16
ISRPREEMPT	ISRPENDING	Reserved				VECTPENDING	
15	14	13	12	11	10	9	8
VECTPENDING				RETTOBASE	Reserved		
7	6	5	4	3	2	1	0
Reserved	VECTACTIVE						

Bits	Description	
[31]	**NMIPENDSET**	**NMI Set-pending Bit** Write Operation: 0 = No effect. 1 = Change NMI exception state to pending. Read Operation: 0 = NMI exception is not pending. 1 = NMI exception is pending. **Note:** Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it detects a write of 1 to this bit. Entering the handler then clears this bit to 0. This means a read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
[30:29]	**Reserved**	Reserved.
[28]	**PENDSVSET**	**PendSV Set-pending Bit** Write Operation: 0 = No effect. 1 = Change PendSV exception state to pending. Read Operation: 0 = PendSV exception is not pending. 1 = PendSV exception is pending. **Note:** Writing 1 to this bit is the only way to set the PendSV exception state to pending.
[27]	**PENDSVRTC_CAL**	**PendSV Clear-pending Bit (Write Only)** Write Operation: 0 = No effect. 1 = Remove the pending state from the PendSV exception. **Note:** This is a write only bit. To clear the PENDSV bit, you must “write 0 to PENDSVSET and write 1 to PENDSVRTC_CAL” at the same time.

[26]	**PENDSTSET**	**SysTick Exception Set-pending Bit** Write Operation: 0 = No effect. 1 = Change SysTick exception state to pending. Read Operation: 0 = SysTick exception is not pending. 1 = SysTick exception is pending.
[25]	**PENDSTRTC_CAL**	**SysTick Exception Clear-pending Bit (Write Only)** Write Operation: 0 = No effect. 1 = Remove the pending state from the SysTick exception. **Note:** This is a write only bit. To clear the PENDST bit, you must "write 0 to PENDSTSET and write 1 to PENDSTRTC_CAL" at the same time.
[24]	**Reserved**	Reserved.
[23]	**ISRPREEMPT**	**Interrupt Preempt Bit (Read Only)** If set, a pending exception will be serviced on exit from the debug halt state.
[22]	**ISRPENDING**	**Interrupt Pending Flag, Excluding NMI and Faults (Read Only)** 0 = Interrupt not pending. 1 = Interrupt pending.
[21:18]	**Reserved**	Reserved.
[17:12]	**VECTPENDING**	**Number of the Highest Pended Exception** Indicate the Exception Number of the Highest Priority Pending Enabled Exception 0 = No pending exceptions. Nonzero = the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the BASEPRI and FAULTMASK registers, but not any effect of the PRIMASK register.
[11]	**RETTOBASE**	**Preempted Active Exceptions Indicator** Indicate whether There are Preempted Active Exceptions 0 = There are preempted active exceptions to execute. 1 = There are no active exceptions, or the currently-executing exception is the only active exception.
[10:7]	**Reserved**	Reserved.
[6:0]	**VECTACTIVE**	**Number of the Current Active Exception** 0 = Thread mode. Non-zero = The exception number of the currently active exception.

Application Interrupt and Reset Control Register (AIRCR)

Register	Offset	R/W	Description	Reset Value
AIRCR	SCS_BA+0xD0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000

31	30	29	28	27	26	25	24
VECTORKEY							
23	22	21	20	19	18	17	16
VECTORKEY							
15	14	13	12	11	10	9	8
ENDIANNESS	Reserved				PRIGROUP		
7	6	5	4	3	2	1	0
Reserved					SYSRESETREQ	VECTCLRACTIVE	VECTRESET

Bits	Description	
[31:16]	**VECTORKEY**	**Register Access Key** When writing this register, this field should be 0x05FA, otherwise the write action will be unpredictable. The VECTORKEY filed is used to prevent accidental write to this register from resetting the system or clearing of the exception status.
[15]	**ENDIANNESS**	**Data Endianness** 0 = Little-endian. 1 = Big-endian.
[14:11]	**Reserved**	Reserved.
[10:8]	**PRIGROUP**	**Interrupt Priority Grouping** This field determines the Split Of Group priority from subpriority,
[7:3]	**Reserved**	Reserved.
[2]	**SYSRESETREQ**	**System Reset Request (Write Only)** Writing This Bit to 1 Will Cause A Reset Signal To Be Asserted To The Chip And Indicate A Reset Is Requested This bit is write only and self-cleared as part of the reset sequence.
[1]	**VECTCLRACTIVE**	**Exception Active Status Clear Bit (Write Only)** Setting This Bit To 1 Will Clears All Active State Information For Fixed And Configurable Exceptions This bit is write only and can only be written when the core is halted. **Note:** It is the debugger's responsibility to re-initialize the stack.
[0]	**VECTRESET**	**Vector Reset Request (Write Only)** Writing 1 to this bit causes a local system reset. This bit self-clears. The effect of writing a 1 to this bit if the processor is not halted in Debug state is unpredictable When the processor is halted in Debug state, if a write to the register writes a 1 to both VECTRESET and SYSRESETREQ, the behavior is unpredictable.

PRIGROUP	Binary Point	Group Priority Bits	Subpriority Bits	Number Of Group Priorities	Subpriorities
0b000	bxxxxxxx.y	[7:1]	[0]	128	2
0b001	bxxxxxx.yy	[7:2]	[1:0]	64	4
0b010	bxxxxx.yyy	[7:3]	[2:0]	32	8
0b011	bxxxx.yyyy	[7:4]	[3;0]	16	16
0b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
0b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
0b110	bx.yyyyyyy	[7]	[6:0]	2	128
0b111	b.yyyyyyyy	None	[7:0]	1	256

Table 6.2-10 Priority Grouping

System Control Register (SCR)

Register	Offset	R/W	Description	Reset Value
SCR	SCS_BA+0xD10	R/W	System Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			SEVONPEND	Reserved	SLEEPDEEP	SLEEPONEXIT	Reserved

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**SEVONPEND**	**Send Event on Pending** 0 = Only enabled interrupts or events can wake up the processor, while disabled interrupts are excluded. 1 = Enabled events and all interrupts, including disabled interrupts, can wake up the processor. When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or an external event.
[3]	**Reserved**	Reserved.
[2]	**SLEEPDEEP**	**Processor Deep Sleep and Sleep Mode Selection** Control whether the Processor Uses Sleep Or Deep Sleep as its Low Power Mode. 0 = Sleep. 1 = Deep sleep.
[1]	**SLEEPONEXIT**	**Sleep-on-exit Enable Control** This bit indicates Sleep-On-Exit when Returning from Handler Mode to Thread Mode. 0 = Do not sleep when returning to Thread mode. 1 = Enter sleep, or deep sleep, on return from an ISR to Thread mode. Set this bit to 1 enable an interrupt driven application to avoid returning to an empty main application.
[0]	**Reserved**	Reserved.

System Handler Priority Register 1 (SHPR1)

Register	Offset	R/W	Description	Reset Value
SHPR1	SCS_BA+0xD18	R/W	System Handler Priority Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_6				**Reserved**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_5				**Reserved**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_4				**Reserved**			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**PRI_6**	Priority of system handler 6, UsageFault
[20:16]	**Reserved**	Reserved.
[15:12]	**PRI_5**	Priority of system handler 5, BusFault
[11:8]	**Reserved**	Reserved.
[7:4]	**PRI_4**	Priority of system handler 4, MemManage
[3:0]	**Reserved**	Reserved.

System Handler Priority Register 2 (SHPR2)

Register	Offset	R/W	Description	Reset Value
SHPR2	SCS_BA+0xD1C	R/W	System Handler Priority Register 2	0x0000_0000

31	30	29	28	27	26	25	24
PRI_11				**Reserved**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:28]	**PRI_11**	**Priority of System Handler 11 – SVCall** “0” denotes the highest priority and “15” denotes the lowest priority.
[27:0]	**Reserved**	Reserved.

System Handler Priority Register 3 (SHPR3)

Register	Offset	R/W	Description	Reset Value
SHPR3	SCS_BA+0xD20	R/W	System Handler Priority Register 3	0x0000_0000

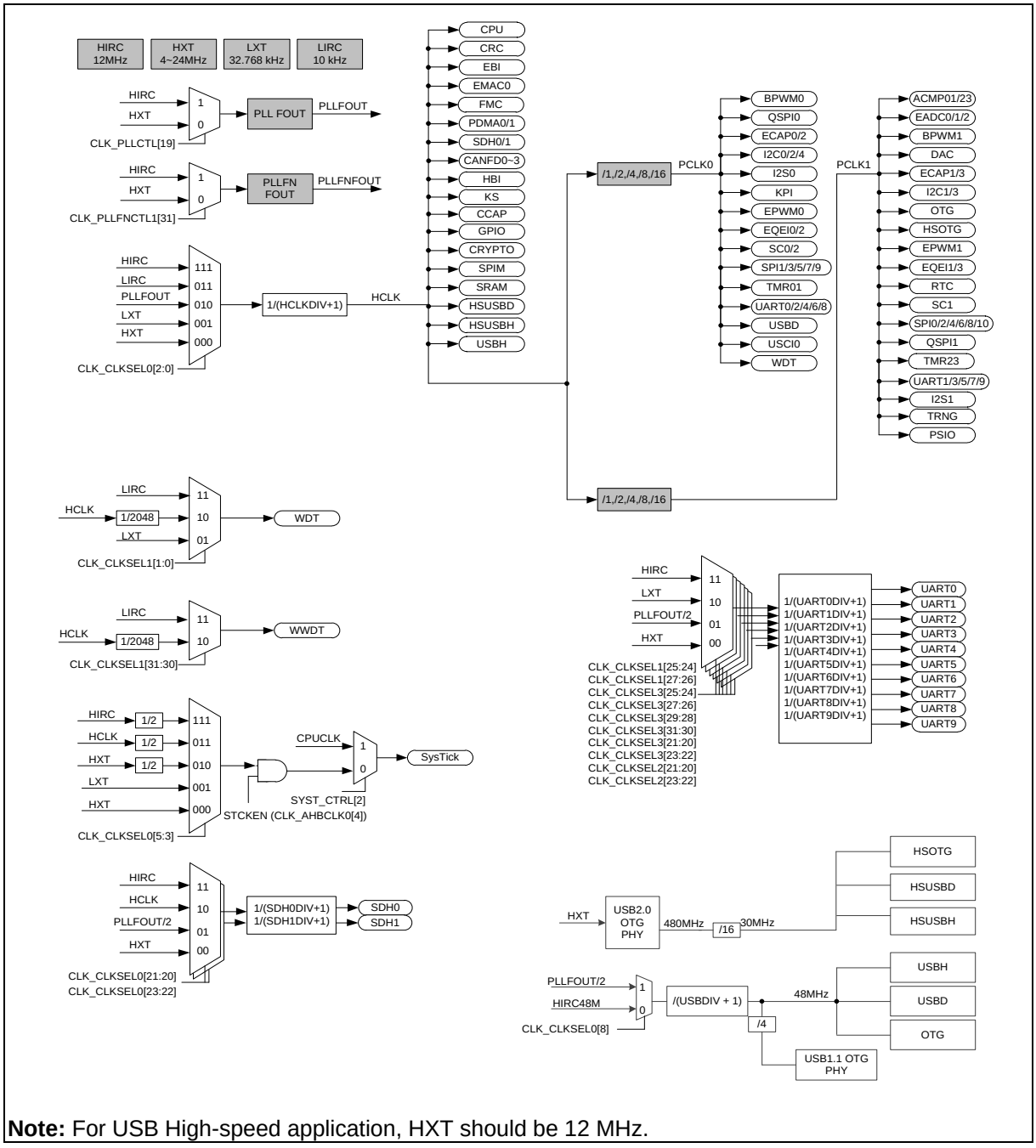
31	30	29	28	27	26	25	24
PRI_15				**Reserved**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_14				**Reserved**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:28]	**PRI_15**	**Priority of System Handler 15 – SysTick** “0” denotes the highest priority and “15” denotes the lowest priority.
[27:24]	**Reserved**	Reserved.
[23:20]	**PRI_14**	**Priority of System Handler 14 – PendSV** “0” denotes the highest priority and “15” denotes the lowest priority.
[19:0]	**Reserved**	Reserved.

6.3 Clock Controller

6.3.1 Overview

The clock controller generates clocks for the whole chip, including system clocks and all peripheral clocks. The clock controller also implements the power control function with the individually clock ON/OFF control, clock source selection and a clock divider. The chip will not enter Power-down mode until CPU sets the Power-down enable bit PDEN(CLK_PWRCTL[7]) and Cortex-M4 core executes the WFI instruction. After that, chip enters Power-down mode and wait for wake-up interrupt source triggered to leave Power-down mode. In Power-down mode, the clock controller turns off the 4~24 MHz external high speed crystal (HXT), 12 MHz internal high speed RC oscillator (HIRC) and PLL to reduce the overall system power consumption. Figure 6.3-1 shows the clock generator and the overview of the clock source control.



Note: For USB High-speed application, HXT should be 12 MHz.

Figure 6.3-1 Clock Generator Global View Diagram (1/3)

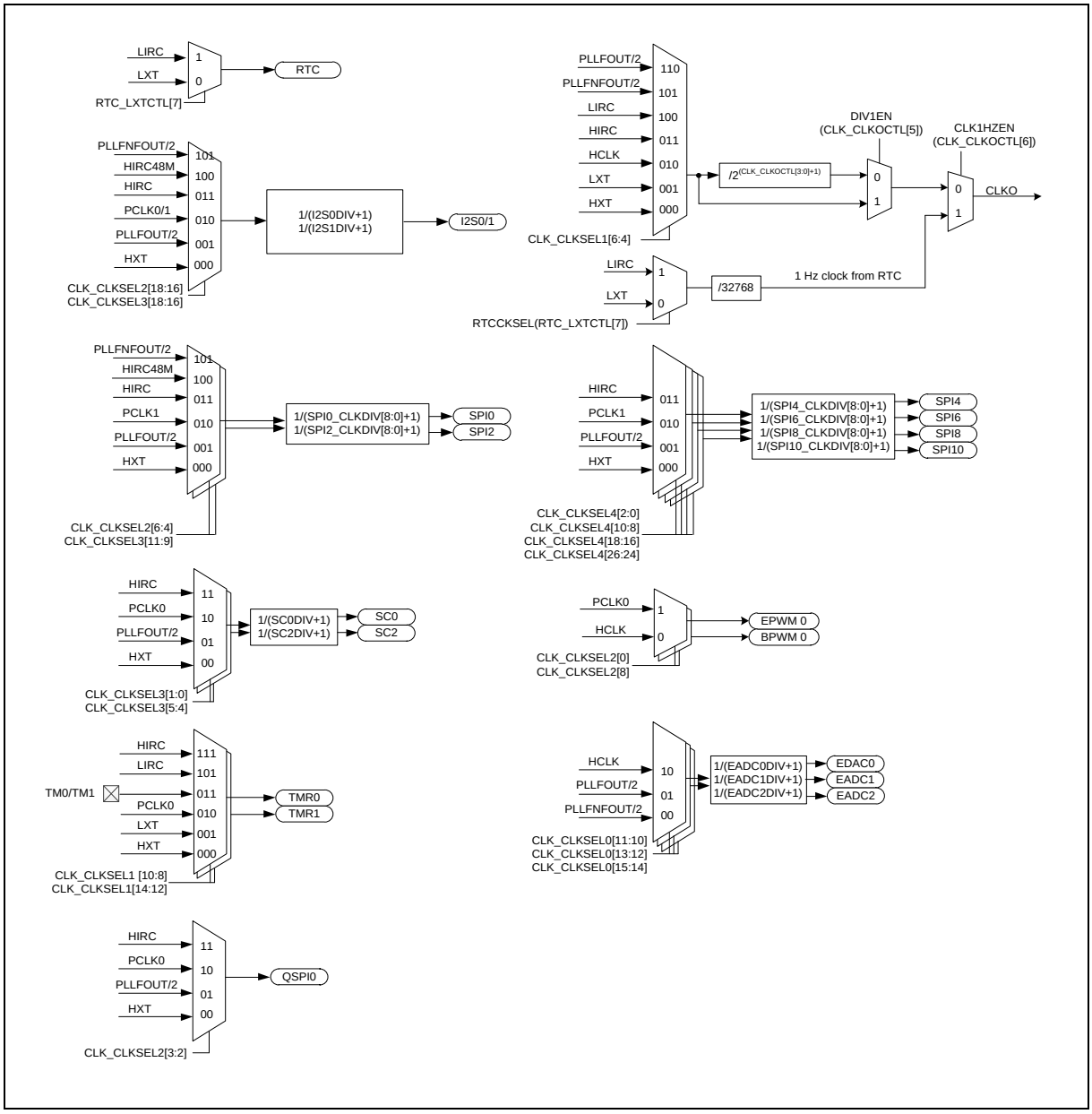


Figure 6.3-2 Clock Generator Global View Diagram (2/3)

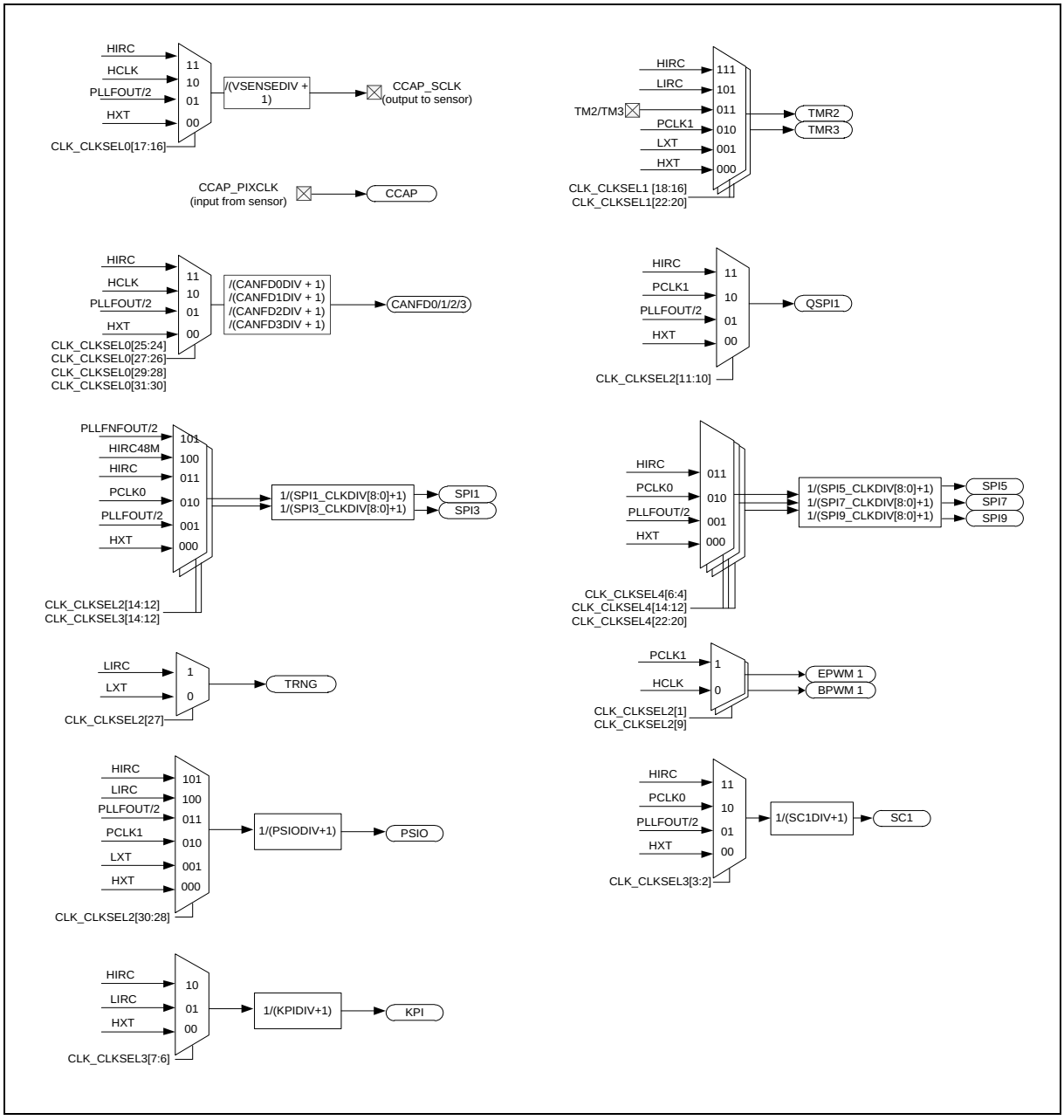


Figure 6.3-3 Clock Generator Global View Diagram (3/3)

6.3.2 Clock Generator

The clock generator consists of 7 clock sources, which are listed below:

- 32.768 kHz external low speed crystal oscillator (LXT)
- 4~24 MHz external high speed crystal oscillator (HXT)
- Programmable PLL output clock frequency (PLLFOUT), PLL source can be selected from external 4~24 MHz external high speed crystal (HXT) or 12 MHz internal high speed oscillator (HIRC)
- Programmable Audio PLL (PLLFN) output clock frequency (PLLFNFOUT), PLLFN source can be selected from external 4~24 MHz external high speed crystal (HXT) or 12 MHz internal high speed oscillator (HIRC)

- 12 MHz internal high speed RC oscillator (HIRC)
- 10 kHz internal low speed RC oscillator (LIRC)
- 48 MHz internal high speed RC oscillator (HIRC48M)

Each of these clock sources has certain stable time to wait for clock operating at stable frequency. When clock source is enabled, a stable counter start counting and correlated clock stable index is set to 1 after stable counter value reach a define value as shown in Table 6.3-1.

System and peripheral can use the clock as its operating clock only when correlate clock stable index is set to 1. The clock stable index as shown in Table 6.3-2 will auto clear when user disables the clock source.

Besides, the clock stable index of HXT, HIRC, HIRC48M, PLL and PLLFN will auto clear when chip enter power-down and clock stable counter will re-counting after chip wake-up if correlate clock is enabled.

Clock Source	Clock Stable Count Value	Clock Stable Time
HXT	4096 HXT clocks	341.33us for 12 MHz
LXT	16384 LXT clocks	500ms for 32.768 kHz
PLL	1200 PLL source clocks (CLK_PLLCTL[23]=0) 2400 PLL source clocks (CLK_PLLCTL[23]=1)	100us for 12 MHz 200us for 12 MHz
LIRC	1 LIRC clock	100us for 10 kHz
HIRC	64 HIRC clocks (CLK_PWRCTL[17:16]=0x0) 24 HIRC clocks (CLK_PWRCTL[17:16]=0x1)	5.33us for 12 MHz 2us for 12 MHz
HIRC48M	256 HIRC48M clocks	5.33us for 48 MHz
PLLFN	1200 PLL source clocks (CLK_PLLFNCTL1[28]=0) 2400 PLL source clocks (CLK_PLLFNCTL1[28]=1)	100us for 12 MHz 200us for 12 MHz

Table 6.3-1 Clock Stable Count Value Table

Clock Source	Clock Source Enable Bit	Correlated Clock Stable Index
HXT	HXTEN(CLK_PWRCTL[0])	HXTSTB(CLK_STATUS[0])
LXT	LXTEN(CLK_PWRCTL[1])	LXTSTB(CLK_STATUS[1])
PLL	PD(CLK_PLLCTL[16])	PLLSTB(CLK_STATUS[2])
LIRC	LIRCEN(CLK_PWRCTL[3])	LIRCSTB(CLK_STATUS[3])
HIRC	HIRCEN(CLK_PWRCTL[4])	HIRCSTB(CLK_STATUS[4])
HIRC48M	HIRC48MEN(CLK_PWRCTL[18])	HIRC48MSTB(CLK_STATUS[6])
PLLFN	PD(CLK_PLLFNCTL1[28])	PLLFNSTB(CLK_STATUS[10])

Table 6.3-2 Each Clock Source Enable Bit and Corresponding Stable Flag Table

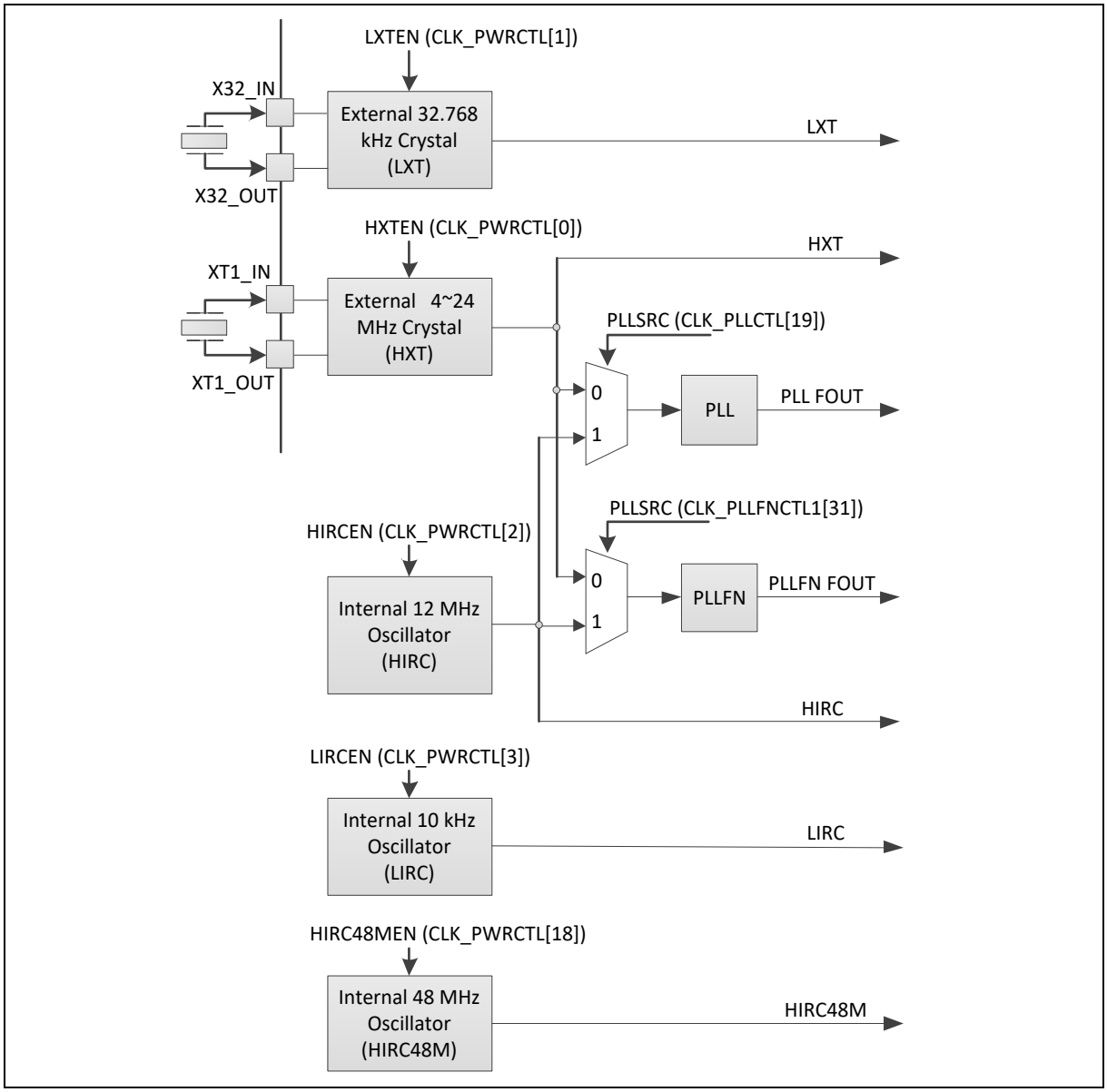


Figure 6.3-4 Clock Generator Block Diagram

6.3.3 System Clock and SysTick Clock

The system clock has 5 clock sources, which were generated from clock generator block. The clock source switch depends on the register HCLKSEL (CLK_CLKSEL0[2:0]). The block diagram is shown in Figure 6.3-5.

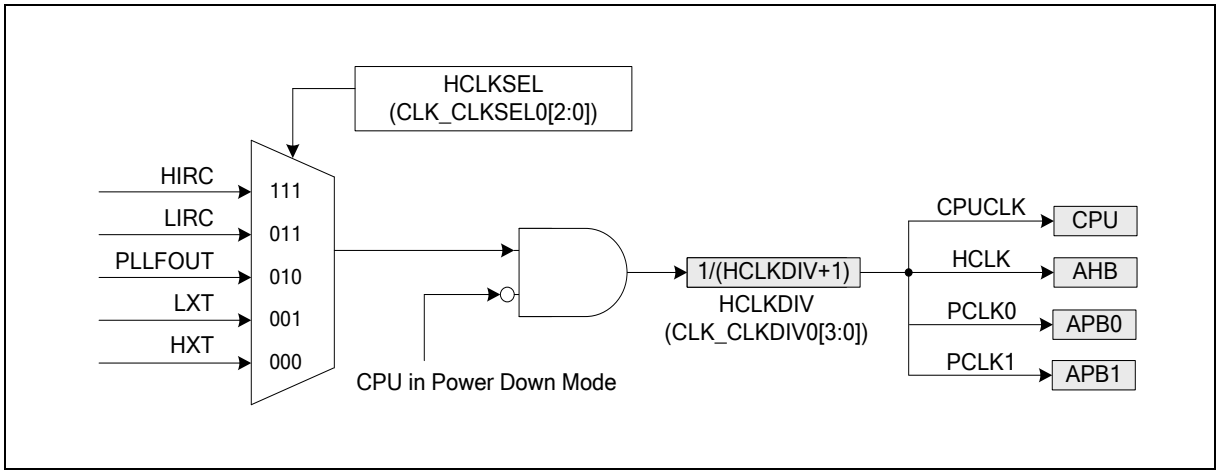


Figure 6.3-5 System Clock Block Diagrams

There are two clock fail detectors to observe HXT and LXT clock source and they have individual enable and interrupt control. When HXT detector is enabled, the HIRC clock is enabled automatically. When LXT detector is enabled, the LIRC clock is enabled automatically.

When HXT clock detector is enabled, the system clock will auto switch to HIRC if HXT clock stop being detected on the following condition: system clock source comes from HXT or system clock source comes from PLL with HXT as the input of PLL. If HXT clock stop condition is detected, the HXTFIF (CLK_CLKDSTS[0]) is set to 1 and chip will enter interrupt if HXTFIE (CLK_CLKDCTL[5]) is set to 1. User can try to recover HXT by disable HXT and enable HXT again to check if the clock stable bit is set to 1 or not. If HXT clock stable bit is set to 1, it means HXT is recover to oscillate after re-enable action and user can switch system clock to HXT again.

Figure 6.3-6 shows that the HXT clock stops detection and system clock switches to HIRC procedure.

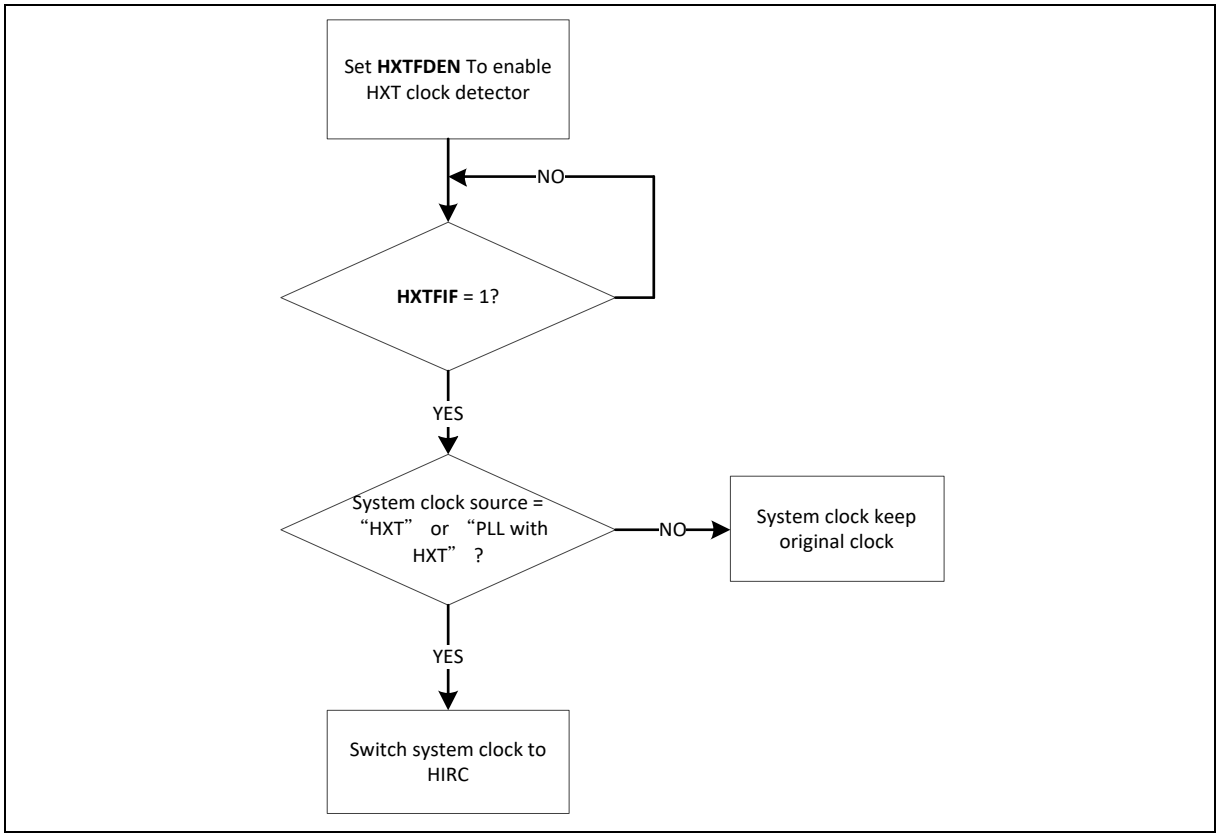


Figure 6.3-6 HXT Stop Protect Procedure

The clock source of SysTick in Cortex-M4 core can use CPU clock or external clock (SYST_CTRL[2]). If using external clock, the SysTick clock (STCLK) has 5 clock sources. The clock source switch depends on the setting of the register STCLKSEL (CLK_CLKSEL0[5:3]). The block diagram is shown in Figure 6.3-7.

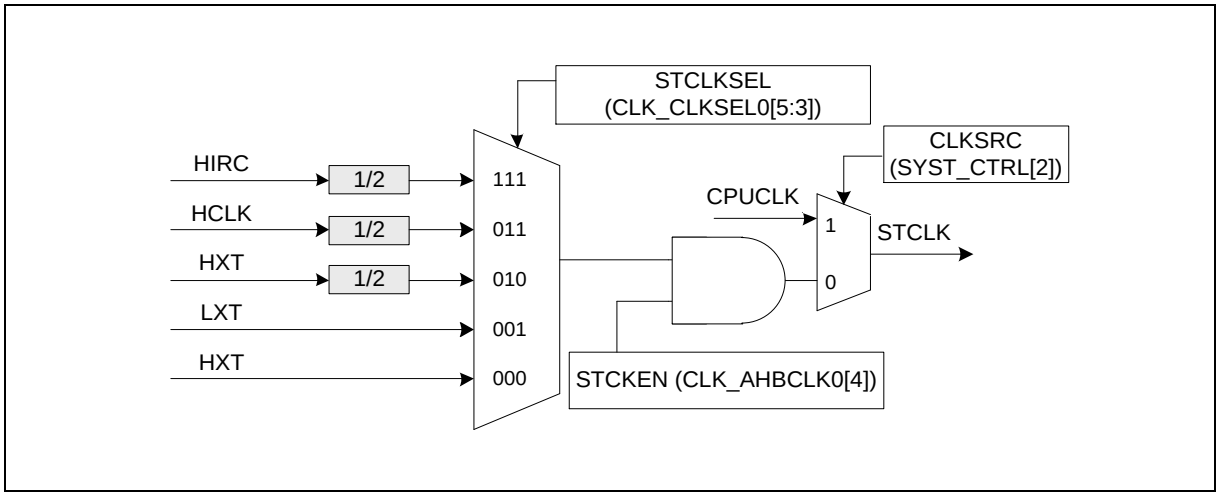


Figure 6.3-7 SysTick Clock Control Block Diagram

6.3.4 Peripherals Clock

Each peripheral clock has its own clock source selection. Refer to the CLK_CLKSEL0, CLK_CLKSEL1, CLK_CLKSEL2, CLK_CLKSEL3 and CLK_CLKSEL4 register.

6.3.5 Power-down Mode Clock

When entering Power-down mode, system clocks, some clock sources and some peripheral clocks are disabled. Some clock sources and peripherals clock are still active in Power-down mode.

The clocks, which still keep active, are listed below:

- Clock Generator
  - 10 kHz internal low speed RC oscillator (LIRC) clock
  - 32.768 kHz external low speed crystal oscillator (LXT) clock
- Peripherals Clock (When the modules adopt LXT or LIRC as clock source)

6.3.6 Clock Output

This device is equipped with a power-of-2 frequency divider that is composed of 16 chained divide-by-2 shift registers. One of the 16 shift register outputs selected by a sixteen to one multiplexer is reflected to CLKO function pin. Therefore, there are 16 options of power-of-2 divided clocks with the frequency from $F_{in}/2^1$ to $F_{in}/2^{16}$ where F_{in} is input clock frequency to the clock divider.

The output formula is $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$, where F_{in} is the input clock frequency, F_{out} is the clock divider output frequency and N is the 4-bit value in FREQSEL (CLK_CLKOCTL[3:0]).

When writing 1 to CLKOEN (CLK_CLKOCTL[4]), the chained counter starts to count. When writing 0 to CLKOEN (CLK_CLKOCTL[4]), the chained counter continuously runs till divided clock reaches low state and stays in low state.

When entering Power-down mode, clock output does not output clock even if the CKO clock source is LXT.

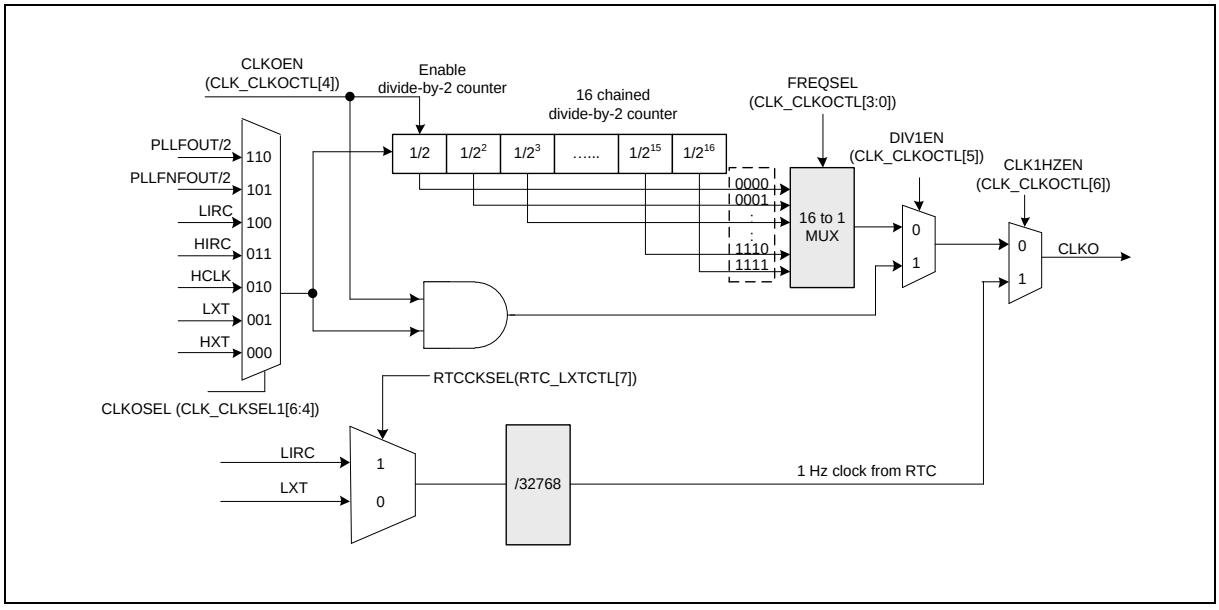


Figure 6.3-8 Clock Output Block Diagram

6.3.7 USB Clock Source

The clock sources of USB 1.1 and 2.0 systems are generated from USB2.0 PHY clock or programmable PLL output. The generated clocks are shown in Figure 6.3-9.

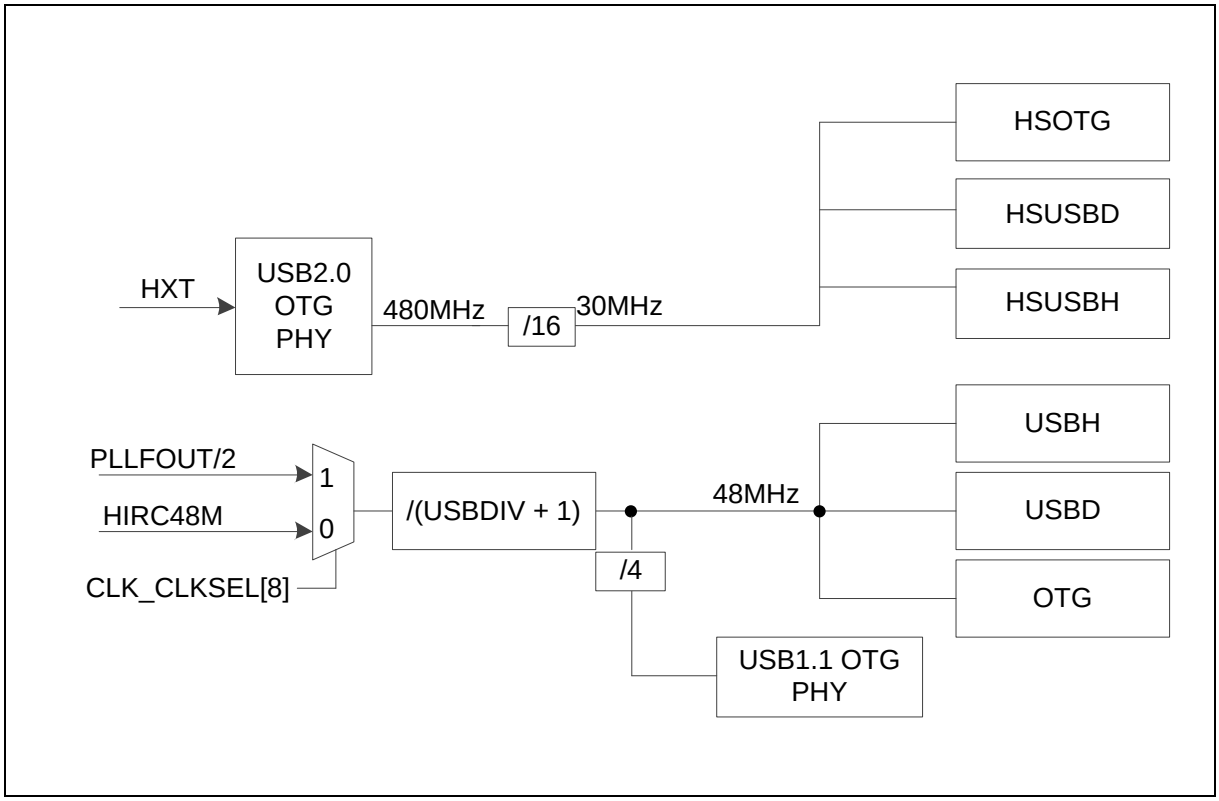


Figure 6.3-9 USB Clock Source

6.3.8 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
CLK Base Address: **CLK_BA = 0x4000_0200**				
CLK_PWRCTL	CLK_BA+0x00	R/W	System Power-down Control Register	0x0000_XX1X
CLK_AHBCLK0	CLK_BA+0x04	R/W	AHB Devices Clock Enable Control Register 0	0x0000_8004
CLK_APBCLK0	CLK_BA+0x08	R/W	APB Devices Clock Enable Control Register 0	0x0000_0001
CLK_APBCLK1	CLK_BA+0x0C	R/W	APB Devices Clock Enable Control Register 1	0x0000_0000
CLK_CLKSEL0	CLK_BA+0x10	R/W	Clock Source Select Control Register 0	0xFFF3_A93A
CLK_CLKSEL1	CLK_BA+0x14	R/W	Clock Source Select Control Register 1	0xBF77_7723
CLK_CLKSEL2	CLK_BA+0x18	R/W	Clock Source Select Control Register 2	0x20F0_2B2B
CLK_CLKSEL3	CLK_BA+0x1C	R/W	Clock Source Select Control Register 3	0xFFF0_24BF
CLK_CLKDIV0	CLK_BA+0x20	R/W	Clock Divider Number Register 0	0x0000_0000
CLK_CLKDIV1	CLK_BA+0x24	R/W	Clock Divider Number Register 1	0x0000_0000
CLK_CLKDIV2	CLK_BA+0x28	R/W	Clock Divider Number Register 2	0x0000_0000
CLK_CLKDIV3	CLK_BA+0x2C	R/W	Clock Divider Number Register 3	0x0000_0000
CLK_CLKDIV4	CLK_BA+0x30	R/W	Clock Divider Number Register 4	0x0000_0000
CLK_PCLKDIV	CLK_BA+0x34	R/W	APB Clock Divider Register	0x0000_0011
CLK_APBCLK2	CLK_BA+0x38	R/W	APB Devices Clock Enable Control Register 2	0x0000_0000
CLK_CLKDIV5	CLK_BA+0x3C	R/W	Clock Divider Number Register 5	0x0000_0000
CLK_PLLCTL	CLK_BA+0x40	R/W	PLL Control Register	0x0008_421E
CLK_PLLFNCTL0	CLK_BA+0x48	R/W	PLLFN Control Register 0	0x0000_842B
CLK_PLLFNCTL1	CLK_BA+0x4C	R/W	PLLFN Control Register 1	0x5000_0000
CLK_STATUS	CLK_BA+0x50	R	Clock Status Monitor Register	0x0000_0XXX
CLK_AHBCLK1	CLK_BA+0x58	R/W	AHB Devices Clock Enable Control Register 1	0x0000_0000
CLK_CLKSEL4	CLK_BA+0x5C	R/W	Clock Source Select Control Register 4	0x0222_2222
CLK_CLKOCTL	CLK_BA+0x60	R/W	Clock Output Control Register	0x0000_0000
CLK_CLKDCTL	CLK_BA+0x70	R/W	Clock Fail Detector Control Register	0x0000_0000
CLK_CLKDSTS	CLK_BA+0x74	R/W	Clock Fail Detector Status Register	0x0000_0000

CLK_CDUPB	CLK_BA+0x78	R/W	Clock Frequency Range Detector Upper Boundary Register	0x0000_0000
CLK_CDLOWB	CLK_BA+0x7C	R/W	Clock Frequency Range Detector Lower Boundary Register	0x0000_0000
CLK_STOPREQ	CLK_BA+0x80	R/W	Clock Stop Request Register	0x0000_0000
CLK_STOPACK	CLK_BA+0x84	R	Clock Stop Acknowledge Register	0x0000_0000
CLK_PMUCTL	CLK_BA+0x90	R/W	Power Manager Control Register	0x0000_0040
CLK_PMUSTS	CLK_BA+0x94	R/W	Power Manager Status Register	0x0000_0000
CLK_SWKDBCTL	CLK_BA+0x9C	R/W	GPIO Standby Power-down Wake-up De-bounce Control Register	0x0000_0000
CLK_PASWKCTL	CLK_BA+0xA0	R/W	GPA Standby Power-down Wake-up Control Register	0x0000_0000
CLK_PBSWKCTL	CLK_BA+0xA4	R/W	GPB Standby Power-down Wake-up Control Register	0x0000_0000
CLK_PCSWKCTL	CLK_BA+0xA8	R/W	GPC Standby Power-down Wake-up Control Register	0x0000_0000
CLK_PDSWKCTL	CLK_BA+0xAC	R/W	GPD Standby Power-down Wake-up Control Register	0x0000_0000
CLK_IOPDCTL	CLK_BA+0xB0	R/W	GPIO Standby Power-down Control Register	0x0000_0000

6.3.9 Register Description

System Power-down Control Register (CLK_PWRCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PWRCTL	CLK_BA+0x00	R/W	System Power-down Control Register	0x0000_XX1X

31	30	29	28	27	26	25	24
HXTMD	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**HIRC48MEN**	**HIRCSTBS**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**HXTSELTYP**	**HXTGAIN**		**Reserved**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDEN	**PDWKIF**	**PDWKIEN**	**PDWKDLY**	**LIRCEN**	**HIRCEN**	**LXTEN**	**HXTEN**

Bits	Description	
[31]	**HXTMD**	**HXT Mode Selection (Write Protect)** 0 = HXT work as crystal mode. PF.2 and PF.3 are configured as external high speed crystal (HXT) pins. 1 = HXT works as external clock mode. PF.3 is configured as external clock input pin. **Note 1:** This bit is write protected. Refer to the SYS_REGCTL register. **Note 2:** When external clock mode enable, HXTSELTYP(CLK_PWRCTL[12]) must be set as GM type.
[30:19]	**Reserved**	Reserved.
[18]	**HIRC48MEN**	**HIRC48M Enable Bit (Write Protect)** 0 = 48 MHz internal high speed RC oscillator (HIRC48M) Disabled. 1 = 48 MHz internal high speed RC oscillator (HIRC48M) Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[17:16]	**HIRCSTBS**	**HIRC Stable Count Select (Write Protect)** 00 = HIRC stable count is 64 clocks. 01 = HIRC stable count is 24 clocks. Others = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15:13]	**Reserved**	Reserved.
[12]	**HXTSELTYP**	**HXT Crystal Type Select Bit (Write Protect)** 0 = Select INV type. 1 = Select GM type. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11:10]	**HXTGAIN**	**HXT Gain Control Bit (Write Protect)** Gain control is used to enlarge the gain of crystal to make sure crystal work normally. 00 = HXT frequency is lower than from 8 MHz.

		01 = HXT frequency is from 8 MHz to 12 MHz. 10 = HXT frequency is from 12 MHz to 16 MHz. 11 = HXT frequency is higher than 16 MHz. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[9:8]	**Reserved**	Reserved.
[7]	**PDEN**	**System Power-down Enable (Write Protect)** When this bit is set to 1, Power-down mode is enabled and chip keeps active till the CPU sleep mode is also active and then the chip enters Power-down mode. When chip wakes up from Power-down mode, this bit is auto cleared. Users need to set this bit again for next Power-down. In Power-down mode, HXT and the HIRC will be disabled in this mode, but LXT and LIRC are not controlled by Power-down mode. In Power-down mode, the PLL, PLLFN and system clock are disabled, and ignored the clock source selection. The clocks of peripheral are not controlled by Power-down mode, if the peripheral clock source is from LXT or LIRC. 0 = Chip will not enter Power-down mode after CPU sleep command WFI. 1 = Chip enters Power-down mode after CPU sleep command WFI. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** System tick interrupt TICKINT(SYS_CTRL[1]) has to be disabled before entering to Power-down mode, to avoid system tick interrupt may influence system not entering power-down mode and keep operation.
[6]	**PDWKIF**	**Power-down Mode Wake-up Interrupt Status** Set by “Power-down wake-up event”, it indicates that resume from Power-down mode. The flag is set if any wake-up source occurred. Refer Power Modes and Wake-up Sources chapter. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** This bit works only if PDWKIEN (CLK_PWRCTL[5]) set to 1.
[5]	**PDWKIEN**	**Power-down Mode Wake-up Interrupt Enable Bit (Write Protect)** 0 = Power-down mode wake-up interrupt Disabled. 1 = Power-down mode wake-up interrupt Enabled. **Note 1:** The interrupt will occur when both PDWKIF and PDWKIEN are high. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**PDWKDLY**	**Enable the Wake-up Delay Counter (Write Protect)** When the chip wakes up from Power-down mode, the clock control will delay certain clock cycles to wait system clock stable. The delayed clock cycle is 4096 clock cycles when chip works at 4~24 MHz external high speed crystal oscillator (HXT), and 64 or 24 clock cycles when chip works at 12 MHz internal high speed RC oscillator (HIRC). 0 = Clock cycles delay Disabled. 1 = Clock cycles delay Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**LIRCEN**	**LIRC Enable Bit (Write Protect)** 0 = 10 kHz internal low speed RC oscillator (LIRC) Disabled. 1 = 10 kHz internal low speed RC oscillator (LIRC) Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**HIRCEN**	**HIRC Enable Bit (Write Protect)** 0 = 12 MHz internal high speed RC oscillator (HIRC) Disabled. 1 = 12 MHz internal high speed RC oscillator (HIRC) Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[1]	**LXTEN**	**LXT Enable Bit (Write Protect)** 0 = 32.768 kHz external low speed crystal (LXT) Disabled. 1 = 32.768 kHz external low speed crystal (LXT) Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**HXTEN**	**HXT Enable Bit (Write Protect)** 0 = 4~24 MHz external high speed crystal (HXT) Disabled. 1 = 4~24 MHz external high speed crystal (HXT) Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

AHB Devices Clock Enable Control Register 0 (CLK_AHBCLK0)

The bits in this register are used to enable/disable clock for system clock, AHB bus devices clock.

Register	Offset	R/W	Description	Reset Value
CLK_AHBCLK0	CLK_BA+0x04	R/W	AHB Devices Clock Enable Control Register 0	0x0000_8004

31	30	29	28	27	26	25	24
GPHCKEN	GPGCKEN	GPFCKEN	GPECKEN	GPDCKEN	GPCCKEN	GPBCKEN	GPACKEN
23	22	21	20	19	18	17	16
Reserved				TRACECKEN	PDMA1CKEN	SDH1CKEN	USBHCKEN
15	14	13	12	11	10	9	8
FMCIDLE	SPIMCKEN	KSCKEN	CRPTCKEN	HBICKEN	HSUSBDCKEN	SENCKEN	CCAPCKEN
7	6	5	4	3	2	1	0
CRCCKEN	SDH0CKEN	EMAC0CKEN	STCKEN	EBICKEN	ISPCKEN	PDMA0CKEN	Reserved

Bits	Description	
[31]	**GPHCKEN**	**GPIOH Clock Enable Bit** 0 = GPIOH clock Disabled. 1 = GPIOH clock Enabled.
[30]	**GPGCKEN**	**GPIOG Clock Enable Bit** 0 = GPIOG clock Disabled. 1 = GPIOG clock Enabled.
[29]	**GPFCKEN**	**GPIOF Clock Enable Bit** 0 = GPIOF clock Disabled. 1 = GPIOF clock Enabled.
[28]	**GPECKEN**	**GPIOE Clock Enable Bit** 0 = GPIOE clock Disabled. 1 = GPIOE clock Enabled.
[27]	**GPDCKEN**	**GPIOD Clock Enable Bit** 0 = GPIOD clock Disabled. 1 = GPIOD clock Enabled.
[26]	**GPCCKEN**	**GPIOC Clock Enable Bit** 0 = GPIOC clock Disabled. 1 = GPIOC clock Enabled.
[25]	**GPBCKEN**	**GPIOB Clock Enable Bit** 0 = GPIOB clock Disabled. 1 = GPIOB clock Enabled.
[24]	**GPACKEN**	**GPIOA Clock Enable Bit** 0 = GPIOA clock Disabled. 1 = GPIOA clock Enabled.

[23:20]	**Reserved**	Reserved.
[19]	**TRACECKEN**	**TRACE Clock Enable Bit** 0 = TRACE clock Disabled. 1 = TRACE clock Enabled.
[18]	**PDMA1CKEN**	**PDMA1 Clock Enable Bit** 0 = PDMA1 clock Disabled. 1 = PDMA1 clock Enabled.
[17]	**SDH1CKEN**	**SDH1 Controller Clock Enable Bit** 0 = SDH1 clock Disabled. 1 = SDH1 clock Enabled.
[16]	**USBHCKEN**	**USB HOST Controller Clock Enable Bit** 0 = USB HOST peripheral clock Disabled. 1 = USB HOST peripheral clock Enabled.
[15]	**FMCIDLE**	**Flash Memory Controller Clock Enable Bit in IDLE Mode** 0 = FMC clock Disabled when chip is under IDLE mode. 1 = FMC clock Enabled when chip is under IDLE mode.
[14]	**SPIMCKEN**	**SPIM Controller Clock Enable Bit** 0 = SPIM controller clock Disabled. 1 = SPIM controller clock Enabled.
[13]	**KSCKEN**	**Key Store Clock Enable Bit** 0 = Key Store clock Disabled. 1 = Key Store clock Enabled.
[12]	**CRPTCKEN**	**Cryptographic Accelerator Clock Enable Bit** 0 = Cryptographic Accelerator clock Disabled. 1 = Cryptographic Accelerator clock Enabled.
[11]	**HBICKEN**	**HyperBus Interface Clock Enable Bit** 0 = HBI clock Disabled. 1 = HBI clock Enabled.
[10]	**HSUSBDCKEN**	**HSUSB Device Clock Enable Bit** 0 = HSUSB device controller clock Disabled. 1 = HSUSB device controller clock Enabled.
[9]	**SENCKEN**	**CCAP Sensor Clock Enable Control** 0 = CCAP Sensor clock Disabled. 1 = CCAP Sensor clock Enabled.
[8]	**CCAPCKEN**	**Camera Capture Interface Controller Clock Enable Control** 0 = CCAP controller clock Disabled. 1 = CCAP controller clock Enabled.
[7]	**CRCCKEN**	**CRC Generator Controller Clock Enable Bit** 0 = CRC peripheral clock Disabled. 1 = CRC peripheral clock Enabled.
[6]	**SDH0CKEN**	**SDH0 Controller Clock Enable Bit** 0 = SDH0 clock Disabled. 1 = SDH0 clock Enabled.

[5]	**EMAC0CKEN**	**EMAC0 Controller Clock Enable Bit** 0 = EMAC0 controller clock Disabled. 1 = EMAC0 controller clock Enabled.
[4]	**STCKEN**	**System Tick Clock Enable Bit** 0 = System tick clock Disabled. 1 = System tick clock Enabled.
[3]	**EBICKEN**	**EBI Controller Clock Enable Bit** 0 = EBI peripheral clock Disabled. 1 = EBI peripheral clock Enabled.
[2]	**ISPCKEN**	**Flash ISP Controller Clock Enable Bit** 0 = Flash ISP peripheral clock Disabled. 1 = Flash ISP peripheral clock Enabled.
[1]	**PDMA0CKEN**	**PDMA0 Controller Clock Enable Bit** 0 = PDMA0 peripheral clock Disabled. 1 = PDMA0 peripheral clock Enabled.
[0]	**Reserved**	Reserved.

APB Devices Clock Enable Control Register 0 (CLK_APBCLK0)

The bits in this register are used to enable/disable clock for peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
CLK_APBCLK0	CLK_BA+0x08	R/W	APB Devices Clock Enable Control Register 0	0x0000_0001

31	30	29	28	27	26	25	24
Reserved	HSOTGCKEN	I2S0CKEN	EADC0CKEN	USBDCKEN	OTGCKEN	Reserved	
23	22	21	20	19	18	17	16
UART7CKEN	UART6CKEN	UART5CKEN	UART4CKEN	UART3CKEN	UART2CKEN	UART1CKEN	UART0CKEN
15	14	13	12	11	10	9	8
SPI2CKEN	SPI1CKEN	SPI0CKEN	QSPI0CKEN	I2C3CKEN	I2C2CKEN	I2C1CKEN	I2C0CKEN
7	6	5	4	3	2	1	0
ACMP01CKEN	CLKOCKEN	TMR3CKEN	TMR2CKEN	TMR1CKEN	TMR0CKEN	RTCCKEN	WDTCKEN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**HSOTGCKEN**	**HSUSB OTG Clock Enable Bit** 0 = HSUSB OTG clock Disabled. 1 = HSUSB OTG clock Enabled.
[29]	**I2S0CKEN**	**I2S0 Clock Enable Bit** 0 = I2S0 clock Disabled. 1 = I2S0 clock Enabled.
[28]	**EADC0CKEN**	**EADC0 Clock Enable Bit** 0 = EADC0 clock Disabled. 1 = EADC0 clock Enabled.
[27]	**USBDCKEN**	**USB Device Clock Enable Bit** 0 = USB device clock Disabled. 1 = USB device clock Enabled.
[26]	**OTGCKEN**	**USB OTG Clock Enable Bit** 0 = USB OTG clock Disabled. 1 = USB OTG clock Enabled.
[25:24]	**Reserved**	Reserved.
[23]	**UART7CKEN**	**UART7 Clock Enable Bit** 0 = UART7 clock Disabled. 1 = UART7 clock Enabled.
[22]	**UART6CKEN**	**UART6 Clock Enable Bit** 0 = UART6 clock Disabled. 1 = UART6 clock Enabled.
[21]	**UART5CKEN**	**UART5 Clock Enable Bit**

		0 = UART5 clock Disabled. 1 = UART5 clock Enabled.
[20]	**UART4CKEN**	**UART4 Clock Enable Bit** 0 = UART4 clock Disabled. 1 = UART4 clock Enabled.
[19]	**UART3CKEN**	**UART3 Clock Enable Bit** 0 = UART3 clock Disabled. 1 = UART3 clock Enabled.
[18]	**UART2CKEN**	**UART2 Clock Enable Bit** 0 = UART2 clock Disabled. 1 = UART2 clock Enabled.
[17]	**UART1CKEN**	**UART1 Clock Enable Bit** 0 = UART1 clock Disabled. 1 = UART1 clock Enabled.
[16]	**UART0CKEN**	**UART0 Clock Enable Bit** 0 = UART0 clock Disabled. 1 = UART0 clock Enabled.
[15]	**SPI2CKEN**	**SPI2 Clock Enable Bit** 0 = SPI2 clock Disabled. 1 = SPI2 clock Enabled.
[14]	**SPI1CKEN**	**SPI1 Clock Enable Bit** 0 = SPI1 clock Disabled. 1 = SPI1 clock Enabled.
[13]	**SPI0CKEN**	**SPI0 Clock Enable Bit** 0 = SPI0 clock Disabled. 1 = SPI0 clock Enabled.
[12]	**QSPI0CKEN**	**QSPI0 Clock Enable Bit** 0 = QSPI0 clock Disabled. 1 = QSPI0 clock Enabled.
[11]	**I2C3CKEN**	**I2C3 Clock Enable Bit** 0 = I2C3 clock Disabled. 1 = I2C3 clock Enabled.
[10]	**I2C2CKEN**	**I2C2 Clock Enable Bit** 0 = I2C2 clock Disabled. 1 = I2C2 clock Enabled.
[9]	**I2C1CKEN**	**I2C1 Clock Enable Bit** 0 = I2C1 clock Disabled. 1 = I2C1 clock Enabled.
[8]	**I2C0CKEN**	**I2C0 Clock Enable Bit** 0 = I2C0 clock Disabled. 1 = I2C0 clock Enabled.
[7]	**ACMP01CKEN**	**Analog Comparator 0/1 Clock Enable Bit** 0 = Analog comparator 0/1 clock Disabled. 1 = Analog comparator 0/1 clock Enabled.

[6]	**CLKOCKEN**	**CLKO Clock Enable Bit** 0 = CLKO clock Disabled. 1 = CLKO clock Enabled.
[5]	**TMR3CKEN**	**Timer3 Clock Enable Bit** 0 = Timer3 clock Disabled. 1 = Timer3 clock Enabled.
[4]	**TMR2CKEN**	**Timer2 Clock Enable Bit** 0 = Timer2 clock Disabled. 1 = Timer2 clock Enabled.
[3]	**TMR1CKEN**	**Timer1 Clock Enable Bit** 0 = Timer1 clock Disabled. 1 = Timer1 clock Enabled.
[2]	**TMR0CKEN**	**Timer0 Clock Enable Bit** 0 = Timer0 clock Disabled. 1 = Timer0 clock Enabled.
[1]	**RTCCKEN**	**Real-time-clock APB Interface Clock Enable Bit** This bit is used to control the RTC APB clock only. The RTC peripheral clock source is selected from RTCCKSEL(RTC_LXTCTL[7]). It can be selected to 32.768 kHz external low speed crystal (LXT) or 10 kHz internal low speed RC oscillator (LIRC). 0 = RTC clock Disabled. 1 = RTC clock Enabled.
[0]	**WDTCKEN**	**Watchdog Timer Clock Enable Bit (Write Protect)** 0 = Watchdog timer clock Disabled. 1 = Watchdog timer clock Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

APB Devices Clock Enable Control Register 1 (CLK_APBCLK1)

The bits in this register are used to enable/disable clock for peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
CLK_APBCLK1	CLK_BA+0x0C	R/W	APB Devices Clock Enable Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
EADC1CKEN	Reserved	I2S1CKEN	Reserved	ECAP1CKEN	ECAP0CKEN	TRNGCKEN	Reserved
23	22	21	20	19	18	17	16
EQEI1CKEN	EQEI0CKEN	EQEI3CKEN	EQEI2CKEN	BPWM1CKEN	BPWM0CKEN	EPWM1CKEN	EPWM0CKEN
15	14	13	12	11	10	9	8
Reserved	ECAP3CKEN	ECAP2CKEN	DACCKEN	Reserved	PSIOCKEN	Reserved	USCI0CKEN
7	6	5	4	3	2	1	0
SPI4CKEN	SPI3CKEN	Reserved	QSPI1CKEN	I2C4CKEN	SC2CKEN	SC1CKEN	SC0CKEN

Bits	Description	
[31]	**EADC1CKEN**	**EADC1 Clock Enable Bit** 0 = EADC1 clock Disabled. 1 = EADC1 clock Enabled.
[30]	**Reserved**	Reserved.
[29]	**I2S1CKEN**	**I2S1 Clock Enable Bit** 0 = I2S1 clock Disabled. 1 = I2S1 clock Enabled.
[28]	**Reserved**	Reserved.
[27]	**ECAP1CKEN**	**ECAP1 Clock Enable Bit** 0 = ECAP1 clock Disabled. 1 = ECAP1 clock Enabled.
[26]	**ECAP0CKEN**	**ECAP0 Clock Enable Bit** 0 = ECAP0 clock Disabled. 1 = ECAP0 clock Enabled.
[25]	**TRNGCKEN**	**TRNG Clock Enable Bit** 0 = TRNG clock Disabled. 1 = TRNG clock Enabled.
[24]	**Reserved**	Reserved.
[23]	**EQEI1CKEN**	**EQEI1 Clock Enable Bit** 0 = EQEI1 clock Disabled. 1 = EQEI1 clock Enabled.
[22]	**EQEI0CKEN**	**EQEI0 Clock Enable Bit** 0 = EQEI0 clock Disabled. 1 = EQEI0 clock Enabled.

[21]	**EQEI3CKEN**	**EQEI3 Clock Enable Bit** 0 = QEI3 clock Disabled. 1 = QEI3 clock Enabled.
[20]	**EQEI2CKEN**	**EQEI2 Clock Enable Bit** 0 = EQEI2 clock Disabled. 1 = EQEI2 clock Enabled.
[19]	**BPWM1CKEN**	**BPWM1 Clock Enable Bit** 0 = BPWM1 clock Disabled. 1 = BPWM1 clock Enabled.
[18]	**BPWM0CKEN**	**BPWM0 Clock Enable Bit** 0 = BPWM0 clock Disabled. 1 = BPWM0 clock Enabled.
[17]	**EPWM1CKEN**	**EPWM1 Clock Enable Bit** 0 = EPWM1 clock Disabled. 1 = EPWM1 clock Enabled.
[16]	**EPWM0CKEN**	**EPWM0 Clock Enable Bit** 0 = EPWM0 clock Disabled. 1 = EPWM0 clock Enabled.
[15]	**Reserved**	Reserved.
[14]	**ECAP3CKEN**	**ECAP3 Clock Enable Bit** 0 = ECAP3 clock Disabled. 1 = ECAP3 clock Enabled.
[13]	**ECAP2CKEN**	**ECAP2 Clock Enable Bit** 0 = ECAP2 clock Disabled. 1 = ECAP2 clock Enabled.
[12]	**DACCKEN**	**DAC Clock Enable Bit** 0 = DAC clock Disabled. 1 = DAC clock Enabled.
[11]	**Reserved**	Reserved.
[10]	**PSIOCKEN**	**PSIO Clock Enable Bit** 0 = PSIO clock Disabled. 1 = PSIO clock Enabled.
[9]	**Reserved**	Reserved.
[8]	**USCI0CKEN**	**USCI0 Clock Enable Bit** 0 = USCI0 clock Disabled. 1 = USCI0 clock Enabled.
[7]	**SPI4CKEN**	**SPI4 Clock Enable Bit** 0 = SPI4 clock Disabled. 1 = SPI4 clock Enabled.
[6]	**SPI3CKEN**	**SPI3 Clock Enable Bit** 0 = SPI3 clock Disabled. 1 = SPI3 clock Enabled.
[5]	**Reserved**	Reserved.

[4]	**QSPI1CKEN**	**QSPI1 Clock Enable Bit** 0 = QSPI1 clock Disabled. 1 = QSPI1 clock Enabled.
[3]	**I2C4CKEN**	**I2C4 Clock Enable Bit** 0 = I2C4 clock Disabled. 1 = I2C4 clock Enabled.
[2]	**SC2CKEN**	**SC2 Clock Enable Bit** 0 = SC2 clock Disabled. 1 = SC2 clock Enabled.
[1]	**SC1CKEN**	**SC1 Clock Enable Bit** 0 = SC1 clock Disabled. 1 = SC1 clock Enabled.
[0]	**SC0CKEN**	**SC0 Clock Enable Bit** 0 = SC0 clock Disabled. 1 = SC0 clock Enabled.

Clock Source Select Control Register 0 (CLK_CLKSEL0)

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL0	CLK_BA+0x10	R/W	Clock Source Select Control Register 0	0xFFF3_A93A

31	30	29	28	27	26	25	24
CANFD3SEL		CANFD2SEL		CANFD1SEL		CANFD0SEL	
23	22	21	20	19	18	17	16
SDH1SEL		SDH0SEL		Reserved		CCAPSEL	
15	14	13	12	11	10	9	8
EADC2SEL		EADC1SEL		EADC0SEL		Reserved	USBSEL
7	6	5	4	3	2	1	0
Reserved		STCLKSEL			HCLKSEL		

Bits	Description	
[31:30]	**CANFD3SEL**	**CANFD3 Clock Source Selection (Write Protect)** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2 clock. 10 = Clock source from HCLK. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[29:28]	**CANFD2SEL**	**CANFD2 Clock Source Selection (Write Protect)** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2 clock. 10 = Clock source from HCLK. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[27:26]	**CANFD1SEL**	**CANFD1 Clock Source Selection (Write Protect)** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2 clock. 10 = Clock source from HCLK. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[25:24]	**CANFD0SEL**	**CANFD0 Clock Source Selection (Write Protect)** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2 clock. 10 = Clock source from HCLK. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[23:22]	**SDH1SEL**	**SDH1 Clock Source Selection (Write Protect)** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT).

		01 = Clock source from PLL/2 clock. 10 = Clock source from HCLK. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[21:20]	**SDH0SEL**	**SDH0 Clock Source Selection (Write Protect)** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2 clock. 10 = Clock source from HCLK. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[19:18]	**Reserved**	Reserved.
[17:16]	**CCAPSEL**	**CCAP Sensor Clock Source Selection (Write Protect)** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from HCLK. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15:14]	**EADC2SEL**	**EADC2 Clock Source Selection (Write Protect)** 00 = Clock source from PLLFN/2. 01 = Clock source from PLL/2. 10 = Clock source from HCLK. 11 = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[13:12]	**EADC1SEL**	**EADC1 Clock Source Selection (Write Protect)** 00 = Clock source from PLLFN/2. 01 = Clock source from PLL/2. 10 = Clock source from HCLK. 11 = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[11:10]	**EADC0SEL**	**EADC0 Clock Source Selection (Write Protect)** 00 = Clock source from PLLFN/2. 01 = Clock source from PLL/2. 10 = Clock source from HCLK. 11 = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[9]	**Reserved**	Reserved.
[8]	**USBSEL**	**USB Clock Source Selection (Write Protect)** 0 = Clock source from 48 MHz internal high speed RC oscillator (HIRC48M). 1 = Clock source from PLL/2. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7:6]	**Reserved**	Reserved.
[5:3]	**STCLKSEL**	**Cortex-M4 SysTick Clock Source Selection (Write Protect)** If SYST_CTRL[2]=0, SysTick uses listed clock source below. 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT).

		010 = Clock source from HXT/2. 011 = Clock source from HCLK/2. 111 = Clock source from HIRC/2. **Note 1:** if SysTick clock source is not from HCLK (i.e. SYST_CTRL[2] = 0), SysTick needs to enable STCKEN(CLK_AHBCLK0[4]). SysTick clock source must less than or equal to HCLK/2. **Note 2:** These bits are write protected. Refer to the SYS_REGLCTL register.
[2:0]	**HCLKSEL**	**HCLK Clock Source Selection (Write Protect)** Before clock switching, the related clock sources (both pre-select and new-select) must be turned on. 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PLL. 011 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved. **Note:** Theses bits are write protected. Refer to the SYS_REGLCTL register.

Clock Source Select Control Register 1 (CLK_CLKSEL1)

Before clock switching, the related clock sources (pre-selected and newly-selected) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL1	CLK_BA+0x14	R/W	Clock Source Select Control Register 1	0xBF77_7723

31	30	29	28	27	26	25	24
WWDTSEL		Reserved		UART1SEL		UART0SEL	
23	22	21	20	19	18	17	16
Reserved	TMR3SEL			Reserved	TMR2SEL		
15	14	13	12	11	10	9	8
Reserved	TMR1SEL			Reserved	TMR0SEL		
7	6	5	4	3	2	1	0
Reserved	CLKOSEL			Reserved		WDTSEL	

Bits	Description	
[31:30]	**WWDTSEL**	**Window Watchdog Timer Clock Source Selection (Write Protect)** 10 = Clock source from HCLK/2048. 11 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). Others = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[29:28]	**Reserved**	Reserved.
[27:26]	**UART1SEL**	**UART1 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[25:24]	**UART0SEL**	**UART0 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[23]	**Reserved**	Reserved.
[22:20]	**TMR3SEL**	**TIMER3 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK1. 011 = Clock source from external clock TM3 pin. 101 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.

[19]	**Reserved**	Reserved.
[18:16]	**TMR2SEL**	**TIMER2 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK1. 011 = Clock source from external clock TM2 pin. 101 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[15]	**Reserved**	Reserved.
[14:12]	**TMR1SEL**	**TIMER1 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK0. 011 = Clock source from external clock TM1 pin. 101 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[11]	**Reserved**	Reserved.
[10:8]	**TMR0SEL**	**TIMER0 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK0. 011 = Clock source from external clock TM0 pin. 101 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[7]	**Reserved**	Reserved.
[6:4]	**CLKOSEL**	**Clock Output Clock Source Selection (Write Protect)** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from HCLK. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 101 = Clock source from PLLFN/2. 110 = Clock source from PLL/2. 111 = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[3:2]	**Reserved**	Reserved.
[1:0]	**WDTSEL**	**Watchdog Timer Clock Source Selection (Write Protect)** 00 = Reserved. 01 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 10 = Clock source from HCLK/2048. 11 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Clock Source Select Control Register 2 (CLK_CLKSEL2)

Before clock switching, the related clock sources (pre-select and new-select) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL2	CLK_BA+0x18	R/W	Clock Source Select Control Register 2	0x20F0_2B2B

31	30	29	28	27	26	25	24
Reserved	PSIOSEL			TRNGSEL	Reserved		
23	22	21	20	19	18	17	16
UART9SEL		UART8SEL		Reserved	I2S1SEL		
15	14	13	12	11	10	9	8
Reserved	SPI1SEL			QSPI1SEL		BPWM1SEL	BPWM0SEL
7	6	5	4	3	2	1	0
Reserved	SPI0SEL			QSPI0SEL		EPWM1SEL	EPWM0SEL

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**PSIOSEL**	**PSIO Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK1. 011 = Clock source from PLL/2. 100 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 101 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[27]	**TRNGSEL**	**TRNG Clock Source Selection** 0 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 1 = Clock source from 10 kHz internal low speed RC oscillator (LIRC).
[26:24]	**Reserved**	Reserved.
[23:22]	**UART9SEL**	**UART9 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[21:20]	**UART8SEL**	**UART8 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[19]	**Reserved**	Reserved.
[18:16]	**I2S1SEL**	**I2S1 Clock Source Selection**

		000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK1. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from 48 MHz internal high speed RC oscillator (HIRC48M). 101 = Clock source from PLLFN/2. Others = Reserved.
[15]	**Reserved**	Reserved.
[14:12]	**SPI1SEL**	**SPI1 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK0. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from 48 MHz internal high speed RC oscillator (HIRC48M). 101 = Clock source from PLLFN/2. Others = Reserved.
[11:10]	**QSPI1SEL**	**QSPI1 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from PCLK1. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[9]	**BPWM1SEL**	**BPWM1 Clock Source Selection** The peripheral clock source of BPWM1 is defined by BPWM1SEL. 0 = Clock source from HCLK. 1 = Clock source from PCLK1.
[8]	**BPWM0SEL**	**BPWM0 Clock Source Selection** The peripheral clock source of BPWM0 is defined by BPWM0SEL. 0 = Clock source from HCLK. 1 = Clock source from PCLK0.
[7]	**Reserved**	Reserved.
[6:4]	**SPI0SEL**	**SPI0 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK1. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from 48 MHz internal high speed RC oscillator (HIRC48M). 101 = Clock source from PLLFN/2. Others = Reserved.
[3:2]	**QSPI0SEL**	**QSPI0 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from PCLK0. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[1]	**EPWM1SEL**	**EPWM1 Clock Source Selection** The peripheral clock source of EPWM1 is defined by EPWM1SEL. 0 = Clock source from HCLK.

		1 = Clock source from PCLK1.
[0]	**EPWM0SEL**	**EPWM0 Clock Source Selection** The peripheral clock source of EPWM0 is defined by EPWM0SEL. 0 = Clock source from HCLK. 1 = Clock source from PCLK0.

Clock Source Select Control Register 3 (CLK_CLKSEL3)

Before clock switching, the related clock sources (pre-select and new-select) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL3	CLK_BA+0x1C	R/W	Clock Source Select Control Register 3	0xFFF0_24BF

31	30	29	28	27	26	25	24
UART5SEL		UART4SEL		UART3SEL		UART2SEL	
23	22	21	20	19	18	17	16
UART7SEL		UART6SEL		Reserved	I2S0SEL		
15	14	13	12	11	10	9	8
Reserved	SPI3SEL			SPI2SEL			Reserved
7	6	5	4	3	2	1	0
KPISEL		SC2SEL		SC1SEL		SC0SEL	

Bits	Description	
[31:30]	**UART5SEL**	**UART5 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[29:28]	**UART4SEL**	**UART4 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[27:26]	**UART3SEL**	**UART3 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[25:24]	**UART2SEL**	**UART2 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[23:22]	**UART7SEL**	**UART7 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[21:20]	**UART6SEL**	**UART6 Clock Source Selection**

		00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[19]	**Reserved**	Reserved.
[18:16]	**I2S0SEL**	**I2S0 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK0. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from 48 MHz internal high speed RC oscillator (HIRC48M). 101 = Clock source from PLLFN/2. Others = Reserved.
[15]	**Reserved**	Reserved.
[14:12]	**SPI3SEL**	**SPI3 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK0. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from 48 MHz internal high speed RC oscillator (HIRC48M). 101 = Clock source from PLLFN/2. Others = Reserved.
[11:9]	**SPI2SEL**	**SPI2 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK1. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 100 = Clock source from 48 MHz internal high speed RC oscillator (HIRC48M). 101 = Clock source from PLLFN/2. Others = Reserved.
[8]	**Reserved**	Reserved.
[7:6]	**KPISEL**	**KPI Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 10 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). 11 = Reserved.
[5:4]	**SC2SEL**	**SC2 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from PCLK0. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).
[3:2]	**SC1SEL**	**SC0 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from PCLK1. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).

[1:0]	**SC0SEL**	**SC0 Clock Source Selection** 00 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL/2. 10 = Clock source from PCLK0. 11 = Clock source from 12 MHz internal high speed RC oscillator (HIRC).

Clock Divider Number Register 0 (CLK_CLKDIV0)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV0	CLK_BA+0x20	R/W	Clock Divider Number Register 0	0x0000_0000

31	30	29	28	27	26	25	24
SDH0DIV							
23	22	21	20	19	18	17	16
EADC0DIV							
15	14	13	12	11	10	9	8
UART1DIV				UART0DIV			
7	6	5	4	3	2	1	0
USBDIV				HCLKDIV			

Bits	Description	
[31:24]	SDH0DIV	**SDH0 Clock Divide Number from SDH0 Clock Source** SDH0 clock frequency = (SDH0 clock source frequency) / (SDH0DIV + 1).
[23:16]	EADC0DIV	**EADC0 Clock Divide Number from EADC0 Clock Source** EADC0 clock frequency = (EADC0 clock source frequency) / (EADC0DIV + 1).
[15:12]	UART1DIV	**UART1 Clock Divide Number from UART1 Clock Source** UART1 clock frequency = (UART1 clock source frequency) / (UART1DIV + 1).
[11:8]	UART0DIV	**UART0 Clock Divide Number from UART0 Clock Source** UART0 clock frequency = (UART0 clock source frequency) / (UART0DIV + 1).
[7:4]	USBDIV	**USB Clock Divide Number from PLL/2 Clock** USB clock frequency = ((PLL frequency)/2) / (USBDIV + 1).
[3:0]	HCLKDIV	**HCLK Clock Divide Number from HCLK Clock Source** HCLK clock frequency = (HCLK clock source frequency) / (HCLKDIV + 1).

Clock Divider Number Register 1 (CLK_CLKDIV1)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV1	CLK_BA+0x24	R/W	Clock Divider Number Register 1	0x0000_0000

31	30	29	28	27	26	25	24
PSIODIV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SC2DIV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SC1DIV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SC0DIV							

Bits	Description	
[31:24]	**PSIODIV**	**PSIO Clock Divide Number from PSIO Clock Source** PSIO clock frequency = (PSIO clock source frequency) / (PSIODIV + 1).
[23:16]	**SC2DIV**	**SC2 Clock Divide Number from SC2 Clock Source** SC2 clock frequency = (SC2 clock source frequency) / (SC2DIV + 1).
[15:8]	**SC1DIV**	**SC1 Clock Divide Number from SC1 Clock Source** SC1 clock frequency = (SC1 clock source frequency) / (SC1DIV + 1).
[7:0]	**SC0DIV**	**SC0 Clock Divide Number from SC0 Clock Source** SC0 clock frequency = (SC0 clock source frequency) / (SC0DIV + 1).

Clock Divider Number Register 2 (CLK_CLKDIV2)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV2	CLK_BA+0x28	R/W	Clock Divider Number Register 2	0x0000_0000

31	30	29	28	27	26	25	24
EADC1DIV							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
KPIDIV							
7	6	5	4	3	2	1	0
I2S1DIV				I2S0DIV			

Bits	Description	
[31:24]	**EADC1DIV**	**EADC1 Clock Divide Number from EADC1 Clock Source** EADC1 clock frequency = (EADC1 clock source frequency) / (EADC1DIV + 1).
[23:16]	**Reserved**	Reserved.
[15:8]	**KPIDIV**	**KPI Clock Divide Number from KPI Clock Source** KPI clock frequency = (KPI clock source frequency) / (KPIDIV + 1).
[7:4]	**I2S1DIV**	**I2S1 Clock Divide Number from I2S1 Clock Source** I2S1 clock frequency = (I2S1 clock source frequency) / (I2S1DIV + 1).
[3:0]	**I2S0DIV**	**I2S0 Clock Divide Number from I2S0 Clock Source** I2S0 clock frequency = (I2S0 clock source frequency) / (I2S0DIV + 1).

Clock Divider Number Register 3 (CLK_CLKDIV3)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV3	CLK_BA+0x2C	R/W	Clock Divider Number Register 3	0x0000_0000

31	30	29	28	27	26	25	24
SDH1DIV							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
VSENSEDIV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:24]	**SDH1DIV**	**SDH1 Clock Divide Number from SDH1 Clock Source** SDH1 clock frequency = (SDH1 clock source frequency) / (SDH1DIV + 1).
[23:16]	**Reserved**	Reserved.
[15:8]	**VSENSEDIV**	**Video Pixel Clock Divide Number from CCAP Sensor Clock Source** Video pixel clock frequency = (CCAP sensor clock source frequency) / (VSENSEDIV + 1).
[7:0]	**Reserved**	Reserved.

Clock Divider Number Register 4 (CLK_CLKDIV4)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV4	CLK_BA+0x30	R/W	Clock Divider Number Register 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
UART7DIV				UART6DIV			
15	14	13	12	11	10	9	8
UART5DIV				UART4DIV			
7	6	5	4	3	2	1	0
UART3DIV				UART2DIV			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**UART7DIV**	**UART7 Clock Divide Number from UART7 Clock Source** UART7 clock frequency = (UART7 clock source frequency) / (UART7DIV + 1).
[19:16]	**UART6DIV**	**UART6 Clock Divide Number from UART6 Clock Source** UART6 clock frequency = (UART6 clock source frequency) / (UART6DIV + 1).
[15:12]	**UART5DIV**	**UART5 Clock Divide Number from UART5 Clock Source** UART5 clock frequency = (UART5 clock source frequency) / (UART5DIV + 1).
[11:8]	**UART4DIV**	**UART4 Clock Divide Number from UART4 Clock Source** UART4 clock frequency = (UART4 clock source frequency) / (UART4DIV + 1).
[7:4]	**UART3DIV**	**UART3 Clock Divide Number from UART3 Clock Source** UART3 clock frequency = (UART3 clock source frequency) / (UART3DIV + 1).
[3:0]	**UART2DIV**	**UART2 Clock Divide Number from UART2 Clock Source** UART2 clock frequency = (UART2 clock source frequency) / (UART2DIV + 1).

APB Clock Divider Register (CLK_PCLKDIV)

Register	Offset	R/W	Description	Reset Value
CLK_PCLKDIV	CLK_BA+0x34	R/W	APB Clock Divider Register	0x0000_0011

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**APB1DIV**			**Reserved**	**APB0DIV**		

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:4]	**APB1DIV**	**APB1 Clock Divider** APB1 clock can be divided from HCLK. 000 = PCLK1 frequency is HCLK. 001 = PCLK1 frequency is HCLK/2. 010 = PCLK1 frequency is HCLK/4. 011 = PCLK1 frequency is HCLK/8. 100 = PCLK1 frequency is HCLK/16. Others = Reserved.
[3]	**Reserved**	Reserved.
[2:0]	**APB0DIV**	**APB0 Clock Divider** APB0 clock can be divided from HCLK. 000 = PCLK0 frequency is HCLK. 001 = PCLK0 frequency is HCLK/2. 010 = PCLK0 frequency is HCLK/4. 011 = PCLK0 frequency is HCLK/8. 100 = PCLK0 frequency is HCLK/16. Others = Reserved.

APB Devices Clock Enable Control Register 2 (CLK_APBCLK2)

The bits in this register are used to enable/disable clock for peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
CLK_APBCLK2	CLK_BA+0x38	R/W	APB Devices Clock Enable Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						UART9CKEN	UART8CKEN
15	14	13	12	11	10	9	8
Reserved		SPI10CKEN	SPI9CKEN	SPI8CKEN	SPI7CKEN	SPI6CKEN	SPI5CKEN
7	6	5	4	3	2	1	0
ACMP23CKEN	EADC2CKEN	Reserved					KPICKEN

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**UART9CKEN**	**UART9 Clock Enable Bit** 0 = UART9 clock Disabled. 1 = UART9 clock Enabled.
[16]	**UART8CKEN**	**UART8 Clock Enable Bit** 0 = UART8 clock Disabled. 1 = UART8 clock Enabled.
[15:14]	**Reserved**	Reserved.
[13]	**SPI10CKEN**	**SPI10 Clock Enable Bit** 0 = SPI10 clock Disabled. 1 = SPI10 clock Enabled.
[12]	**SPI9CKEN**	**SPI9 Clock Enable Bit** 0 = SPI9 clock Disabled. 1 = SPI9 clock Enabled.
[11]	**SPI8CKEN**	**SPI8 Clock Enable Bit** 0 = SPI8 clock Disabled. 1 = SPI8 clock Enabled.
[10]	**SPI7CKEN**	**SPI7 Clock Enable Bit** 0 = SPI7 clock Disabled. 1 = SPI7 clock Enabled.
[9]	**SPI6CKEN**	**SPI6 Clock Enable Bit** 0 = SPI6 clock Disabled. 1 = SPI6 clock Enabled.
[8]	**SPI5CKEN**	**SPI5 Clock Enable Bit**

		0 = SPI5 clock Disabled. 1 = SPI5 clock Enabled.
[7]	**ACMP23CKEN**	**Analog Comparator 2/3 Clock Enable Bit** 0 = Analog comparator 2/3 clock Disabled. 1 = Analog comparator 2/3 clock Enabled.
[6]	**EADC2CKEN**	**EADC2 Clock Enable Bit** 0 = EADC2 clock Disabled. 1 = EADC2 clock Enabled.
[5:1]	**Reserved**	Reserved.
[0]	**KPICKEN**	**KPI Clock Enable Bit** 0 = KPI clock Disabled. 1 = KPI clock Enabled.

Clock Divider Number Register 5 (CLK_CLKDIV5)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV5	CLK_BA+0x3C	R/W	Clock Divider Number Register 5	0x0000_0000

31	30	29	28	27	26	25	24
EADC2DIV							
23	22	21	20	19	18	17	16
UART9DIV				UART8DIV			
15	14	13	12	11	10	9	8
CANFD3DIV				CANFD2DIV			
7	6	5	4	3	2	1	0
CANFD1DIV				CANFD0DIV			

Bits	Description	
[31:24]	EADC2DIV	**EADC2 Clock Divide Number from EADC2 Clock Source** EADC2 clock frequency = (EADC2 clock source frequency) / (EADC2DIV + 1).
[23:20]	UART9DIV	**UART9 Clock Divide Number from UART9 Clock Source** UART9 clock frequency = (UART9 clock source frequency) / (UART9DIV + 1).
[19:16]	UART8DIV	**UART8 Clock Divide Number from UART8 Clock Source** UART8 clock frequency = (UART8 clock source frequency) / (UART8DIV + 1).
[15:12]	CANFD3DIV	**CANFD3 Clock Divide Number from CANFD3 Clock Source** CANFD3 clock frequency = (CANFD3 clock source frequency) / (CANFD3DIV + 1).
[11:8]	CANFD2DIV	**CANFD2 Clock Divide Number from CANFD2 Clock Source** CANFD2 clock frequency = (CANFD2 clock source frequency) / (CANFD2DIV + 1).
[7:4]	CANFD1DIV	**CANFD1 Clock Divide Number from CANFD1 Clock Source** CANFD1 clock frequency = (CANFD1 clock source frequency) / (CANFD1DIV + 1).
[3:0]	CANFD0DIV	**CANFD0 Clock Divide Number from CANFD0 Clock Source** CANFD0 clock frequency = (CANFD0 clock source frequency) / (CANFD0DIV + 1).

PLL Control Register (CLK_PLLCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PLLCTL	CLK_BA+0x40	R/W	PLL Control Register	0x0008_421E

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
STBSEL	**Reserved**			**PLLSRC**	**OE**	**BP**	**PD**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
OUTDIV		**INDIV**					**FBDIV**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FBDIV							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**STBSEL**	**PLL Stable Counter Selection (Write Protect)** 0 = PLL stable time is 1200 PLL source clock (suitable for source clock equal to or less than 12 MHz). 1 = PLL stable time is 2400 PLL source clock (suitable for source clock larger than 12 MHz). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[22:20]	**Reserved**	Reserved.
[19]	**PLLSRC**	**PLL Source Clock Selection (Write Protect)** 0 = PLL source clock from 4~24 MHz external high-speed crystal oscillator (HXT). 1 = PLL source clock from 12 MHz internal high-speed oscillator (HIRC). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[18]	**OE**	**PLL FOUT Enable Control (Write Protect)** 0 = PLL FOUT Enabled. 1 = PLL FOUT is fixed low. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[17]	**BP**	**PLL Bypass Control (Write Protect)** 0 = PLL is in normal mode (default). 1 = PLL clock output is same as PLL input clock FIN. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[16]	**PD**	**Power-down Mode (Write Protect)** If set the PDEN bit to 1 in CLK_PWRCTL register, the PLL will enter Power-down mode, too. 0 = PLL is in normal mode. 1 = PLL is in Power-down mode (default). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[15:14]	**OUTDIV**	**PLL Output Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[13:9]	**INDIV**	**PLL Input Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[8:0]	**FBDIV**	**PLL Feedback Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Output Clock Frequency formula:

$$FREF = FIN \times \frac{1}{NR}$$

$$FVCO = FIN \times \frac{2 * NF}{NR}$$

$$FOUT = FIN \times \frac{2 * NF}{NR} \times \frac{1}{NO}$$

where FREF is the comparison frequency for the PFD (phase frequency detector).

For proper operation in normal mode, the following constraints must be satisfied:

4 MHz ≤ FREF ≤ 8 MHz

200 MHz ≤ FVCO ≤ 500 MHz

50 MHz ≤ FOUT ≤ 500 MHz

Symbol	Description
FOUT	Output Clock Frequency
FIN	Input (Reference) Clock Frequency
NR	Input Divider (INDIV + 1)
NF	Feedback Divider (FBDIV + 2)
NO	OUTDIV = “00” : NO = 1 OUTDIV = “01” : NO = 2 OUTDIV = “10” : NO = 2 OUTDIV = “11” : NO = 4

Table 6.3-3 Symbol definition of PLL Output Frequency formula

Default Frequency Setting

The default value: CLK_PLLCTL[15:0] = 0x421E

FIN = 12 MHz

NR = (1+1) = 2

NF = (30+2) = 32

NO = 2

FOUT = (12 MHz/2) x 2(32) x (1/2) = 192 MHz

PLLFN Control Register 0 (CLK_PLLFNCTL0)

Register	Offset	R/W	Description	Reset Value
CLK_PLLFNCTL0	CLK_BA+0x48	R/W	PLLFN Control Register 0	0x0000_842B

31	30	29	28	27	26	25	24
Reserved				FRDIV			
23	22	21	20	19	18	17	16
FRDIV							
15	14	13	12	11	10	9	8
OUTDIV		INDIV					FBDIV
7	6	5	4	3	2	1	0
FBDIV							

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:16]	**FRDIV**	**PLL Fractional Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15:14]	**OUTDIV**	**PLL Output Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[13:9]	**INDIV**	**PLL Input Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[8:0]	**FBDIV**	**PLL Feedback Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Output Clock Frequency formula:

$$FREF = FIN \times \frac{1}{NR}$$

$$FVCO = FIN \times \frac{2 * NF.x}{NR}$$

$$FOUT = FIN \times \frac{2 * NF.x}{NR} \times \frac{1}{NO}$$

where FREF is the comparison frequency for the PFD (phase frequency detector).

For proper operation in normal mode, the following constraints must be satisfied:

1 MHz ≤ FREF ≤ 8 MHz

200 MHz ≤ FVCO ≤ 500 MHz

50 MHz ≤ FOUT ≤ 500 MHz

12≤ NF ≤255

Symbol	Description
FOUT	Output Clock Frequency
FIN	Input (Reference) Clock Frequency
NR	Input Divider (INDIV + 1)
NF	Feedback Divider (FBDIV + 2)
x	Fractional Value FRDIV/4096
NO	OUTDIV = “00” : NO = 1 OUTDIV = “01” : NO = 2 OUTDIV = “10” : NO = 2 OUTDIV = “11” : NO = 4

Default Frequency Setting

The default value: CLK_PLLFCTL[15:0] = 0x842B

FIN = 12 MHz

NR = (2+1) = 3

NF = (43+2) = 45

NO = 2

FOUT = (12 MHz/3) x 2(45) x (1/2) = 180 MHz

PLLFN Control Register 1 (CLK_PLLFNCTL1)

Register	Offset	R/W	Description	Reset Value
CLK_PLLFNCTL1	CLK_BA+0x4C	R/W	PLLFN Control Register 1	0x5000_0000

31	30	29	28	27	26	25	24
PLLSRC	OE	BP	PD	STBSEL	Reserved		
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31]	**PLLSRC**	**PLL Source Clock Selection (Write Protect)** 0 = PLL source clock from 4~32 MHz external high-speed crystal oscillator (HXT). 1 = PLL source clock from 12 MHz internal high-speed oscillator (HIRC). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	**OE**	**PLL FOUT Enable Control (Write Protect)** 0 = PLL FOUT Enabled. 1 = PLL FOUT is fixed low. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[29]	**BP**	**PLL Bypass Control (Write Protect)** 0 = PLL is in normal mode (default). 1 = PLL clock output is same as PLL input clock FIN. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[28]	**PD**	**Power-down Mode (Write Protect)** If set the PDEN bit to 1 in CLK_PWRCTL register, the PLL will enter Power-down mode, too. 0 = PLL is in normal mode. 1 = PLL is in Power-down mode (default). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[27]	**STBSEL**	**PLL Stable Counter Selection (Write Protect)** 0 = PLL stable time is 1200 PLL source clock (suitable for source clock equal to or less than 12 MHz). 1 = PLL stable time is 2400 PLL source clock (suitable for source clock larger than 12 MHz). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[26:0]	**Reserved**	Reserved.

Clock Status Monitor Register (CLK_STATUS)

The bits in this register are used to monitor if the chip clock source is stable or not, and whether the clock switch is failed.

Register	Offset	R/W	Description	Reset Value
CLK_STATUS	CLK_BA+0x50	R	Clock Status Monitor Register	0x0000_0XXX

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					PLLFNSTB	Reserved	
7	6	5	4	3	2	1	0
CLKSFAIL	HIRC48MSTB	Reserved	HIRCSTB	LIRCSTB	PLLSTB	LXTSTB	HXTSTB

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**PLLFNSTB**	**Internal PLLFN Clock Source Stable Flag (Read Only)** 0 = Internal PLLFN clock is not stable or disabled. 1 = Internal PLLFN clock is stable.
[9:8]	**Reserved**	Reserved.
[7]	**CLKSFAIL**	**Clock Switching Fail Flag (Read Only)** This bit is updated when software switches system clock source. If switch target clock is stable, this bit will be set to 0. If switch target clock is not stable, this bit will be set to 1. 0 = Clock switching success. 1 = Clock switching failure. **Note:** This bit is read only. After selected clock source is stable, hardware will switch system clock to selected clock automatically, and CLKSFAIL will be cleared automatically by hardware.
[6]	**HIRC48MSTB**	**HIRC48M Clock Source Stable Flag (Read Only)** 0 = 48 MHz internal high speed RC oscillator (HIRC48M) clock is not stable or disabled. 1 = 48 MHz internal high speed RC oscillator (HIRC48M) clock is stable and enabled.
[5]	**Reserved**	Reserved.
[4]	**HIRCSTB**	**HIRC Clock Source Stable Flag (Read Only)** 0 = 12 MHz internal high speed RC oscillator (HIRC) clock is not stable or disabled. 1 = 12 MHz internal high speed RC oscillator (HIRC) clock is stable and enabled.
[3]	**LIRCSTB**	**LIRC Clock Source Stable Flag (Read Only)** 0 = 10 kHz internal low speed RC oscillator (LIRC) clock is not stable or disabled. 1 = 10 kHz internal low speed RC oscillator (LIRC) clock is stable and enabled.
[2]	**PLLSTB**	**Internal PLL Clock Source Stable Flag (Read Only)**

		0 = Internal PLL clock is not stable or disabled. 1 = Internal PLL clock is stable and enabled.
[1]	**LXTSTB**	**LXT Clock Source Stable Flag (Read Only)** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock is not stable or disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock is stabled and enabled.
[0]	**HXTSTB**	**HXT Clock Source Stable Flag (Read Only)** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock is not stable or disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock is stable and enabled.

AHB Devices Clock Enable Control Register 1 (CLK_AHBCLK1)

The bits in this register are used to enable/disable clock for AHB bus peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
CLK_AHBCLK1	CLK_BA+0x58	R/W	AHB Devices Clock Enable Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						GPJCKEN	GPICKEN
23	22	21	20	19	18	17	16
CANFD3CKEN	CANFD2CKEN	CANFD1CKEN	CANFD0CKEN	Reserved			
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**GPJCKEN**	**GPIOJ Clock Enable Bit** 0 = GPIOJ clock Disabled. 1 = GPIOJ clock Enabled.
[24]	**GPICKEN**	**GPIOI Clock Enable Bit** 0 = GPIOI clock Disabled. 1 = GPIOI clock Enabled.
[23]	**CANFD3CKEN**	**CANFD3 Clock Enable Bit** 0 = CANFD3 clock Disabled. 1 = CANFD3 clock Enabled.
[22]	**CANFD2CKEN**	**CANFD2 Clock Enable Bit** 0 = CANFD2 clock Disabled. 1 = CANFD2 clock Enabled.
[21]	**CANFD1CKEN**	**CANFD1 Clock Enable Bit** 0 = CANFD1 clock Disabled. 1 = CANFD1 clock Enabled.
[20]	**CANFD0CKEN**	**CANFD0 Clock Enable Bit** 0 = CANFD0 clock Disabled. 1 = CANFD0 clock Enabled.
[19:0]	**Reserved**	Reserved.

Clock Source Select Control Register 4 (CLK_CLKSEL4)

Before clock switching, the related clock sources (pre-select and new-select) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL4	CLK_BA+0x5C	R/W	Clock Source Select Control Register 4	0x0222_2222

31	30	29	28	27	26	25	24
Reserved					SPI10SEL		
23	22	21	20	19	18	17	16
Reserved	SPI9SEL			Reserved	SPI8SEL		
15	14	13	12	11	10	9	8
Reserved	SPI7SEL			Reserved	SPI6SEL		
7	6	5	4	3	2	1	0
Reserved	SPI5SEL			Reserved	SPI4SEL		

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:24]	**SPI10SEL**	**SPI10 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK1. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[23]	**Reserved**	Reserved.
[22:20]	**SPI9SEL**	**SPI9 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK0. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[19]	**Reserved**	Reserved.
[18:16]	**SPI8SEL**	**SPI8 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK1. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[15]	**Reserved**	Reserved.
[14:12]	**SPI7SEL**	**SPI7 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2.

		010 = Clock source from PCLK0. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[11]	**Reserved**	Reserved.
[10:8]	**SPI6SEL**	**SPI6 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK1. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[7]	**Reserved**	Reserved.
[6:4]	**SPI5SEL**	**SPI5 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK0. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[3]	**Reserved**	Reserved.
[2:0]	**SPI4SEL**	**SPI4 Clock Source Selection** 000 = Clock source from 4~24 MHz external high speed crystal oscillator (HXT). 001 = Clock source from PLL/2. 010 = Clock source from PCLK1. 011 = Clock source from 12 MHz internal high speed RC oscillator (HIRC). Others = Reserved.

Clock Output Control Register (CLK_CLKOCTL)

Register	Offset	R/W	Description	Reset Value
CLK_CLKOCTL	CLK_BA+0x60	R/W	Clock Output Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	CLK1HZEN	DIV1EN	CLKOEN	FREQSEL			

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**CLK1HZEN**	**Clock Output 1Hz Enable Bit** 0 = 1 Hz clock output for 32.768 kHz frequency compensation Disabled. 1 = 1 Hz clock output for 32.768 kHz frequency compensation Enabled.
[5]	**DIV1EN**	**Clock Output Divide One Enable Bit** 0 = Clock Output will output clock with source frequency divided by FREQSEL. 1 = Clock Output will output clock with source frequency.
[4]	**CLKOEN**	**Clock Output Enable Bit** 0 = Clock Output function Disabled. 1 = Clock Output function Enabled.
[3:0]	**FREQSEL**	**Clock Output Frequency Selection** The formula of output frequency is $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$. F_{in} is the input clock frequency. F_{out} is the frequency of divider output clock. N is the 4-bit value of FREQSEL[3:0].

Clock Fail Detector Control Register (CLK_CLKDCTL)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDCTL	CLK_BA+0x70	R/W	Clock Fail Detector Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					HXTFQASW	HXTFQIEN	HXTFQDEN
15	14	13	12	11	10	9	8
Reserved		LXTFIEN	LXTFDEN	Reserved			
7	6	5	4	3	2	1	0
Reserved		HXTFIEN	HXTFDEN	Reserved			

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**HXTFQASW**	**HXT Clock Frequency Range Detector Event Auto Switch Enable Bit** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency range detector fail event happened and HCLK will not switch to HIRC automatically. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency range detector fail event happened and HCLK will switch to HIRC automatically. **Note:** This bit should be set before HXTFQDEN(CLK_CLKDCTL[16]).
[17]	**HXTFQIEN**	**HXT Clock Frequency Range Detector Interrupt Enable Bit** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency range detector fail interrupt Disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency range detector fail interrupt Enabled.
[16]	**HXTFQDEN**	**HXT Clock Frequency Range Detector Enable Bit** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency range detector Disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency range detector Enabled.
[15:14]	**Reserved**	Reserved.
[13]	**LXTFIEN**	**LXT Clock Fail Interrupt Enable Bit** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail interrupt Disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail interrupt Enabled.
[12]	**LXTFDEN**	**LXT Clock Fail Detector Enable Bit** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail detector Disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail detector Enabled.
[11:6]	**Reserved**	Reserved.
[5]	**HXTFIEN**	**HXT Clock Fail Interrupt Enable Bit** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail interrupt Disabled.

		1 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail interrupt Enabled.
[4]	**HXTFDEN**	**HXT Clock Fail Detector Enable Bit** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail detector Disabled. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock fail detector Enabled.
[3:0]	**Reserved**	Reserved.

Clock Fail Detector Status Register (CLK_CLKDSTS)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDSTS	CLK_BA+0x74	R/W	Clock Fail Detector Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							HXTFQIF
7	6	5	4	3	2	1	0
Reserved						LXTFIF	HXTFIF

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**HXTFQIF**	**HXT Clock Frequency Range Detector Interrupt Flag (Write Protect)** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency is normal. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock frequency is abnormal. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7:2]	**Reserved**	Reserved.
[1]	**LXTFIF**	**LXT Clock Fail Interrupt Flag (Write Protect)** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock is normal. 1 = 32.768 kHz external low speed crystal oscillator (LXT) stops. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**HXTFIF**	**HXT Clock Fail Interrupt Flag (Write Protect)** 0 = 4~24 MHz external high speed crystal oscillator (HXT) clock is normal. 1 = 4~24 MHz external high speed crystal oscillator (HXT) clock stops. **Note 1:** Write 1 to clear the bit to 0. **Note 2:** This bit is write protected. Refer to the SYS_REGLCTL register.

Clock Frequency Range Detector Upper Boundary Register (CLK_CDUPB)

Register	Offset	R/W	Description	Reset Value
CLK_CDUPB	CLK_BA+0x78	R/W	Clock Frequency Range Detector Upper Boundary Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						UPERBD	
7	6	5	4	3	2	1	0
UPERBD							

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**UPERBD**	**HXT Clock Frequency Range Detector Upper Boundary Value** The bits define the maximum value of frequency range detector window. When HXT frequency higher than this maximum frequency value, the HXT Clock Frequency Range Detector Interrupt Flag will set to 1.

Clock Frequency Range Detector Lower Boundary Register (CLK_CDLOWB)

Register	Offset	R/W	Description	Reset Value
CLK_CDLOWB	CLK_BA+0x7C	R/W	Clock Frequency Range Detector Lower Boundary Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**LOWERBD**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LOWERBD							

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**LOWERBD**	**HXT Clock Frequency Range Detector Lower Boundary Value** The bits define the minimum value of frequency range detector window. When HXT frequency lower than this minimum frequency value, the HXT Clock Frequency Range Detector Interrupt Flag will set to 1.

Clock Stop Request Register (CLK_STOPREQ)

Register	Offset	R/W	Description	Reset Value
CLK_STOPREQ	CLK_BA+0x80	R/W	Clock Stop Request Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CANFD3STR	CANFD2STR	CANFD1STR	CANFD0STR

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CANFD3STR**	**CANFD3 Clock Stop Request** This bit is used to stop CANFD3 clock. 0 = CANFD3 clock is not stopped by this bit. (default) 1 = Set this bit and check the CANFD3STA(CLK_STOPACK[3]) is 1, then CANFD3 clock stop.
[2]	**CANFD2STR**	**CANFD2 Clock Stop Request** This bit is used to stop CANFD2 clock. 0 = CANFD2 clock is not stopped by this bit. (default) 1 = Set this bit and check the CANFD2STA(CLK_STOPACK[2]) is 1, then CANFD2 clock stop.
[1]	**CANFD1STR**	**CANFD1 Clock Stop Request** This bit is used to stop CANFD1 clock. 0 = CANFD1 clock is not stopped by this bit. (default) 1 = Set this bit and check the CANFD1STA(CLK_STOPACK[1]) is 1, then CANFD1 clock stop.
[0]	**CANFD0STR**	**CANFD0 Clock Stop Request** This bit is used to stop CANFD0 clock. 0 = CANFD0 clock is not stopped by this bit. (default) 1 = Set this bit and check the CANFD0STA(CLK_STOPACK[0]) is 1, then CANFD0 clock stop.

Clock Stop Acknowledge Register (CLK_STOPACK)

Register	Offset	R/W	Description	Reset Value
CLK_STOPACK	CLK_BA+0x84	R	Clock Stop Acknowledge Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**CANFD3STA**	**CANFD2STA**	**CANFD1STA**	**CANFD0STA**

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CANFD3STA**	**CANFD3 Clock Stop Acknowledge (Read Only)** This bit is used to check CANFD3 clock stop by setting CANFD3STR(CLK_STOPREQ[3]). 0 = CANFD3 clock not stopped. 1 = CANFD3 clock stopped.
[2]	**CANFD2STA**	**CANFD2 Clock Stop Acknowledge (Read Only)** This bit is used to check CANFD2 clock stop by setting CANFD2STR(CLK_STOPREQ[2]). 0 = CANFD2 clock not stopped. 1 = CANFD2 clock stopped.
[1]	**CANFD1STA**	**CANFD1 Clock Stop Acknowledge (Read Only)** This bit is used to check CANFD1 clock stop by setting CANFD1STR(CLK_STOPREQ[1]). 0 = CANFD1 clock not stopped. 1 = CANFD1 clock stopped.
[0]	**CANFD0STA**	**CANFD0 Clock Stop Acknowledge (Read Only)** This bit is used to check CANFD0 clock stop by setting CANFD0STR(CLK_STOPREQ[0]). 0 = CANFD0 clock not stopped. 1 = CANFD0 clock stopped.

Power Manager Control Register (CLK_PMUCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PMUCTL	CLK_BA+0x90	R/W	Power Manager Control Register	0x0000_0040

31	30	29	28	27	26	25	24
WKPINEN4		WKPINEN3		WKPINEN2		WKPINEN1	
23	22	21	20	19	18	17	16
RTCWKEN	VBUSWKEN	Reserved			ACMPSPWK	WKPINEN0	
15	14	13	12	11	10	9	8
Reserved			WKTMRIS				WKTMREN
7	6	5	4	3	2	1	0
Reserved	SRETSEL			DPDHOLDEN	PDMSEL		

Bits	Description	
[31:30]	**WKPINEN4**	**Wake-up Pin4 Enable Bit (Write Protect)** This is control register for GPF.6 to wake-up pin. 00 = Wake-up pin disabled at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note 1:** These bits are write protected. Refer to the SYS_REGLCTL register. **Note 2:** To use GPF.6 wake-up pin function at Deep Power-down mode, user has to set IOCTLSEL(RTC_LXTCTL[8])=1 to control GPF.6 in VBAT power domain by RTC_GPIOCTL0. Otherwise, GPF.6 will be digital off (digital input tied to low) in DPD mode and cause unexpected falling edge to wake-up system.
[29:28]	**WKPINEN3**	**Wake-up Pin3 Enable Bit (Write Protect)** This is control register for GPB.12 to wake-up pin. 00 = Wake-up pin disabled at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[27:26]	**WKPINEN2**	**Wake-up Pin2 Enable Bit (Write Protect)** This is control register for GPB.2 to wake-up pin. 00 = Wake-up pin disabled at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[25:24]	**WKPINEN1**	**Wake-up Pin1 Enable Bit (Write Protect)** This is control register for GPB.0 to wake-up pin. 00 = Wake-up pin disabled at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode.

		10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[23]	**RTCWKEN**	**RTC Wake-up Enable Bit (Write Protect)** 0 = RTC wake-up disabled at Deep Power-down mode or Standby Power-down mode. 1 = RTC wake-up enabled at Deep Power-down mode or Standby Power-down mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[22]	**VBUSWKEN**	**V_{BUS} Wake-up Enable Bit (Write Protect)** 0 = V_{BUS} transition wake-up disabled at Deep Power-down mode. 1 = V_{BUS} transition wake-up enabled at Deep Power-down mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[21:19]	**Reserved**	Reserved.
[18]	**ACMPSPWK**	**ACMP Standby Power-down Mode Wake-up Enable Bit (Write Protect)** 0 = ACMP wake-up disabled at Standby Power-down mode. 1 = ACMP wake-up enabled at Standby Power-down mode. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** Set FILTSEL(ACMP_CTLx[15:13]) for comparator output filter count selection, the filter clock is LIRC in ACMP SPD mode wakeup function.
[17:16]	**WKPINEN0**	**Wake-up Pin0 Enable Bit (Write Protect)** This is control register for GPC.0 to wake-up pin. 00 = Wake-up pin disabled at Deep Power-down mode. 01 = Wake-up pin rising edge enabled at Deep Power-down mode. 10 = Wake-up pin falling edge enabled at Deep Power-down mode. 11 = Wake-up pin both edge enabled at Deep Power-down mode. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[15:13]	**Reserved**	Reserved.
[12:9]	**WKTMRIS**	**Wake-up Timer Time-out Interval Select (Write Protect)** These bits control wake-up timer time-out interval when chip at DPD/SPD mode. 0000 = Time-out interval is 128 LIRC clocks (12.8ms). 0001 = Time-out interval is 256 LIRC clocks (25.6ms). 0010 = Time-out interval is 512 LIRC clocks (51.2ms). 0011 = Time-out interval is 1024 LIRC clocks (102.4ms). 0100 = Time-out interval is 4096 LIRC clocks (409.6ms). 0101 = Time-out interval is 8192 LIRC clocks (819.2ms). 0110 = Time-out interval is 16384 LIRC clocks (1638.4ms). 0111 = Time-out interval is 65536 LIRC clocks (6553.6ms). 1000 = Time-out interval is 131072 LIRC clocks (13107.2ms). 1001 = Time-out interval is 262144 LIRC clocks (26214.4ms). 1010 = Time-out interval is 524288 LIRC clocks (52428.8ms). 1011 = Time-out interval is 1048576 LIRC clocks (104857.6ms). Others = Time-out interval is 128 LIRC clocks (12.8ms). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[8]	**WKTMREN**	**Wake-up Timer Enable Bit (Write Protect)** 0 = Wake-up timer disabled at DPD/SPD mode. 1 = Wake-up timer enabled at DPD/SPD mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[7]	**Reserved**	Reserved.
[6:4]	**SRETSEL**	**SRAM Retention Range Select Bit (Write Protect)** Select SRAM retention range when chip enter SPD mode. 000 = No SRAM retention. 001 = 16K SRAM retention when chip enter SPD mode. 010 = 32K SRAM retention when chip enter SPD mode. 011 = 64K SRAM retention when chip enter SPD mode. 100 = 128K SRAM retention when chip enter SPD mode. (default) 101 = 256K SRAM retention when chip enter SPD mode. Others = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[3]	**DPDHOLDEN**	**Deep-Power-Down Mode GPIO Hold Enable Bit (Write Protect)** 0= When GPIO enters deep power-down mode, all I/O status are tri-state. 1= When GPIO enters deep power-down mode, all I/O status are hold to keep normal operating status. After chip is woken up from deep power-down mode, the I/O are still keep hold status until user set CLK_IOPDCTL[0] to release I/O hold status. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2:0]	**PDMSEL**	**Power-down Mode Selection (Write Protect)** These bits control chip power-down mode grade selection when CPU execute WFI/WFE instruction. 000 = Normal Power-down mode is selected (NPD). 001 = Low leakage Power-down mode is selected (LLPD). 010 = Fast wake-up Power-down mode is selected (FWPD). 011 = Reserved. 100 = Standby Power-down mode is selected (SPD). 101 = Reserved. 110 = Deep Power-down mode is selected (DPD). 111 = Reserved. **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Power Manager Status Register (CLK_PMUSTS)

Register	Offset	R/W	Description	Reset Value
CLK_PMUSTS	CLK_BA+0x94	R/W	Power Manager Status Register	0x0000_0000

31	30	29	28	27	26	25	24
CLRWK	Reserved						
23	22	21	20	19	18	17	16
Reserved				ACMPWK3	ACMPWK2	ACMPWK1	ACMPWK0
15	14	13	12	11	10	9	8
RSTWK	Reserved	BODWK	LVRWK	GPDWK	GPCWK	GPBWK	GPAWK
7	6	5	4	3	2	1	0
VBUSWK	PINWK4	PINWK3	PINWK2	PINWK1	RTCWK	TMRWK	PINWK0

Bits	Description	
[31]	**CLRWK**	**Clear Wake-up Flag** 0 = No clear. 1 = Clear all wake-up flag. **Note:** This bit is auto cleared by hardware.
[30:20]	**Reserved**	Reserved.
[19]	**ACMPWK3**	**ACMP3 Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with an ACMP3 transition. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[18]	**ACMPWK2**	**ACMP2 Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with an ACMP2 transition. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[17]	**ACMPWK1**	**ACMP1 Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with an ACMP1 transition. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[16]	**ACMPWK0**	**ACMP0 Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with an ACMP0 transition. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[15]	**RSTWK**	**RST pin Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Deep Power-down mode (DPD) or Standby Power-down (SPD) mode was requested with a RST pin trigger happened. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering

		SPD/DPD mode.
[14]	**Reserved**	Reserved.
[13]	**BODWK**	**BOD Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode (SPD) was requested with a BOD happened. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[12]	**LVRWK**	**LVR Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Standby Power-down mode was requested with a LVR happened. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[11]	**GPDWK**	**GPD Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Standby Power-down mode was requested by a transition of selected one GPD group pins. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[10]	**GPCWK**	**GPC Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Standby Power-down mode was requested by a transition of selected one GPC group pins. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[9]	**GPBWK**	**GPB Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Standby Power-down mode was requested by a transition of selected one GPB group pins. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[8]	**GPAWK**	**GPA Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Standby Power-down mode was requested by a transition of selected one GPA group pins. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD mode.
[7]	**VBUSWK**	**V_{BUS} Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode was requested by a transition of the WAKEUP pin (PA.12). **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering DPD mode.
[6]	**PINWK4**	**Pin4 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode was requested by a transition of the WAKEUP pin (PF.6). **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering DPD mode.
[5]	**PINWK3**	**Pin3 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode was requested by a transition of the WAKEUP pin (PB.12). **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering DPD mode.
[4]	**PINWK2**	**Pin2 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode was requested by a transition of the WAKEUP pin (PB.2). **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering

		DPD mode.
[3]	**PINWK1**	**Pin1 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode was requested by a transition of the WAKEUP pin (PB.0). **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering DPD mode.
[2]	**RTCWK**	**RTC Wake-up Flag (Read Only)** This flag indicates that wakeup of device from Deep Power-down mode (DPD) or Standby Power-down (SPD) mode was requested with a RTC alarm, tick time or tamper happened. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD/DPD mode.
[1]	**TMRWK**	**Timer Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode (DPD) or Standby Power-down (SPD) mode was requested by wakeup timer time-out. **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering SPD/DPD mode.
[0]	**PINWK0**	**Pin0 Wake-up Flag (Read Only)** This flag indicates that wake-up of chip from Deep Power-down mode was requested by a transition of the WAKEUP pin (GPC.0). **Note:** This flag needs to be cleared by setting CLRWK(CLK_PMUSTS[31] when entering DPD mode.

GPIO Standby Power-down Wake-up De-bounce Control Register (CLK_SWKDBCTL)

Register	Offset	R/W	Description	Reset Value
CLK_SWKDBCTL	CLK_BA+0x9C	R/W	GPIO Standby Power-down Wake-up De-bounce Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				SWKDBCLKSEL			

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**SWKDBCLKSEL**	**Standby Power-down Wake-up De-bounce Sampling Cycle Selection** 0000 = Sample wake-up input once per 1 clock. 0001 = Sample wake-up input once per 2 clocks. 0010 = Sample wake-up input once per 4 clocks. 0011 = Sample wake-up input once per 8 clocks. 0100 = Sample wake-up input once per 16 clocks. 0101 = Sample wake-up input once per 32 clocks. 0110 = Sample wake-up input once per 64 clocks. 0111 = Sample wake-up input once per 128 clocks. 1000 = Sample wake-up input once per 256 clocks. 1001 = Sample wake-up input once per 2*256 clocks. 1010 = Sample wake-up input once per 4*256 clocks. 1011 = Sample wake-up input once per 8*256 clocks. 1100 = Sample wake-up input once per 16*256 clocks. 1101 = Sample wake-up input once per 32*256 clocks. 1110 = Sample wake-up input once per 64*256 clocks. 1111 = Sample wake-up input once per 128*256 clocks. **Note:** De-bounce counter clock source is the 10 kHz internal low speed RC oscillator (LIRC).

GPA Standby Power-down Wake-up Control Register (CLK_PASWKCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PASWKCTL	CLK_BA+0xA0	R/W	GPA Standby Power-down Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DBEN
7	6	5	4	3	2	1	0
WKPSEL				Reserved	PFWKEN	PRWKEN	WKEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DBEN**	**GPA Input Signal De-bounce Enable Bit** The DBEN bit is used to enable the de-bounce function for each corresponding I/O. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the wake-up. The de-bounce clock source is the 10 kHz internal low speed RC oscillator (LIRC). 0 = Standby power-down wake-up pin De-bounce function Disabled. 1 = Standby power-down wake-up pin De-bounce function Enabled. The de-bounce function is valid only for edge triggered.
[7:4]	**WKPSEL**	**GPA Standby Power-down Wake-up Pin Select** 0000 = GPA.0 wake-up function Enabled. 0001 = GPA.1 wake-up function Enabled. 0010 = GPA.2 wake-up function Enabled. 0011 = GPA.3 wake-up function Enabled. 0100 = GPA.4 wake-up function Enabled. 0101 = GPA.5 wake-up function Enabled. 0110 = GPA.6 wake-up function Enabled. 0111 = GPA.7 wake-up function Enabled. 1000 = GPA.8 wake-up function Enabled. 1001 = GPA.9 wake-up function Enabled. 1010 = GPA.10 wake-up function Enabled. 1011 = GPA.11 wake-up function Enabled. 1100 = GPA.12 wake-up function Enabled. 1101 = GPA.13 wake-up function Enabled. 1110 = GPA.14 wake-up function Enabled. 1111 = GPA.15 wake-up function Enabled.
[3]	**Reserved**	Reserved.
[2]	**PFWKEN**	**Pin Falling Edge Wake-up Enable Bit**

		0 = GPA group pin falling edge wake-up function Disabled. 1 = GPA group pin falling edge wake-up function Enabled.
[1]	**PRWKEN**	**Pin Rising Edge Wake-up Enable Bit** 0 = GPA group pin rising edge wake-up function Disabled. 1 = GPA group pin rising edge wake-up function Enabled.
[0]	**WKEN**	**Standby Power-down Pin Wake-up Enable Bit** 0 = GPA group pin wake-up function Disabled. 1 = GPA group pin wake-up function Enabled.

GPB Standby Power-down Wake-up Control Register (CLK_PBSWKCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PBSWKCTL	CLK_BA+0xA4	R/W	GPB Standby Power-down Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DBEN
7	6	5	4	3	2	1	0
WKPSEL				Reserved	PFWKEN	PRWKEN	WKEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DBEN**	**GPB Input Signal De-bounce Enable Bit** The DBEN bit is used to enable the de-bounce function for each corresponding I/O. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the wake-up. The de-bounce clock source is the 10 kHz internal low speed RC oscillator (LIRC). 0 = Standby power-down wake-up pin De-bounce function Disabled. 1 = Standby power-down wake-up pin De-bounce function Enabled. The de-bounce function is valid only for edge triggered.
[7:4]	**WKPSEL**	**GPB Standby Power-down Wake-up Pin Select** 0000 = GPB.0 wake-up function Enabled. 0001 = GPB.1 wake-up function Enabled. 0010 = GPB.2 wake-up function Enabled. 0011 = GPB.3 wake-up function Enabled. 0100 = GPB.4 wake-up function Enabled. 0101 = GPB.5 wake-up function Enabled. 0110 = GPB.6 wake-up function Enabled. 0111 = GPB.7 wake-up function Enabled. 1000 = GPB.8 wake-up function Enabled. 1001 = GPB.9 wake-up function Enabled. 1010 = GPB.10 wake-up function Enabled. 1011 = GPB.11 wake-up function Enabled. 1100 = GPB.12 wake-up function Enabled. 1101 = GPB.13 wake-up function Enabled. 1110 = GPB.14 wake-up function Enabled. 1111 = GPB.15 wake-up function Enabled.
[3]	**Reserved**	Reserved.
[2]	**PFWKEN**	**Pin Falling Edge Wake-up Enable Bit**

		0 = GPB group pin falling edge wake-up function Disabled. 1 = GPB group pin falling edge wake-up function Enabled.
[1]	**PRWKEN**	**Pin Rising Edge Wake-up Enable Bit** 0 = GPB group pin rising edge wake-up function Disabled. 1 = GPB group pin rising edge wake-up function Enabled.
[0]	**WKEN**	**Standby Power-down Pin Wake-up Enable Bit** 0 = GPB group pin wake-up function Disabled. 1 = GPB group pin wake-up function Enabled.

GPC Standby Power-down Wake-up Control Register (CLK_PCSWKCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PCSWKCTL	CLK_BA+0xA8	R/W	GPC Standby Power-down Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DBEN
7	6	5	4	3	2	1	0
WKPSEL				Reserved	PFWKEN	PRWKEN	WKEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DBEN**	**GPC Input Signal De-bounce Enable Bit** The DBEN bit is used to enable the de-bounce function for each corresponding I/O. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the wake-up. The de-bounce clock source is the 10 kHz internal low speed RC oscillator. 0 = Standby power-down wake-up pin De-bounce function Disabled. 1 = Standby power-down wake-up pin De-bounce function Enabled. **Note:** The de-bounce function is valid only for edge triggered.
[7:4]	**WKPSEL**	**GPC Standby Power-down Wake-up Pin Select** 0000 = GPC.0 wake-up function Enabled. 0001 = GPC.1 wake-up function Enabled. 0010 = GPC.2 wake-up function Enabled. 0011 = GPC.3 wake-up function Enabled. 0100 = GPC.4 wake-up function Enabled. 0101 = GPC.5 wake-up function Enabled. 0110 = GPC.6 wake-up function Enabled. 0111 = GPC.7 wake-up function Enabled. 1000 = GPC.8 wake-up function Enabled. 1001 = GPC.9 wake-up function Enabled. 1010 = GPC.10 wake-up function Enabled. 1011 = GPC.11 wake-up function Enabled. 1100 = GPC.12 wake-up function Enabled. 1101 = GPC.13 wake-up function Enabled. 1110 = GPC.14 wake-up function Enabled. 1111 = GPC.15 wake-up function Enabled.
[3]	**Reserved**	Reserved.
[2]	**PFWKEN**	**Pin Falling Edge Wake-up Enable Bit**

		0 = GPC group pin falling edge wake-up function Disabled. 1 = GPC group pin falling edge wake-up function Enabled.
[1]	**PRWKEN**	**Pin Rising Edge Wake-up Enable Bit** 0 = GPC group pin rising edge wake-up function Disabled. 1 = GPC group pin rising edge wake-up function Enabled.
[0]	**WKEN**	**Standby Power-down Pin Wake-up Enable Bit** 0 = GPC group pin wake-up function Disabled. 1 = GPC group pin wake-up function Enabled.

GPD Standby Power-down Wake-up Control Register (CLK_PDSWKCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PDSWKCTL	CLK_BA+0xAC	R/W	GPD Standby Power-down Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DBEN
7	6	5	4	3	2	1	0
WKPSEL				Reserved	PFWKEN	PRWKEN	WKEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DBEN**	**GPD Input Signal De-bounce Enable Bit** The DBEN bit is used to enable the de-bounce function for each corresponding I/O. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the wake-up. The de-bounce clock source is the 10 kHz internal low speed RC oscillator. 0 = Standby power-down wake-up pin De-bounce function Disabled. 1 = Standby power-down wake-up pin De-bounce function Enabled. **Note:** The de-bounce function is valid only for edge triggered.
[7:4]	**WKPSEL**	**GPD Standby Power-down Wake-up Pin Select** 0000 = GPD.0 wake-up function Enabled. 0001 = GPD.1 wake-up function Enabled. 0010 = GPD.2 wake-up function Enabled. 0011 = GPD.3 wake-up function Enabled. 0100 = GPD.4 wake-up function Enabled. 0101 = GPD.5 wake-up function Enabled. 0110 = GPD.6 wake-up function Enabled. 0111 = GPD.7 wake-up function Enabled. 1000 = GPD.8 wake-up function Enabled. 1001 = GPD.9 wake-up function Enabled. 1010 = GPD.10 wake-up function Enabled. 1011 = GPD.11 wake-up function Enabled. 1100 = GPD.12 wake-up function Enabled. 1101 = GPD.13 wake-up function Enabled. 1110 = GPD.14 wake-up function Enabled. 1111 = GPD.15 wake-up function Enabled.
[3]	**Reserved**	Reserved.
[2]	**PFWKEN**	**Pin Falling Edge Wake-up Enable Bit**

		0 = GPD group pin falling edge wake-up function Disabled. 1 = GPD group pin falling edge wake-up function Enabled.
[1]	**PRWKEN**	**Pin Rising Edge Wake-up Enable Bit** 0 = GPD group pin rising edge wake-up function Disabled. 1 = GPD group pin rising edge wake-up function Enabled.
[0]	**WKEN**	**Standby Power-down Pin Wake-up Enable Bit** 0 = GPD group pin wake-up function Disabled. 1 = GPD group pin wake-up function Enabled.

GPIO Standby Power-down Control Register (CLK_IOPDCTL)

Register	Offset	R/W	Description	Reset Value
CLK_IOPDCTL	CLK_BA+0xB0	R/W	GPIO Standby Power-down Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**IOHR**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**IOHR**	**GPIO Hold Release** When GPIO enters deep power-down mode or standby power-down mode, all I/O status are hold to keep normal operating status. After chip is woken up from deep power-down mode or standby power-down mode, the I/O are still keep hold status until user set this bit to release I/O hold status. **Note:** This bit is auto cleared by hardware.

6.4 True Random Number Generator (TRNG)

6.4.1 Overview

The purpose of True Random Number Generator (TRNG) is to generate the randomness by extracting from physical phenomena.

6.4.2 Features

- 800 random bits per second
- Provides the true random number seed for PRNG

6.4.3 Block Diagram

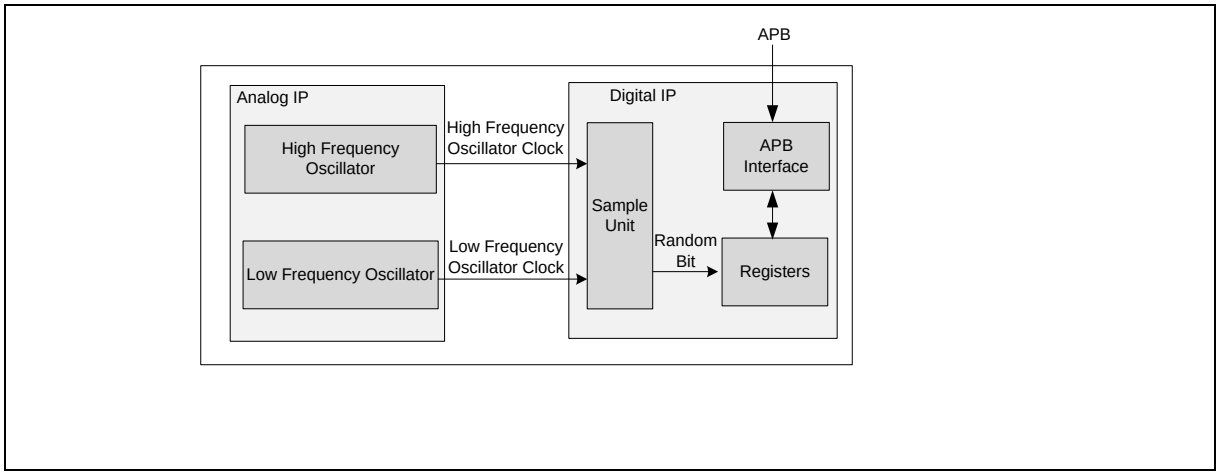


Figure 6.4-1 True Random Number Generator Block Diagram

6.4.4 Basic Configuration

6.4.4.1 Clock Source Configuration

- Enable 32 kHz clock in LXTEN (CLK_PWRCTL[1]), source from LXT or
- Enable 10 kHz clock in TRNGSEL (CLK_CLKSEL2[27]), source from LIRC
- Enable TRNG peripheral clock in TRNGCKEN (CLK_APBCLK1[25]).

6.4.4.2 Reset Configuration

- Reset TRNG controller in TRNGRST (SYS_IPRST1[31]).

6.4.5 Functional Description

The TRNG can provide the random number when activated and enabled. When DVIF (TRNG_CTL[1]) bit is set, the random bits may be read one byte at a time form TRNG_DATA register. If DVIF is 0, the TRNG_DATA register returns the value 0x0 when read. After read the TRNG_DATA register, the DVIF will be set to 0. If DVIEN (TRNG_CTL[6]) is enabled, the TRNG will generate interrupt to CPU when DVIF is set.

Before changing CLKPSC (TRNG_CTL[5:2]), TRNGEN (TRNG_CTL[0]) bit should be set to 0.

The programming steps to get the true random number are depicted below.

1. Select TRNG peripheral clock frequency in CLKPSC (TRNG_CTL[5:2]).
2. Enable ACT (TRNG_ACT[7]) bit.
3. Wait for READY (TRNG_CTL[7]) bit to be set to 1.

4. Enable TRNGEN (TRNG_CTL[0]) bit.
5. Wait DVIF (TRNG_CTL[1]) to bit be set to 1.
6. Read TRNG_DATA register to get random number.

Repeat Step 5 and 6 to get more random number.

TRNG generates the true random number seed for PRNG

The programming steps of PRNG to get the random number with the true random number seed from TRNG are depicted below.

1. Select TRNG peripheral clock frequency in CLKPSC (TRNG_CTL[5:2]).
2. Enable ACT (TRNG_ACT[7]) bit.
3. Wait READY (TRNG_CTL[7]) bit to be set to 1.
4. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
5. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
6. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then software can read the output random numbers from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7.

Some Notices of TRNG:

1. It is not suggested that enabling TRNGEN (TRNG_CTL[0]) and TRNG generates the true random number seed for PRNG at the same time. If users still want to execute the situation, TRNG generates the true random number seed for PRNG will have a higher priority than TRNGEN.
2. After TRNG provides the random number (8 bits), it means that DVIF (TRNG_CTL[1]) bit is set to 1. If users execute that TRNG generates the true random number seed for PRNG, the DVIF (TRNG_CTL[1]) bit is still set to 0 because the random number (8-bits) has been taken away for generating TRNG seed.

6.4.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
TRNG Base Address: **TRNG_BA = 0x400B_9000**				
TRNG_CTL	TRNG_BA+0x00	R/W	TRNG Control Register and Status	0x0000_0000
TRNG_DATA	TRNG_BA+0x04	R	TRNG Data Register	0x0000_0000
TRNG_ACT	TRNG_BA+0x0C	R/W	TRNG Activation Register	0x0000_0002

6.4.7 Register Description

TRNG Control Register and Status (TRNG_CTL)

Register	Offset	R/W	Description	Reset Value
TRNG_CTL	TRNG_BA+0x00	R/W	TRNG Control Register and Status	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
READY	**DVIEN**	**CLKPSC**				**DVIF**	**TRNGEN**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**READY**	**Random Number Generator Ready (Read Only)** After ACT (TRNG_ACT[7]) bit is set, the READY bit becomes 1 after a delay of 90us~120us. 0 = RNG is not ready or not activated. 1 = RNG is ready to be enabled.
[6]	**DVIEN**	**Data Valid Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[5:2]	**CLKPSC**	**Clock Prescaler** The CLKPSC is the peripheral clock frequency range for the selected value. The CLKPSC must be higher than or equal to the actual peripheral clock frequency (for correct random bit generation). However, if the actual peripheral clock frequency is 10 MHz, and users set CLKPSC 4 bits '0' (80 ~ 100 MHz), users still will get random number, but performance will be lower than 800 bits/s. To change the CLKPSC contents, first set TRNGEN bit to 0 and then change CLKPSC; finally, set TRNGEN bit to 1 to re-enable the TRNG module. 0000 = 80 ~ 100 MHz. 0001 = 60 ~ 80 MHz. 0010 = 50 ~60 MHz. 0011 = 40 ~50 MHz. 0100 = 30 ~40 MHz. 0101 = 25 ~30 MHz. 0110 = 20 ~25 MHz. 0111 = 15 ~20 MHz. 1000 = 12 ~15 MHz. 1001 = 9 ~12 MHz. 1010 = 7 ~9 MHz. 1011 = 6 ~7 MHz. 1100 = 5 ~6 MHz.

		1101 = 4 ~5 MHz. 1111 = Reserved.
[1]	**DVIF**	**Data Valid (Read Only)** 0 = Data is not valid. Reading from RNGD returns 0x00000000. 1 = Data is valid. A valid random number can be read form RNGD. **Note:** This bit is cleared to '0' by reading TRNG_DATA.
[0]	**TRNGEN**	**Random Number Generator Enable Bit** This bit can be set to 1 only after ACT (TRNG_ACT[7]) bit is set to 1 and READY (TRNG_CTL[7]) bit became 1. 0 = TRNG Disabled. 1 = TRNG Enabled. **Note:** TRNGEN is an enable bit of digital part. When TRNG is not required to generate random number, TRNGEN bit and ACT (TRNG_ACT[7]) bit should be set to 0 to reduce power consumption.

TRNG Data Register (TRNG_DATA)

Register	Offset	R/W	Description	Reset Value
TRNG_DATA	TRNG_BA+0x04	R	TRNG Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DATA**	**Random Number Generator Data (Read Only)** The DATA stores the random number generated by TRNG and can be read only once.

TRNG Activation Register (TRNG_ACT)

Register	Offset	R/W	Description	Reset Value
TRNG_ACT	TRNG_BA+0x0C	R/W	TRNG Activation Register	0x0000_0002

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACT	**Reserved**						

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**ACT**	**Random Number Generator Activation** After enabling the ACT bit, it will activate the TRNG module and wait the READY (TRNG_CTL[7]) bit to become 1. 0 = TRNG inactive. 1 = TRNG active. **Note:** ACT is an enable bit of analog part. When TRNG is not required to generate random number, TRNGEN (TRNG_CTL[0]) bit and ACT bit should be set to 0 to reduce power consumption.
[6:0]	**Reserved**	Reserved.

6.5 Flash Memory Controller (FMC)

6.5.1 Overview

The FMC is equipped with 256/512/1024 Kbytes on-chip embedded Flash for application and configurable Data Flash to store some application dependent data. Thus, the total size of application rom (APROM) is 256/512/1024 Kbytes. A User Configuration block provides for system initiation. A 8 Kbytes loader ROM (LDROM) is used for In-System-Programming (ISP) function. A 3 Kbytes one-time-program ROM (OTP) is used for recording one-time-program data. A 16/24 Kbytes Boot Loader consists of native ISP functions and secure boot function. A 4 Kbytes cache with zero wait cycle is used to improve Flash access performance. This chip also supports In-Application-Programming (IAP) function. User switches the code executing without chip reset after the embedded Flash is updated.

6.5.2 Features

- Supports dual-bank Flash macro for safe firmware upgrade
- Supports 256/512/1024 Kbytes application ROM (APROM)
- Supports 8 Kbytes loader ROM (LDROM)
- Supports 4 XOM (eXecution Only Memory) regions to conceal user program in APROM.
- Supports Data Flash with configurable memory size
- Supports 16 bytes User Configuration block to control system initiation
- Supports 3 Kbytes one-time-program ROM (OTP)
- Supports 4 Kbytes page erase for all embedded Flash
- Supports bank erase for APROM, except for XOM regions
- Supports Boot Loader with native In-System-Programming (ISP) functions
- Supports Secure Boot function for code integrity and authenticity
- Supports 32-bit/64-bit and multi-word Flash programming function
- Supports fast Flash programming verification function
- Supports CRC32 checksum calculation function
- Supports Flash all one verification function
- Supports In-System-Programming (ISP) / In-Application-Programming (IAP) to update embedded Flash memory
- Supports cache memory to improve Flash access performance and reduce power consumption

FMC Features		M463	M467
Dual Bank Architecture	6.5.4.2 Dual Bank Architecture	-	●
REMAP	6.5.4.4 Physical and Virtual Address Concep 6.5.4.5 APROM Reboot Address Operation Model Selection	-	●

Table 6.5-1 FMC Features Comparison Table at Different Chip

6.5.3 Block Diagram

The Flash memory controller (FMC) consists of AHB slave interface, cache memory controller, boot loader, Flash control registers, Flash initialization controller, Flash operation control and embedded Flash memory. The block diagram of Flash memory controller is shown as follows.

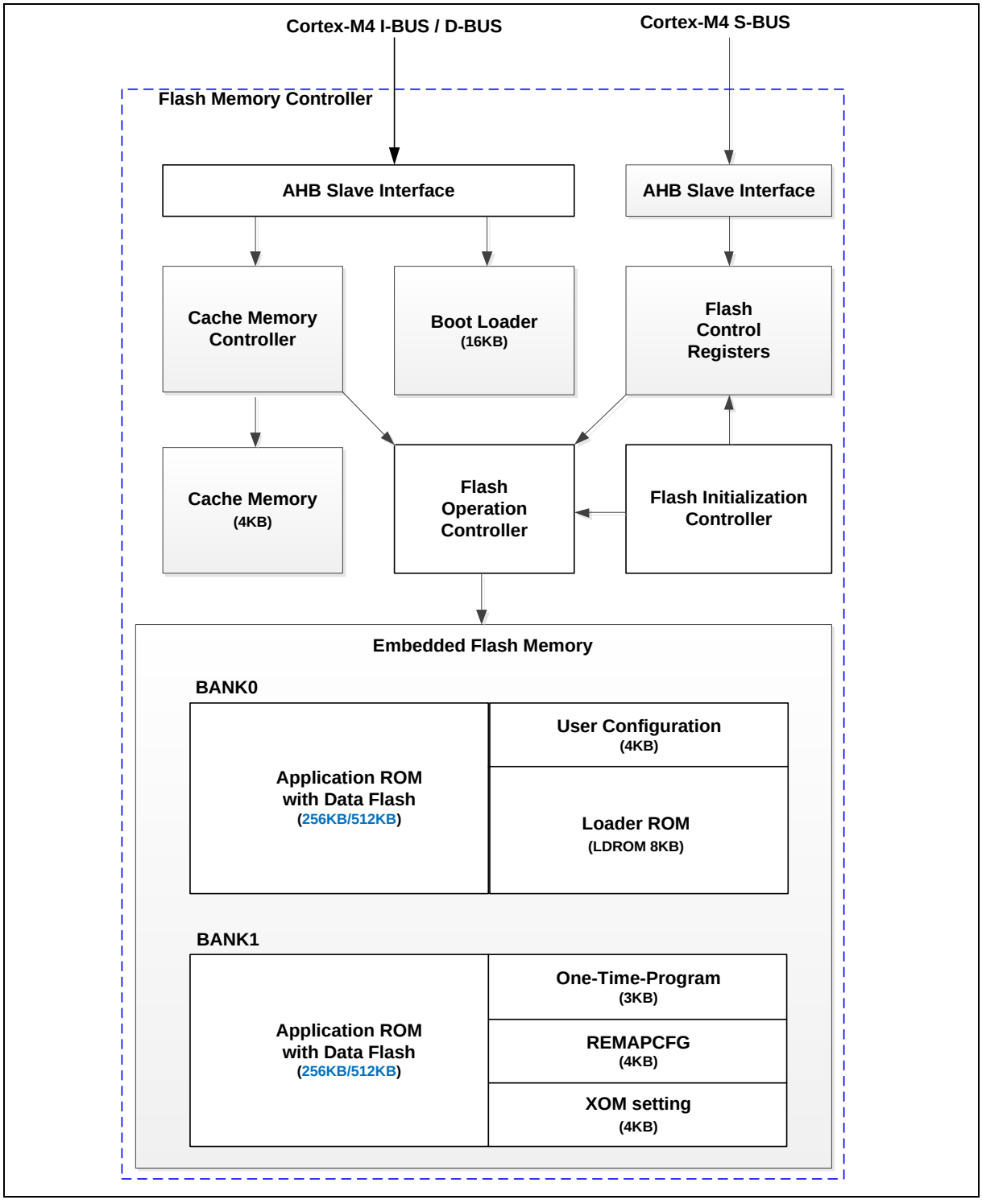


Figure 6.5-1 Flash Memory Controller Block Diagram

AHB Slave Interface

There are two AHB slave interfaces in Flash memory controller, one is from both Cortex-M4 I-Bus and D-Bus for the instruction and data fetch; the other is from Cortex-M4 S-Bus for Flash control registers access including ISP registers.

Cache Memory Controller

A 4 Kbyes cache with zero wait cycle is implemented between Cortex-M4 CPU and embedded Flash memory. This cache memory controller improves the Flash access performance and reduces power consumption of the embedded Flash memory.

Boot Loader

The Boot Loader is 16/24 Kbytes, and consists of native ISP functions to update embedded Flash memory The Boot Loader content is read only, but not programmable.

Flash Control Registers

All of ISP control and status registers are in the Flash control registers. The detail registers description is in the Register Description section

Flash Initialization Controller

When chip is powered on or active from reset, the Flash initialization controller will start to access Flash automatically and check the Flash stability, and also reload User Configuration content to the Flash control registers for system initiation.

Flash Operation Controller

The Flash operations, such as Flash erase, Flash program, and Flash read operation, have specific control timing for embedded Flash memory. The Flash operation controller generates those control timing by requested from the cache memory controller, the Flash control registers and the Flash initialization controller.

Embedded Flash Memory

The embedded Flash memory is the main memory for user application code and parameters. It is consists of the user configuration block, 8 Kbytes LDROM,4 Kbytes XOM setting pages, 3 Kbytes OTP and 256/512/1024 Kbytes APROM with Data Flash. The page erase Flash size is 4 Kbytes, and minimum program bit size is 32 bits.

6.5.4 Functional Description

FMC functions include the memory organization, boot selection, secure boot, IAP, ISP, the embedded Flash programming, and checksum calculation. The Flash memory map and system memory map are also introduced in the memory organization.

6.5.4.1 Memory Organization

The FMC memory consists of the embedded Flash memory and boot loader. The embedded Flash memory is programmable, and includes APROM, LDROM,XOM setting page,Data Flash, the User Configuration block OTP. Boot Loader is a Mask ROM with ISP boot codes to support firmware download, boot control, security control and firmware execution. The address map includes Flash memory map and five system address maps: LDROM with IAP, LDROM without IAP, APROM with IAP, APROM without IAP, and Boot Loader with IAP functions.

BANK	Flash Memory Block	Address Range
0	APROM with 256 Kbytes	0x0000_0000 ~ 0x0003_FFFF
	APROM with 512 Kbytes	0x0000_0000 ~ 0x0003_FFFF
	APROM with 1024 Kbytes	0x0000_0000 ~ 0x0007_FFFF
	User Configuration	0x0F30_0000 ~ 0x0F30_000F
	LDROM	0x0F10_0000 ~ 0x0F10_1FFF
1	APROM with 512 Kbytes	0x0004_0000 ~ 0x0007_FFFF
	APROM with 1024B	0x0008_0000 ~ 0x000F_FFFF
	OTP	0x0F31_0000 ~ 0x0F31_0BFF
	XOM Setting	0x0F20_0000 ~ 0x0F20_07FF

Table 6.5-2 Block Address Range

6.5.4.2 Dual Bank Architecture

The M463/M467 series FMC also provides dual bank architecture where APROM is separated into Bank0 and Bank1. Bank0's Flash address range is from 0x0 to 0x3FFFF/7FFFF while Bank1's Flash address range is from 0x40000/80000 to 0x7FFFF/FFFFF. In such dual bank architecture, a firmware upgrade mechanism is provided, which uses APROM Bank0 as firmware executing region and APROM Bank1 as storage region for newer version of firmware. The firmware in Bank0 can erase Bank1 and program new firmware to Bank1 without stopping instruction fetching form Bank0. This is so called read-while-write (RWW), which provides the ability for firmware to keep software services while firmware updating. The firmware upgrade mechanism here also provides a rollback process to roll back to the original firmware if a new firmware woks abnormally. Such mechanism ensures the firmware to be safe to update in any circumstance.

6.5.4.3 LDROM APROM and Data Flash

LDROM is designed for a loader to implement In-System-Programming (ISP) function by user. LDROM is a 8 Kbyes embedded Flash memory, the Flash address range is from 0x0F10_0000 to 0x0F10_1FFF. APROM is main memory for user applications. APROM size is 256/512/1024 Kbytes. Data Flash is used to store application parameters (not instruction). Data Flash is shared with APROM and size is configurable. The base address of Data Flash is determined by DFBA (CONFIG1[19:0]). All of the embedded Flash memory is 4 Kbytes page erased.

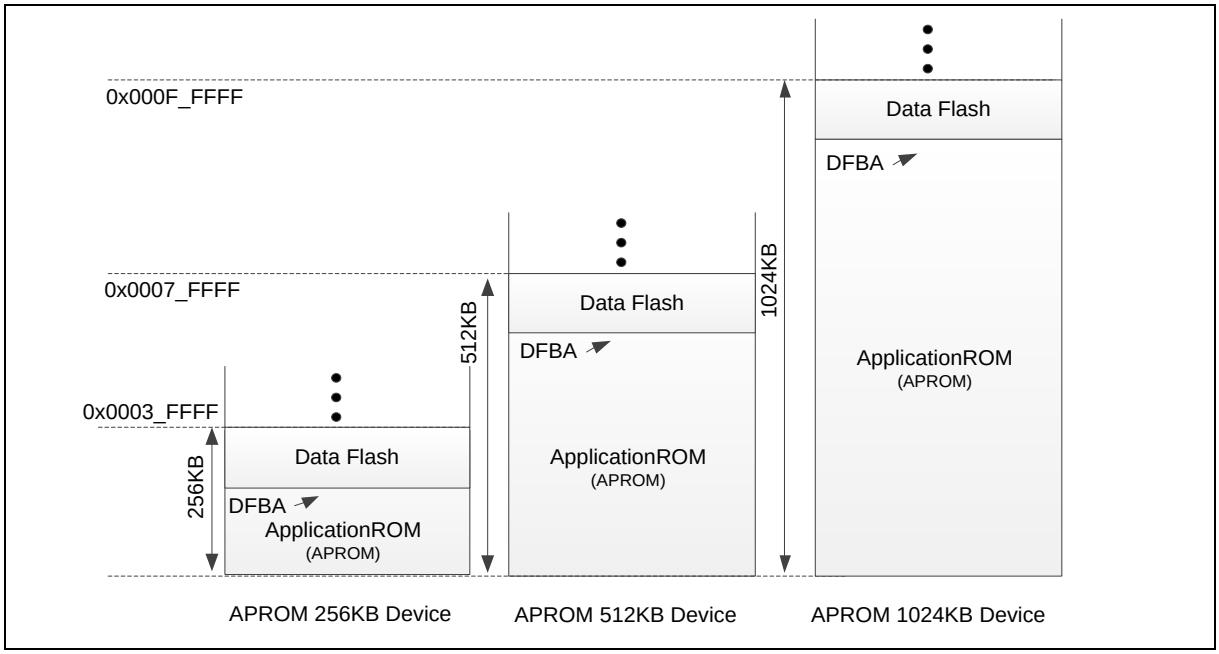


Figure 6.5-2 Data Flash Shared with APROM Examples (256/512/1024 Kbytes)

6.5.4.4 Physical and Virtual Address Concept

For OTA application, the memory space concept is a little different form FMC commands and CPU viewpoint.

- From FMC command viewpoint, physical address concept is adopted. All of FMC commands regard APROM memory as physical address except “remapping” command. It means APROM Flash memory will be flat memory model regardless of memory remapping state.
- From Data Flash viewpoint, virtual address concept is adopted.
- From CPU viewpoint, virtual address concept is adopted. APROM memory space is separated into 2 segments (banks). As runtime stage, the virtual address depends on what address operation model (OP0, OP1) is, as shown in Figure 6.5-3.

6.5.4.5 APROM Reboot Address Operation Model Selection

In M463/M467 series, to activate APROM reboot with different address operation model, use address operation model selection command 0x2C with FMC_ISPDAT “0x5AA5_5AA5” and FMC_ISPADDR ”0x0”(Address OP0) or ”0x1”(Address OP1) to select address operation model.User can program REMAPCFG(0xF32_0000) to active the function.

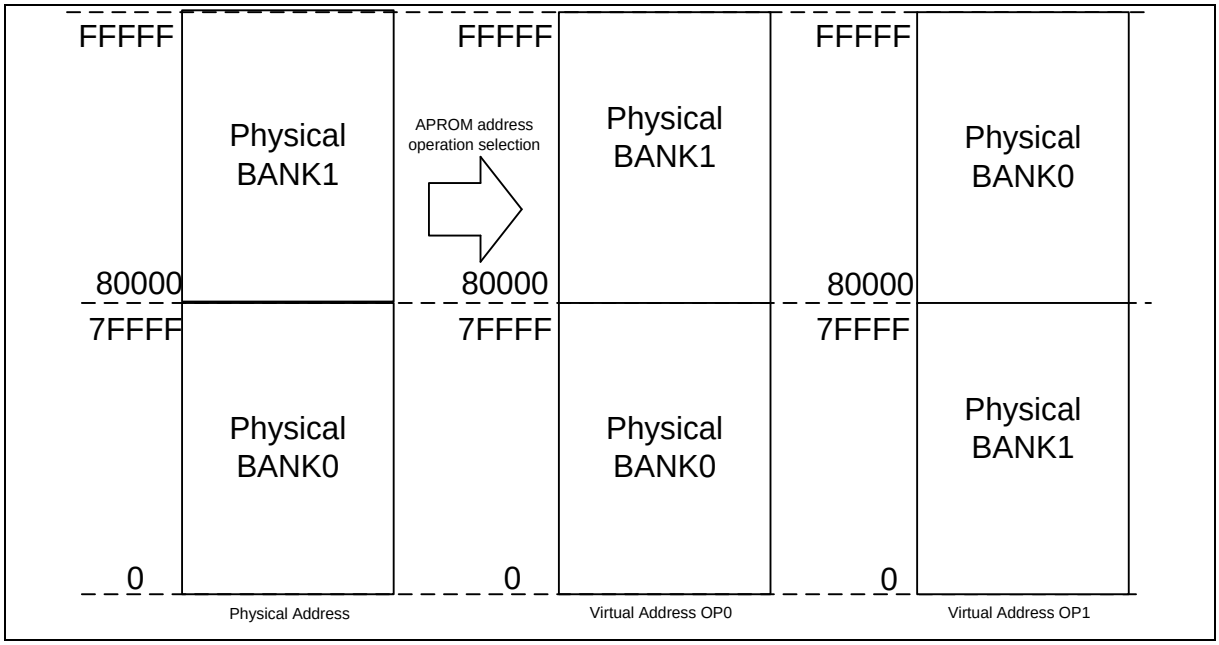


Figure 6.5-3 Address Operation Model

6.5.4.6 User Configuration Block

User Configuration block is internal programmable configuration area for boot options, such as Flash security lock, boot select, brown-out voltage level, and Data Flash base address. It works like a fuse for power on setting. It is loaded from Flash memory to its corresponding control registers during chip power on. User can set these bits according to different application requests. User Configuration block can be updated by ISP function and located at 0x0F30_0000 with four 32 bits words (CONFIG0, CONFIG1, CONFIG2 and CONFIG3). Any change on User Configuration block will take effect after system reboot.

Note:

1. It suggests to program User Configuration Block(CONFIG0, CONFIG1, CONFIG2 and CONFIG3) by ICP tool. If it is updated by firmware and meanwhile the power is shut down, the setting value may not have a correct value.

CONFIG0 (Address = 0x0F30_0000)

31	30	29	28	27	26	25	24
CWDTEN[2]	CWDTPDEN	Reserved					
23	22	21	20	19	18	17	16
CBOV			CBORST	CBODEN	Reserved		
15	14	13	12	11	10	9	8
Reserved				ICELOCK	CIOINI	Reserved	
7	6	5	4	3	2	1	0
CBS	Reserved		CWDTEN[1:0]		Reserved	LOCK	DFEN

Bits	Descriptions	
[31]	**CWDTEN[2]**	**Watchdog Timer Hardware Enable Bit** When watchdog timer hardware enable function is enabled, the watchdog enable bit WDTEN (WDT_CTL[7]) and watchdog reset enable bit RSTEN (WDT_CTL[1]) is set to 1 automatically after power on. The clock source of watchdog timer is force at LIRC and LIRC can't be disabled. **CWDTEN[2:0]** is CONFIG0[31][4][3], 011 = WDT hardware enable function is active. WDT clock is always on except chip enters Power-down mode. When chip enters Power-down mode, WDT clock is always on if CWDTPDEN is 0 or WDT clock is controlled by LIRCEN (CLK_PWRCTL[3]) if CWDTPDEN is 1. Please refer to bit field description of CWDTPDEN. 111 = WDT hardware enable function is inactive. Others = WDT hardware enable function is active. WDT clock is always on.
[30]	**CWDTPDEN**	**Watchdog Clock Power-down Enable Bit** 0 = Watchdog Timer clock kept enabled when chip enters Power-down. 1 = Watchdog Timer clock is controlled by LIRCEN (CLK_PWRCTL[3]) when chip enters Power-down. **Note:** This bit only works if CWDTEN[2:0] is set to 011
[29:24]	**Reserved**	Reserved.
[23:21]	**CBOV**	**Brown-out Voltage Selection** 000 = Brown-out voltage is 1.6V. 001 = Brown-out voltage is 1.8V. 010 = Brown-out voltage is 2.0V. 011 = Brown-out voltage is 2.2V. 100 = Brown-out voltage is 2.4V. 101 = Brown-out voltage is 2.6V. 110 = Brown-out voltage is 2.8V. 111 = Brown-out voltage is 3.0V.
[20]	**CBORST**	**Brown-out Reset Enable Bit** 0 = Brown-out reset Enabled after powered on. 1 = Brown-out reset Disabled after powered on.
[19]	**CBODEN**	**Brown-out Detector Enable Bit** 0= Brown-out detect Enabled after powered on. 1= Brown-out detect Disabled after powered on.

[18:12]	**Reserved**	Reserved.
[11]	**ICELOCK**	**ICE Lock Bit** This bit is only used to disable ICE function. User may use it with LOCK (CONFIG0[1]) bit or ALOCK (CONFIG2[7:0]) bits to increase security level. 0 = ICE function Disabled. 1 = ICE function Enabled.
[10]	**CIOINI**	**I/O Initial State Selection** 0 = All GPIO set as Quasi-bidirectional mode after chip powered on. 1 = All GPIO set as input tri-state mode after powered on.
[9:8]	**Reserved**	Reserved.
[7]	**CBS**	**Chip Booting Selection** IAP mode is supported in M463/M467 series, and the code in LDROM and APROM can be called by each other. 0 = Direct boot from LDROM through bootloader. 1 = Direct boot from APROM through bootloader.
[6:5]	**Reserved**	Reserved.
[4:3]	**CWDTEN**	**Watchdog Timer Hardware Enable Bit** When watchdog timer hardware enable function is enabled, the watchdog enable bit WDTEN (WDT_CTL[7]) and watchdog reset enable bit RSTEN (WDT_CTL[1]) is set to 1 automatically after power on. The clock source of watchdog timer is force at LIRC and LIRC can't be disable. **CWDTEN[2:0]** is CONFIG0[31][4][3], 011 = WDT hardware enable function is active. WDT clock is always on except chip enters Power-down mode. When chip enters Power-down mode, WDT clock is always on if CWDTPDEN is 0 or WDT clock is controlled by LIRCEN (CLK_PWRCTL[3]) if CWDTPDEN is 1. Please refer to bit field description of CWDTPDEN. 111 = WDT hardware enable function is inactive. Others = WDT hardware enable function is active. WDT clock is always on.
[2]	**Reserved**	Reserved.
[1]	**LOCK**	**Security Lock Control** 0 = Flash memory content is locked. 1 = Flash memory content is not locked if ALOCK(CONFIG2[7:0]) and KSPLOCK (CONFIG2[15:8]) are 0x5A. When Flash data is locked by LOCK,user can look up the lock effect of FMC in lock effect tables.
[0]	**DFEN**	**Data Flash Enable Bit** The Data Flash is shared with APROM, and the base address of Data Flash is decided by DFBA (CONFIG1[19:0]) when DFEN is 0. 0 = Data Flash Enabled. 1 = Data Flash Disabled.

Note: The config bits should be 1 if reserved.

CONFIG1 (Address = 0x0F30_0004)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				DFBA			
15	14	13	12	11	10	9	8
DFBA							
7	6	5	4	3	2	1	0
DFBA							

Bits	Descriptions	
[31:20]	**Reserved**	Reserved.
[19:0]	**DFBA**	**Data Flash Base Address** This register works only when DFEN (CONFIG0[0]) is set to 0. If DFEN (CONFIG0[0]) is set to 0, the Data Flash base address is defined by user. Since on-chip Flash erase unit is 4 Kbytes, it is mandatory to keep bit 11-0 as 0.

Note: The config bits should be 1 if reserved.

CONFIG2 (Address = 0x0F30_0008)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
KSPLOCK							
7	6	5	4	3	2	1	0
ALOCK							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved.
[15:8]	**KSPLOCK**	**Key Store Protection Lock** 0x5A = Flash memory content is unlocked if LOCK (CONFIG0[1]) is 1 and ALOCK (CONFIG2[7:0]) is 0x5A. The others = Flash memory content is also locked. When KSPLOCK ≠0x5A with ICE, only Chip Erase ISP command can be performed via ICE. When Flash data is locked by KSPLOCK,user must look the lock effect of FMC up in lock effect tables. When KSPLOCK is enable,user use mass erase would cause Key Store revoke all OTP key.
[7:0]	**ALOCK**	**Advance Security Lock Control** 0x5A = Flash memory content is unlocked if LOCK (CONFIG0[1]) is 1. The others = Flash memory content is locked. When Flash data is locked by ALOCK,user must look the lock effect of FMC up in lock effect tables.

Note: The config bits should be 1 if reserved.

CONFIG3 (Address = 0x0F30_000C)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			SPIISPEN	I2CISPEN	CANISPEN	USBISPDIS	UARTISPEN
7	6	5	4	3	2	1	0
Reserved							

Bits	Descriptions	
[31:13]	**Reserved**	Reserved.
[12]	**SPIISPEN**	**SPI ISP Mode Enable Bit** The pins used in SPI ISP mode are SPI0 MOSI on PA.0, MISO on PA.1, CLK on PA.2 and SS on PA.3. 0= SPI ISP mode Disabled. 1= SPI ISP mode Enabled.
[11]	**I2CISPEN**	**I^{2}C ISP Mode Enable Bit** The pins used in I^{2}C ISP mode are I2C0 SDA on PC.0 and I2C0 SCL on PC.1. 0= I^{2}C ISP mode Disabled. 1= I^{2}C ISP mode Enabled.
[10]	**CANISPEN**	**CAN ISP Mode Enable Bit** The pins used in CAN ISP mode are CAN0 RXD on PA.4 and CAN0 TXD on PA.5. 0= CAN ISP mode Disabled. 1= CAN ISP mode Enabled.
[9]	**USBISPDIS**	**USB ISP Mode Disable Bit** The USB ISP mode runs in HS-USBD mode. 0= USB ISP mode Enabled. 1= USB ISP mode Disabled.
[8]	**UARTISPEN**	**UART ISP Mode Enable Bit** The pins used in UART ISP mode are UART0 RXD on PB.12 and UART0 TXD on PB.13. 0= UART ISP mode Disabled. 1= UART ISP mode Enabled.
[7:0]	**Reserved**	Reserved.

Note: The config bits should be 1 if reserved.

6.5.4.7 Execution Only Memory (XOM)

The execution only memory (XOM) is used to store instructions for security application which are not allowed for data access via AHB-Bus. There are four XOM regions in APROM at most. To define a new XOM region, user must complete XOM settings in XOM setting page (0xF20_0000 ~ 0xF20_0008 for XOM region 0; 0xF20_0010 ~ 0xF20_0018 for XOM region 1 ; 0xF20_0020 ~ 0xF20_0028 for XOM region 2 ; 0xF20_0030 ~ 0xF20_0038 for XOM region 3) via In-System-Programming (ISP) function. When user setting XOM region,need to set Base address first then set XOM page size, as shown in Figure 6.5-4. User should not set XOM at region where chip can boot from.User should not set the XOMs overlap each other, because it would cause XOM cannot normal active. After setting and restaring Flash initialization, user can check XOM status from FMC_XOMSTS and check each XOM range (i.e., base and size) from FMC_XOMR0STS~FMC_XOMR3STS. In ICP function or WRITER mode, user can execute command 0x0 to read XOM setting page and check XOM status. User need to do chip reset (Flash initial) after setting or erasing XOM.

When using XOM setting page and XOM regions, users must follow the limitations below:

- All XOM regions must be located in APROM with page size alignment.
- The XOM setting cannot exceed the APROM range,otherwise unexpected errors will occur.
- Any XOM region cannot be programmed after its XOM setting is active.
- The base address of all XOM cannot be set as zero (first page).
- Any XOM region must not include address 0xFF000~0xFFFFF.
- Any XOM setting cannot be changed during runtime.
- Only use Flash 32-bit program command to program XOM setting page.
- Clear XOM setting and XOM region by mass erase command or XOM page erase function (a special erase command for XOM).
- User must restart Flash initialization to activate any new XOM setting.
- ICE cannot step in XOM only if debug mode.

Once the XOM region is active, user only can adopt mass erase command or XOM page erase function to clear the XOM setting of one XOM region. To execute XOM page erase function, user must perform FLASH Page Erase (0x22) by writing the base address of the target XOM region (address in APROM, not in XOM setting page) in FMC_ISPADDR and the constant value 0x0055aa03 in FMC_ISPDAT. Note thant, the valid commands in every active XOM regsion are chip erase, page erase, chksum and read CID/DID. The XOM setting page is not accessed, except for chip erase, word program and 32-bit read. Note that the read-while-write function in dual-bank architecture will stop when executing XOM page erase function. When setting the XOM region,user need to set Base address first and then set XOM page size. User should not set XOM at region where chip can boot from.User should not set the XOMs overlap each other,because it would cause XOM cannot be normal active.

Example:

XOM0 base = 0x5000

XOM0 size = 3

XOM1 base = 0x4000

XOM1 size = 3 ---> ISPFF

XOM1 won't be active

In this case XOM1 base is already set,user can use XOM page erase to erase XOM1 base to clear the wrong setting.

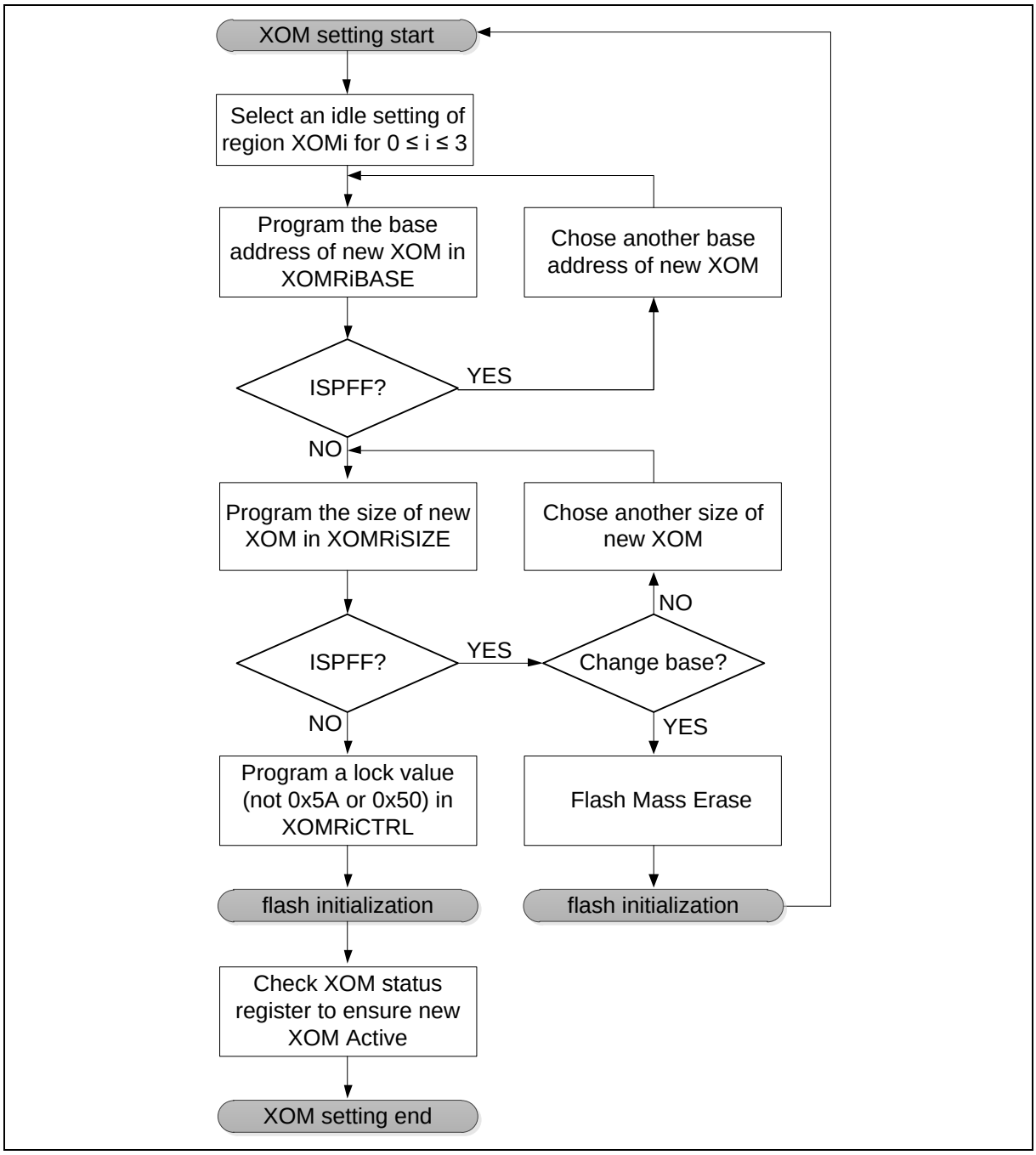


Figure 6.5-4 New XOM Setting Flow

XOMR0BASE (Address = 0x0F20_0000)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
XOMR0BASE							
15	14	13	12	11	10	9	8
XOMR0BASE							
7	6	5	4	3	2	1	0
XOMR0BASE							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved.
[23:0]	**XOMR0BASE**	**Base Address of XOM Region 0** XOMR0BASE must be page-aligned. **Note:** The page size is 4 Kbytes.

XOMR0SIZE (Address = 0x0F20_0004)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR0SIZE							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR0SIZE**	**Page Number of XOM Region 0** XOMR0SIZE must be page-aligned and less than 63 pages. The XOMR0SIZE minimum value is 1 page. If XOMR0SIZE is written to 0, the register value will be set to 1. **Note:** The page size is 4 Kbytes.

XOMR0CTRL (Address = 0x0F20_0008)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR0CTRL							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR0CTRL**	**Control of XOM Region 0** 0x5a = XOM region 0 is off. 0x50= XOM is in debug mode. The others = XOM region 0 is active. **Note 1:** But for Flash behavior,user cannot write 0 to 1.User need to check the lock value is effective to change 0x5A(0101_1010). **Note 2:** The active state has come into effect after power-on or reset cycle.

XOMR1BASE (Address = 0x0F20_0010)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
XOMR1BASE							
15	14	13	12	11	10	9	8
XOMR1BASE							
7	6	5	4	3	2	1	0
XOMR1BASE							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved.
[23:0]	**XOMR1BASE**	**Base Address of XOM Region 1** XOMR1BASE must be page-aligned. **Note:** The page size is 4 Kbytes.

XOMR1SIZE (Address = 0x0F20_0014)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR1SIZE							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR1SIZE**	**Page Number of XOM Region 1** XOMR1SIZE must be page-aligned and less than 63 pages. The XOMR1SIZE minimum value is 1 page. If XOMR1SIZE is written to 0, the register value will be set to 1. **Note:** The page size is 4 Kbytes.

XOMR1CTRL (Address = 0x0F20_0018)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR1CTRL							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR1CTRL**	**Control of XOM Region 1** 0x5a = XOM region 1 is off. 0x50= XOM is in debug mode. The others = XOM region 1 is active. **Note 1:** But for Flash behavior,user cannot write 0 to 1.User need to check the lock value is effective to change 0x5A(0101_1010). **Note 2:** The active state has come into effect after power-on or reset cycle.

XOMR2BASE (Address = 0x0F20_0020)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
XOMR2BASE							
15	14	13	12	11	10	9	8
XOMR2BASE							
7	6	5	4	3	2	1	0
XOMR2BASE							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved.
[23:0]	**XOMR2BASE**	**Base Address of XOM Region 2** XOMR2BASE must be page-aligned. **Note:** The page size is 4 Kbytes.

XOMR2SIZE (Address = 0x0F20_0024)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR2SIZE							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR2SIZE**	**Page Number of XOM Region 2** XOMR2SIZE must be page-aligned and less than 63 pages. The XOMR2SIZE minimum value is 1 page. If XOMR2SIZE is written to 0, the register value will be set to 1. **Note:** The page size is 4 Kbytes.

XOMR2CTRL (Address = 0x0F20_0028)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR2CTRL							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR2CTRL**	**Control of XOM Region 2** 0x5a = XOM region 2 is off. 0x50= XOM is in debug mode. The others = XOM region 2 is active. **Note 1:** But for Flash behavior,user cannot write 0 to 1.User need to check the lock value is effective to change 0x5A(0101_1010). **Note 2:** The active state has come into effect after power-on or reset cycle.

XOMR3BASE (Address = 0x0F20_0030)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
XOMR3BASE							
15	14	13	12	11	10	9	8
XOMR3BASE							
7	6	5	4	3	2	1	0
XOMR3BASE							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved.
[23:0]	**XOMR3BASE**	**Base Address of XOM Region 3** XOMR3BASE must be page-aligned. **Note:** The page size is 4 Kbytes.

XOMR3SIZE (Address = 0x0F20_0034)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR3SIZE							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR3SIZE**	**Page Number of XOM Region 3** XOMR3SIZE must be page-aligned and less than 63 pages. The XOMR3SIZE minimum value is 1 page. If XOMR3SIZE is written to 0, the register value will be set to 1. **Note:** The page size is 4 Kbytes.

XOMR3CTRL (Address = 0x0F20_0038)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
XOMR3CTRL							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved.
[7:0]	**XOMR3CTRL**	**Control of XOM Region 3** 0x5a = XOM region 3 is off. 0x50= XOM is in debug mode. The others = XOM region 3 is active. **Note 1:** But for Flash behavior,user cannot write 0 to 1.User need to check the lock value is effective to change 0x5A(0101_1010). **Note 2:** The active state has come into effect after power-on or reset cycle.

6.5.4.8 One Time Program Memory (OTP)

The One time program memory (OTP) is used to store the important information that are not allowed user to modify them twice. The OTP is 3 Kbytes with location address 0xF31_0000 ~ 0xF31_0BFF. The maximum size of OTP data are 2 Kbytes from 0xF31_0000 ~ 0xF31_07FF. Every 64 bits of OTP data has one 32 bits “LOCK BIT” from 0xF31_0800 to 0xF31_0BFF, as shown in Figure 6.5-5.

The purpose of “LOCK BIT” is for recording if the programed address had been locked (LOCK BIT≠0xFFFF_FFFF) or not (LOCK BIT= 0xFFFF_FFFF) in OTP. For examples, when LOCK BIT0 is not “0xFFFFFFFF”, OTP0 cannot be programmed again, regardless its content is all 1 or not. The “Flash page erase /mass erase command/ FLASH 64-bit Program/ FLASH Multi-Word Program” are never allowed to be executed in OTP. Before updating the content of OTP from 0xF31_0000 to 0xF31_07FF, user must check their “LOCK BIT” firstly. After finishing prograiming OTP data, user must write a non-0xFFFF_FFFF in the “LOCK BIT” of the above programed data to make sure that no one can modify them. OTP is only readable by ISP command and un-readable by CPU read directly.

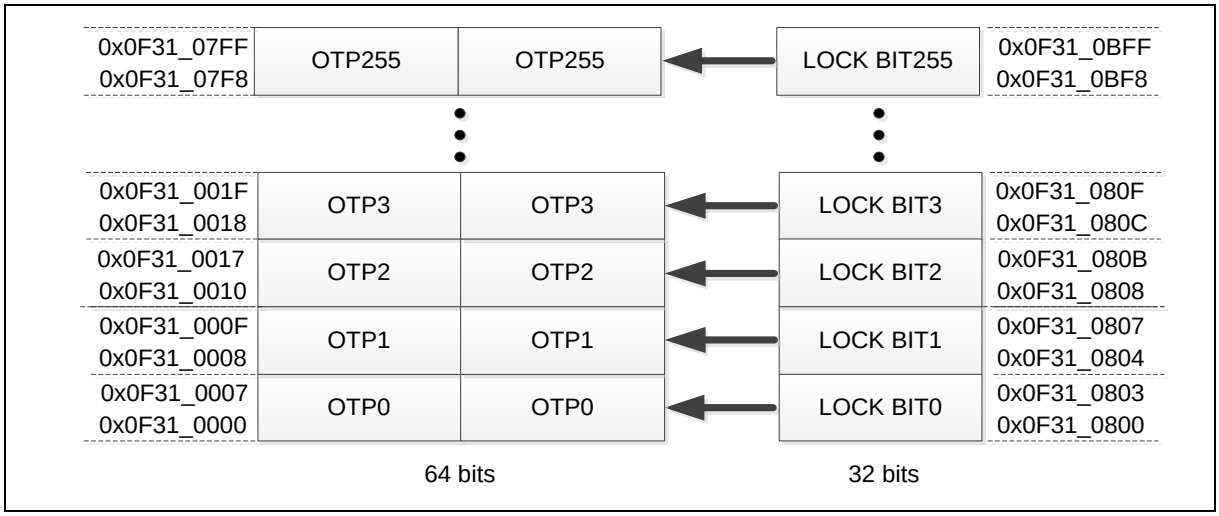


Figure 6.5-5 OTP Memory Map

6.5.4.9 Flash Memory Map

The Flash memory map is different from system memory map. The system memory map is used by CPU fetch code or data from FMC memory. The Flash memory map is used for ISP function to read, program or erase FMC memory. The Flash memory map is as Figure 6.5-6.

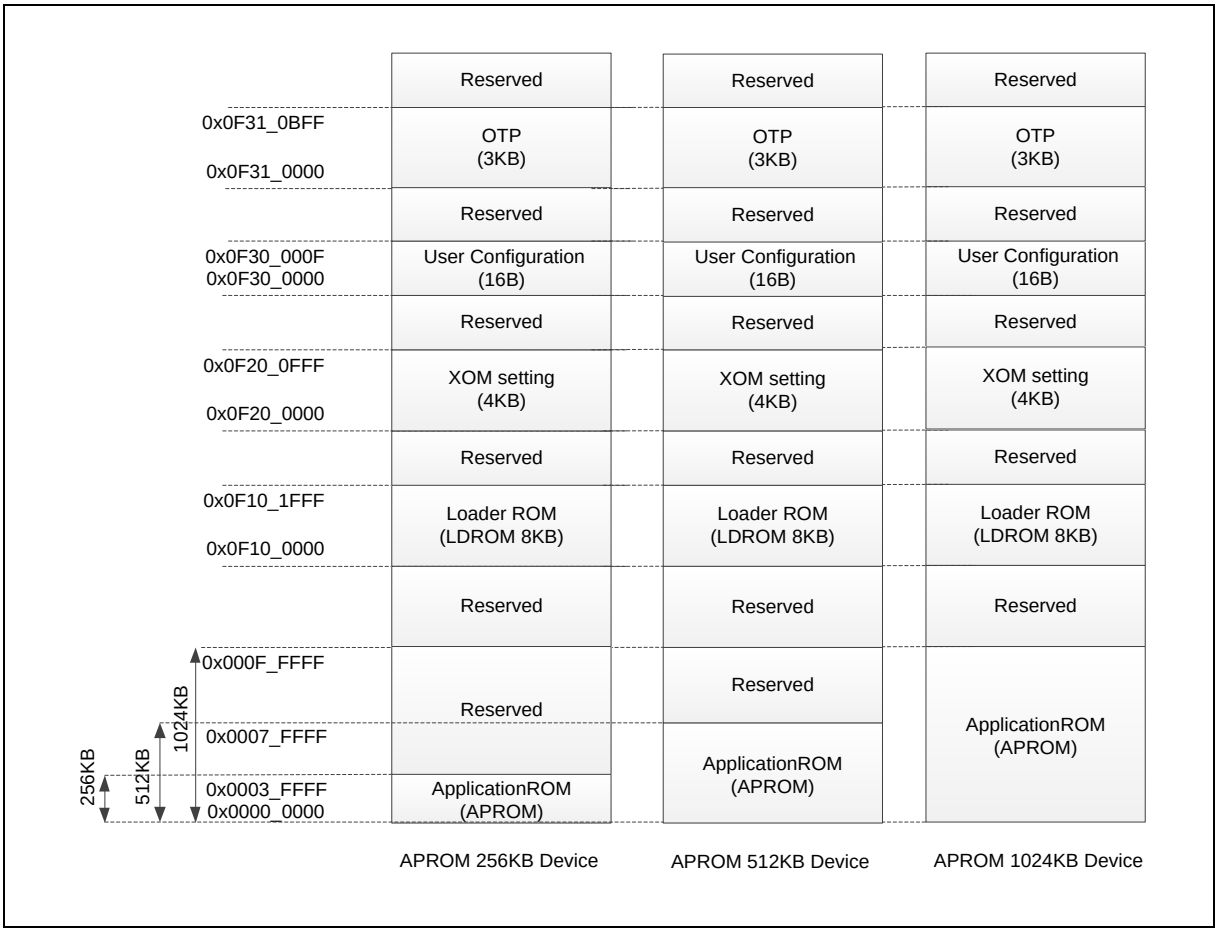


Figure 6.5-6 Flash Memory Map

6.5.4.10 System Memory Map with IAP Mode

The system memory map is used by CPU to fetch code or data from FMC memory. LDROM(0x0F10_0000~0x0F10_1FFF) address map are the same as in the Flash memory map. The Data Flash is shared with APROM and the Data Flash base address is defined by CONFIG1. The content of CONFIG1 is loaded into DFBA (Data Flash Base Address Register) at the Flash initiation.

The DFBA~(0x0003_FFFF/0x0007_FFFF/0x000F_FFFF) is the Data Flash region for Cortex-M4 data access, and 0x0000_0200~(DFBA-1) is APROM region for Cortex-M4 instruction access when APROM size is 256/512/1024 Kbytes, respectively.

The address from 0x0000_0000 to 0x0000_01FF is called system memory vector.

Figure 6.5-7 shows the system mamory map with IAP mode.

APROM, LDROM can map to the system memory vector for CPU start up. There are three kinds of system memory map with IAP mode when chip booting: (1) LDROM with IAP, (2) APROM with IAP,.

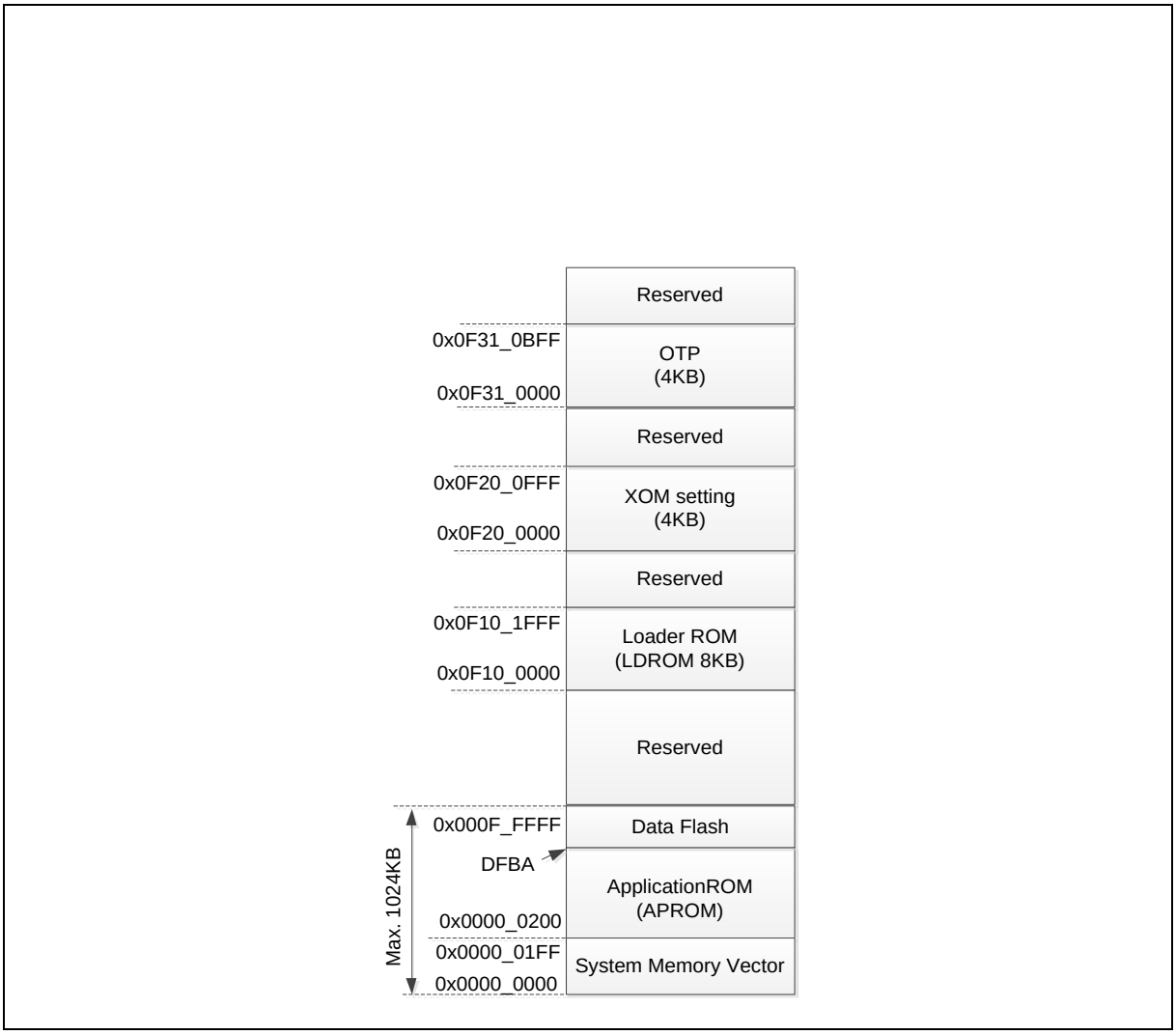


Figure 6.5-7 System Memory Map with IAP Mode

In LDROM with IAP mode, the LDROM (0x0F10_0000~0x0F10_01FF) is mapping to the system memory vector for Cortex-M4 instruction or data access. Figure 6.5-8 shows the memory map of LDROM with IAP mode.

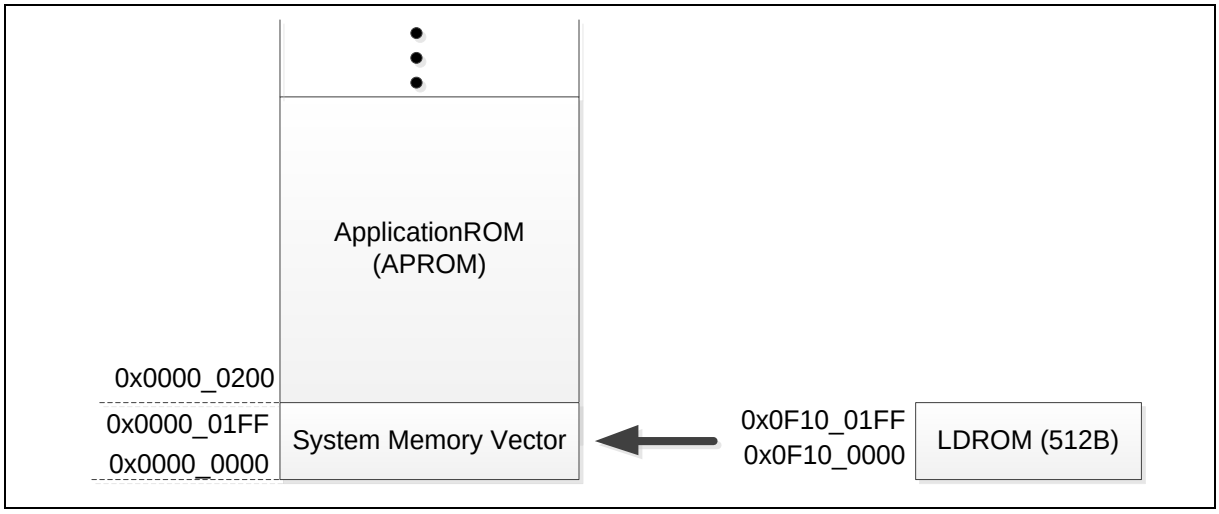


Figure 6.5-8 LDROM with IAP Mode

In APROM with IAP mode, the APROM (0x0000_0000~0x0000_01FF) is mapping to the system memory vector for Cortex-M4 instruction or data access. Figure 6.5-9 shows the memory map of APROM with IAP mode.

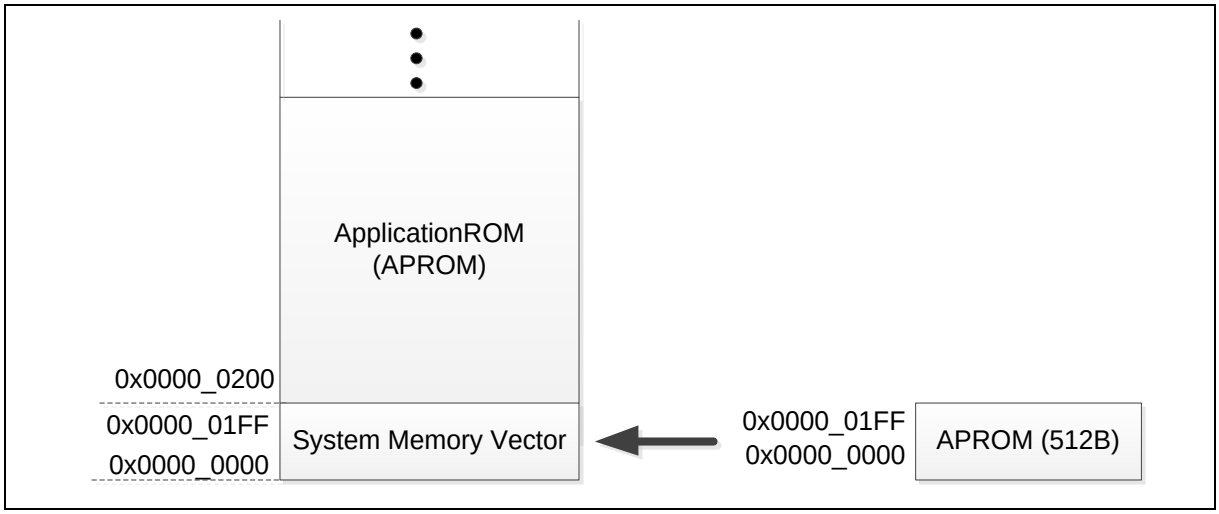


Figure 6.5-9 APROM with IAP Mode

In system memory map with IAP mode, APROM, LDROM, APROM can remap to the system memory vector when CPU running. User can write the target remap address to FMC_ISPADDR register and then trigger ISP procedure with the “Vector Remap” command (0x2E). In VECMAP (FMC_ISPSTS[23:9]), shows the finial system memory vector mapping address.

6.5.4.11 System Memory Map Without IAP Mode

There are two kinds of system memory map without IAP mode when chip booting: (1) LDROM without IAP, (2) APROM without IAP. In LDROM without IAP mode, LDROM base is mapping to 0x0000_0000. CPU program cannot run to access APROM. In APROM without IAP mode, APROM base is mapping to 0x0000_0000. CPU program cannot run to access LDROM. The Data Flash is shared with APROM and the Data Flash base address is defined by CONFIG1. The content of CONFIG1 is loaded into DFBA at the Flash initiation. The DFBA~(0x0003_FFFF/0x0007_FFFF/0x000F_FFFF) is the Data Flash region for Cortex-M4 data access, and 0x0000_0000~(DFBA-1) is APROM region for Cortex-M4 instruction access when APROM size is 256/512/1024 Kbytes, respectively. Figure 6.5-10 shows the system memory map without IAP mode.

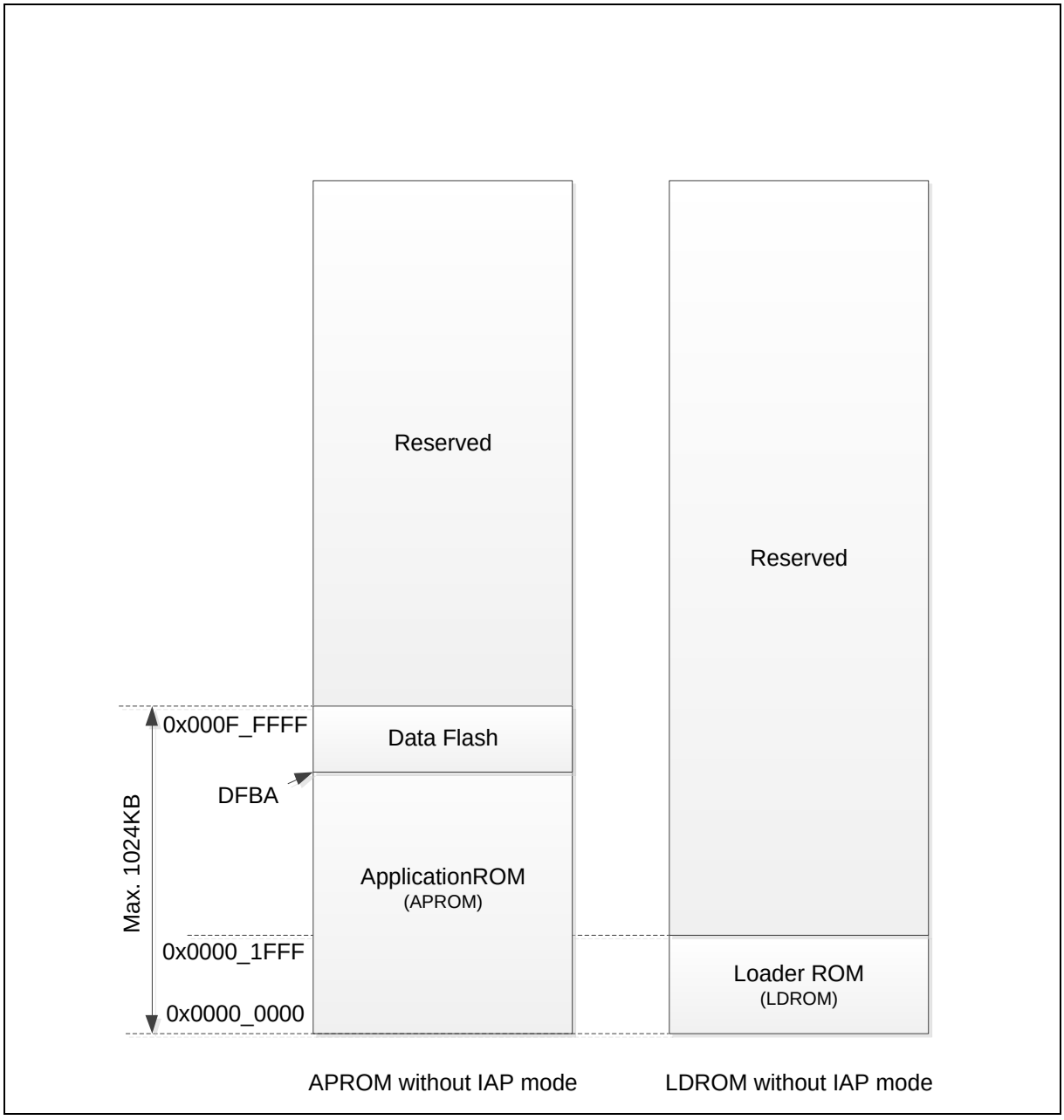


Figure 6.5-10 System Memory Map without IAP Mode

6.5.4.12 In-Application-Programming (IAP)

The M463/M467 series provides In-Application-Programming (IAP) function for user to switch the system memory vector code executing between APROM, LDROM and Boot Loader. User can enable the IAP function by booting chip and setting the chip boot selection bits in in CBS (CONFIG0[7:6]) as 10 or 00, in MBS (CONFIG0[5]) as 0. (If MSB(i.e., CONFIG0[5]) bit is set as 0 or perform secure boot, the CBS[0] (FMC_ISPSTS[1]) registers will show 0 and ignore CONFIG0[6] setting).

When chip boots with IAP function is enabled, any executable code (align to 512 bytes) is allowed to map to the system memory vector any time. User can change the remap address to FMC_ISPADDR and then trigger ISP procedure with the “Vector Remap” command.

6.5.4.13 In-System-Programming (ISP)

The M463/M467 series supports In-System-Programming (ISP) function allowing the embedded Flash memory to be reprogrammed under software control. ISP is performed without removing the microcontroller from the system through the firmware and on-chip connectivity interface, such as UART, USB, I^2C, SPI, and CAN (depended on chip feature). The target Flash memory space that ISP function operates cannot be across banks. Table 6.5-3 lists all ISP commands.

The ISP provides the following functions for embedded Flash memory.

- Supports Flash page erase function
- Supports Flash data program function
- Supports Flash data read function
- Supports company ID read function
- Supports device ID read function
- Supports unique ID read function
- Supports memory CRC32 checksum calculation function
- Supports Flash all one verification function
- Supports system memory vector remap function

ISP Commands

ISP Command	FMC_ISPCMD	FMC_ISPADDR	FMC_ISPDAT FMC_MPDAT0~FMC_MPDAT3
Flash Page Erase	0x22	Valid address of Flash memory organization. It must be page (4 Kbytes) alignment. Note that FMC_ISPADDR[11:0] will be ignored.	N/A
Flash Bank Erase	0x23	Valid address of APROM of the target bank. Note that FMC_ISPADDR[15:0] will be ignored.	N/A
Flash Block Erase	0x25	Valid address of APROM. It must be 4 pages (16 Kbytes) alignment. Note that FMC_ISPADDR[13:0] will be ignored.	N/A
Flash Mass Erase (This command is only valid while MERASE(CONFIG0[13]) bit = 0.)	0x26	0x0000_0000	N/A
Flash 32-bit Program	0x21	Valid address of Flash memory organization	FMC_ISPDAT :Programming Data
Flash 64-bit Program	0x61	Valid address of Flash memory organization	FMC_MPDAT0: LSB Programming Data FMC_MPDAT1: MSB Programming Data
Flash Multi-Word Program	0x27	Valid address of Flash memory organization in APROM and LDROM	FMC_MPDAT0: 1'st Programming Data FMC_MPDAT1: 2'nd Programming Data FMC_MPDAT2: 3'rd

			Programming Data FMC_MPDAT3: 4'th Programming Data
Flash Read	0x00	Valid address of Flash memory organization	FMC_ISPDAT: Return Data
Flash 64-bit Read	0x40	Valid address of Flash memory organization	FMC_ISPDAT: Return Data in FMC_ISPADDR FMC_MPDAT0: Return Data in FMC_ISPADDR FMC_MPDAT1: Return Data in FMC_ISPADDR+4
Read Company ID	0x0B	0x0000_0000	FMC_ISPDAT: 0x5300_00DA
Read Device ID	0x0C	0x0000_0000	FMC_ISPDAT: Return Device ID
Read CRC32 Checksum	0x0D	0x0000_0000	FMC_ISPDAT: Return Checksum
Run CRC32 Checksum Calculation	0x2D	Valid start address of memory organization It must be 4 Kbytes page alignment	FMC_ISPDAT: Size It must be 4 Kbytes alignment
Read Flash All One Result	0x08	Keep address of "Run Flash All One Verification"	FMC_ISPDAT: Return Result 0xA110_0000 : Flash is not all one 0xA11F_FFFF: Flash is all one.
Run Flash All One Verification	0x28	Valid start address of memory organization It must be 4 Kbytes page alignment	FMC_ISPDAT: Size It must be 4 Kbytes alignment
Read Unique ID	0x04	0x0000_0000	FMC_ISPDAT: Unique ID Word 0
		0x0000_0004	FMC_ISPDAT: Unique ID Word 1
		0x0000_0008	FMC_ISPDAT: Unique ID Word 2
Vector Remap	0x2E	Valid address in APROM,LDROM or boot loader It must be 512 bytes alignment	N/A

Table 6.5-3 ISP Command List

ISP Procedure

The FMC controller provides embedded Flash memory read, erase and program operation. Several control bits of FMC control register are write-protected, thus it is necessary to unlock before setting.

After unlocking the protected register bits, user needs to set the FMC_ISPCTL control register to decide to update LDROM, APROM or user configuration block, and then set ISPEN (FMC_ISPCTL[0]) to enable ISP function.

Once the FMC_ISPCTL register is set properly, user can set FMC_ISPCMD (refer to Table 6.5-3 ISP command list) for specify operation. Set FMC_ISPADDR for target Flash memory based on Flash memory organization. FMC_ISPDAT can be used to set the data to program or used to return the read data according to FMC_ISPCMD. The ISP procedure flow is shown in Figure 6.5-11.

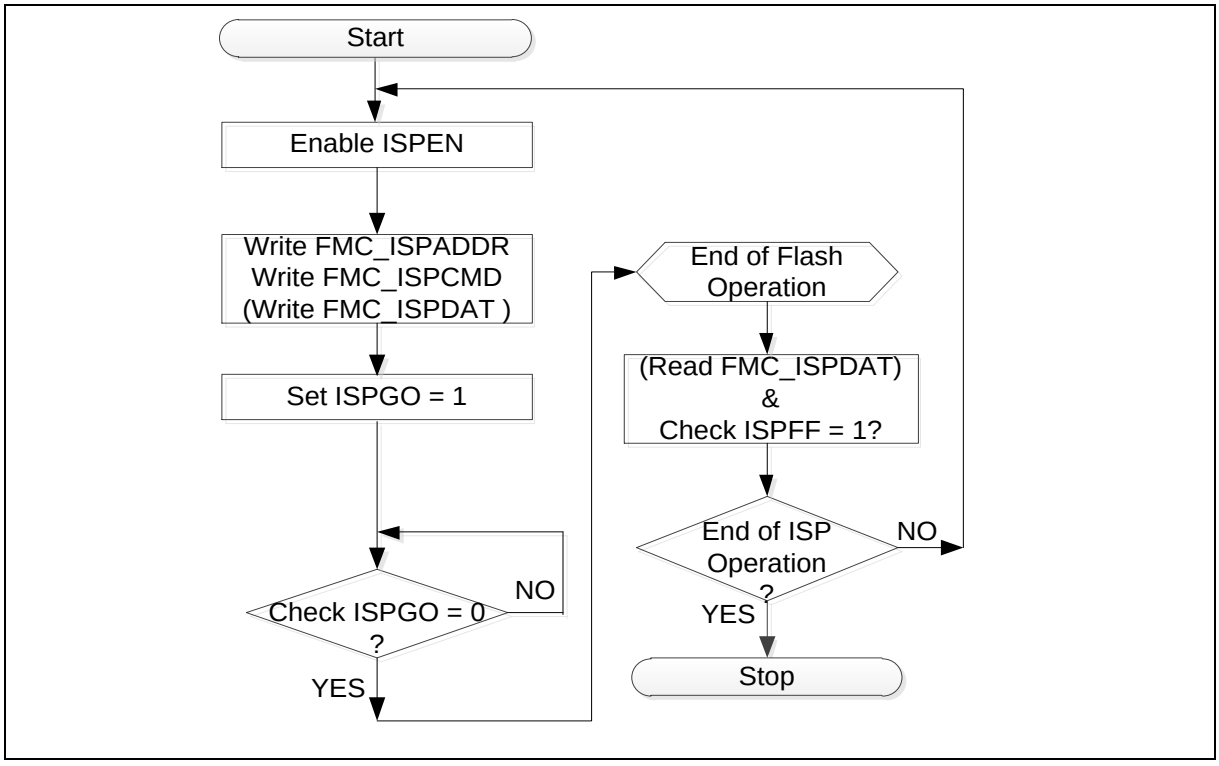


Figure 6.5-11 ISP Procedure Example

Finally, set the ISPGO (FMC_ISPTRG[0]) register to perform the relative ISP function. When ISP function is active, the ISPBUSY(FMC_ISPSTS[0]) and MPBUSY(FMC_MPSTS[0]) be set to 1. The ISPGO(FMC_ISPTRG[0]), ISPBUSY(FMC_ISPSTS[0]) and MPBUSY(FMC_MPSTS[0]) are self-cleared when ISP function has been done.

Several error conditions will be checked after ISP is completed. If an error condition occurs, ISP operation is not started and the ISP fail flag will be set instead. ISPFF(FMC_ISPSTS[6]) flag can only be cleared by software. The next ISP procedure can be started even ISPFF(FMC_ISPSTS[6]) bit is kept as 1. Therefore, it is recommended to check the ISPFF(FMC_ISPSTS[6]) bit and clear it after each ISP operation if it is set to 1.

When the ISPGO(FMC_ISPTRG[0]) bit is set and then CPU access the same bank, CPU will wait for ISP operation to finish during this period; the peripheral still keeps working as usual. If any interrupt request occurs, CPU will not service it till ISP operation is finished. When ISP operation is finished, the ISPGO bit will be cleared by hardware automatically. User can check whether ISP operation is finished or not by the ISPGO(FMC_ISPTRG[0]) bit.

When CPU access operation and ISP command are executed in different bank, CPU and ISP command can operate in parallel.

6.5.4.14 Embedded Flash Memory Programming

The M463/M467 series provides 32-bit, 64-bit and multi-word Flash memory programming function to speed up Flash updated procedure. Table 6.5-4 lists required FMC control registers in each embedded Flash programming function.

Register	Description	32-Bit Programming	64-Bit Programming	Multi-Word Programming
FMC_ISPCTL	ISP Control Register	√	√	√
FMC_ISPADDR	ISP Address Register	√	√	√

FMC_ISPDAT	ISP Data Register	√	N/A	N/A
FMC_ISPCMD	ISP Command Register	0x21	0x61	0x27
FMC_ISPTRG	ISP Trigger Register	√	√	√
FMC_ISPSTS	ISP Status Register	√	√	N/A
FMC_MPDAT0	ISP Data0 Register	N/A	√	√
FMC_MPDAT1	ISP Data1 Register	N/A	√	√
FMC_MPDAT2	ISP Data2 Register	N/A	N/A	√
FMC_MPDAT3	ISP Data3 Register	N/A	N/A	√
FMC_MPSTA	ISP Multi-Program status	N/A	N/A	√
FMC_MPADDR	ISP Multi-Program Address	N/A	N/A	√

Table 6.5-4 FMC Control Registers for Flash Programming

64-bit Programming

The M463/M467 series 64-bit programming function is faster than 32-bit programming. FMC_ISPDAT is used for 32-bit programming data register. In 64-bit programming, there are two programming data registers, one is FMC_MPDAT0 for LSB word, and the other is FMC_MPDAT1 for MSB word, and ISP command is 0x61, the other registers are the same as 32-bit programming. Figure 6.5-12 and Figure 6.5-13 are the ISP 32-bit / 64-bit programing procedure flow.

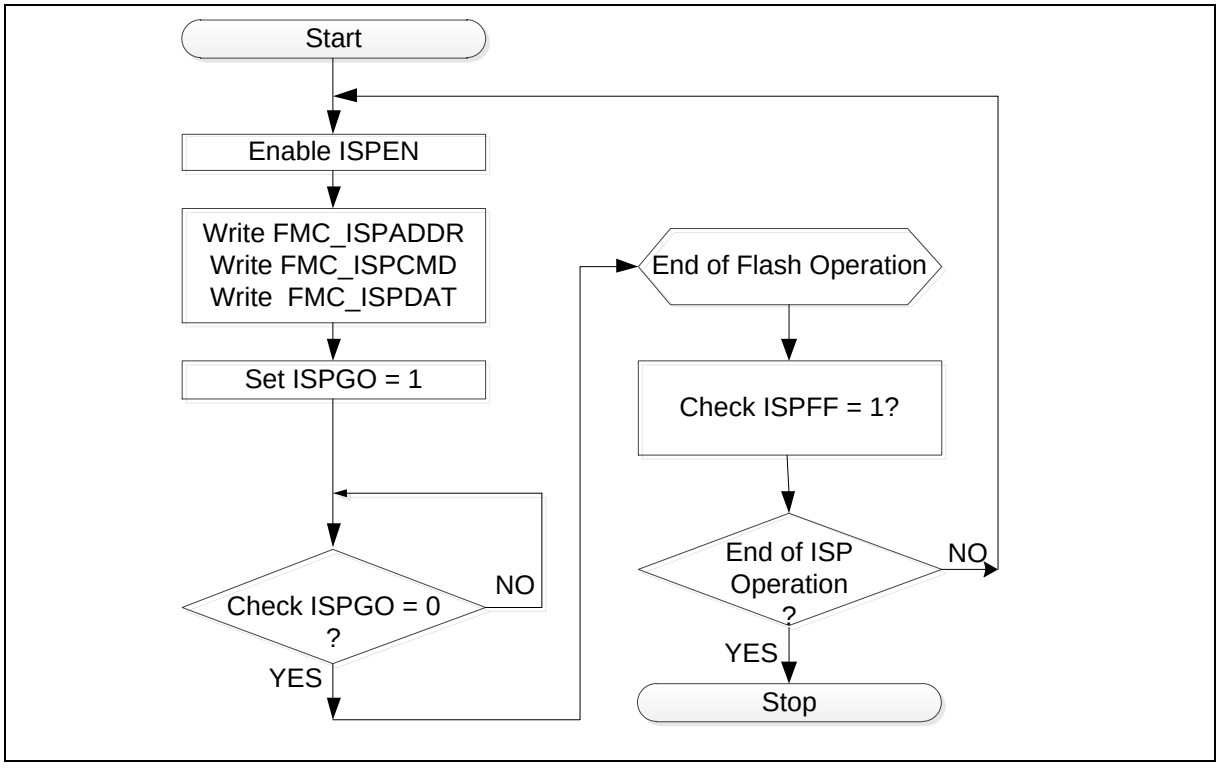


Figure 6.5-12 ISP 32-bit Programming Procedure

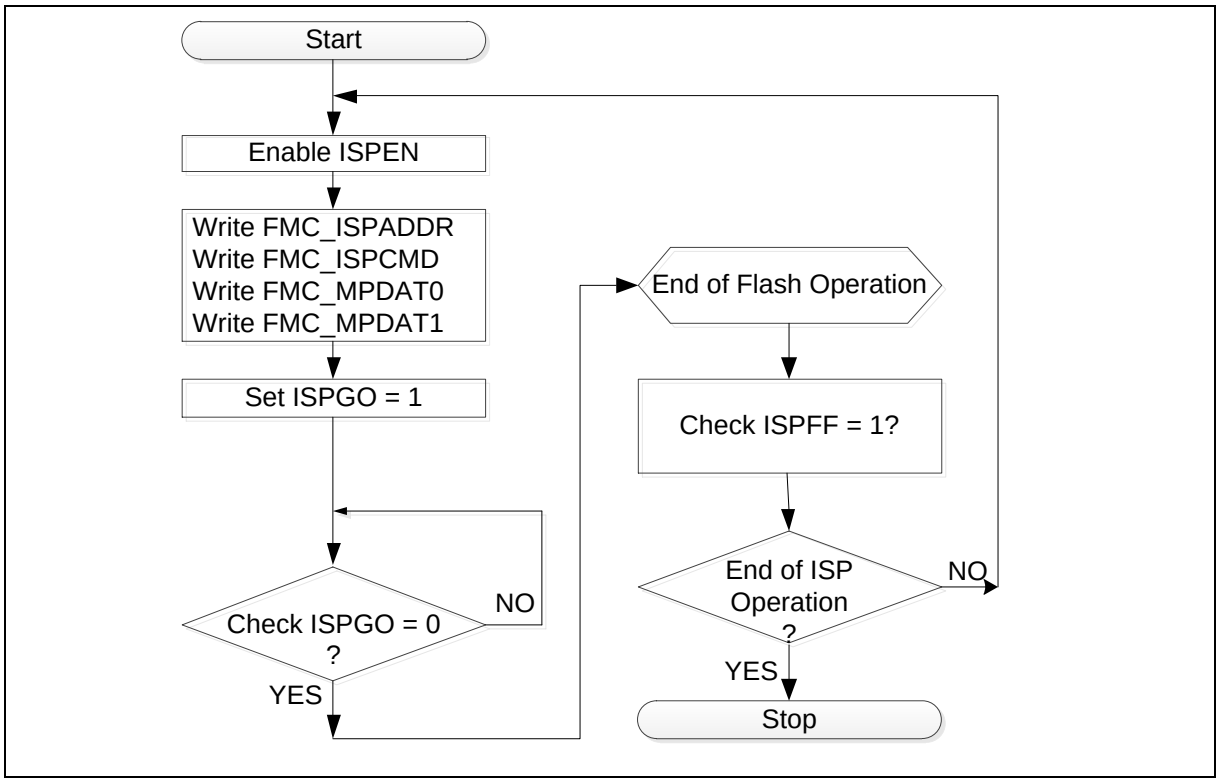


Figure 6.5-13 ISP 64-bit Programming Procedure

Multi-word Programming

The M463/M467 series supports multi-word programming function to speed up Flash updated procedure. The maximum programming length is up to 512 bytes, and the minimum programming length is 8 bytes (2 words). The multi-word programming is the fastest programming function if the programming words more than 8 bytes, because only one set of Flash setup time and hold time needed for one time operation. Figure 6.5-14 compares each programming time.

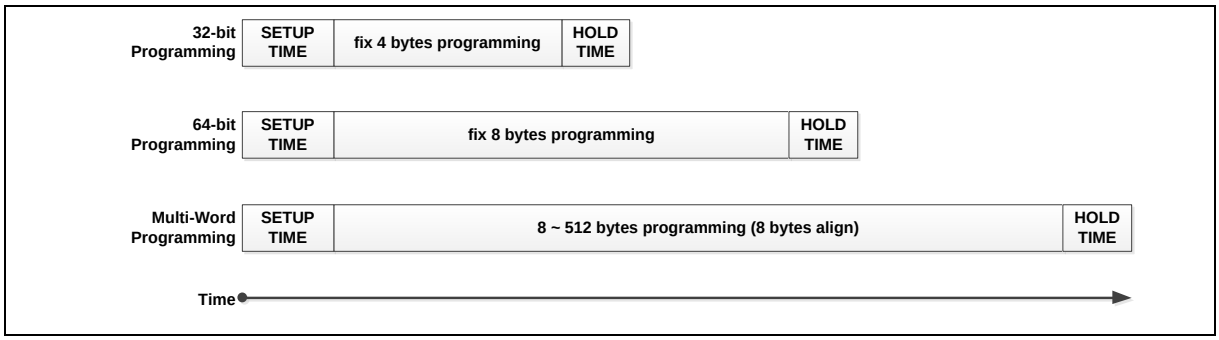


Figure 6.5-14 Multi-word Programming Time

In multi-word programming operation, Cortex-M4 CPU has to monitor the empty status of the programming buffer. CPU has to prepare the next data for programming continuity. The multi-program firmware should not be located in APROM or LDROM, because CPU instruction fetch cannot be hold. The firmware has to be located in Boot loader or embedded SRAM of chip to avoid CPU hold.

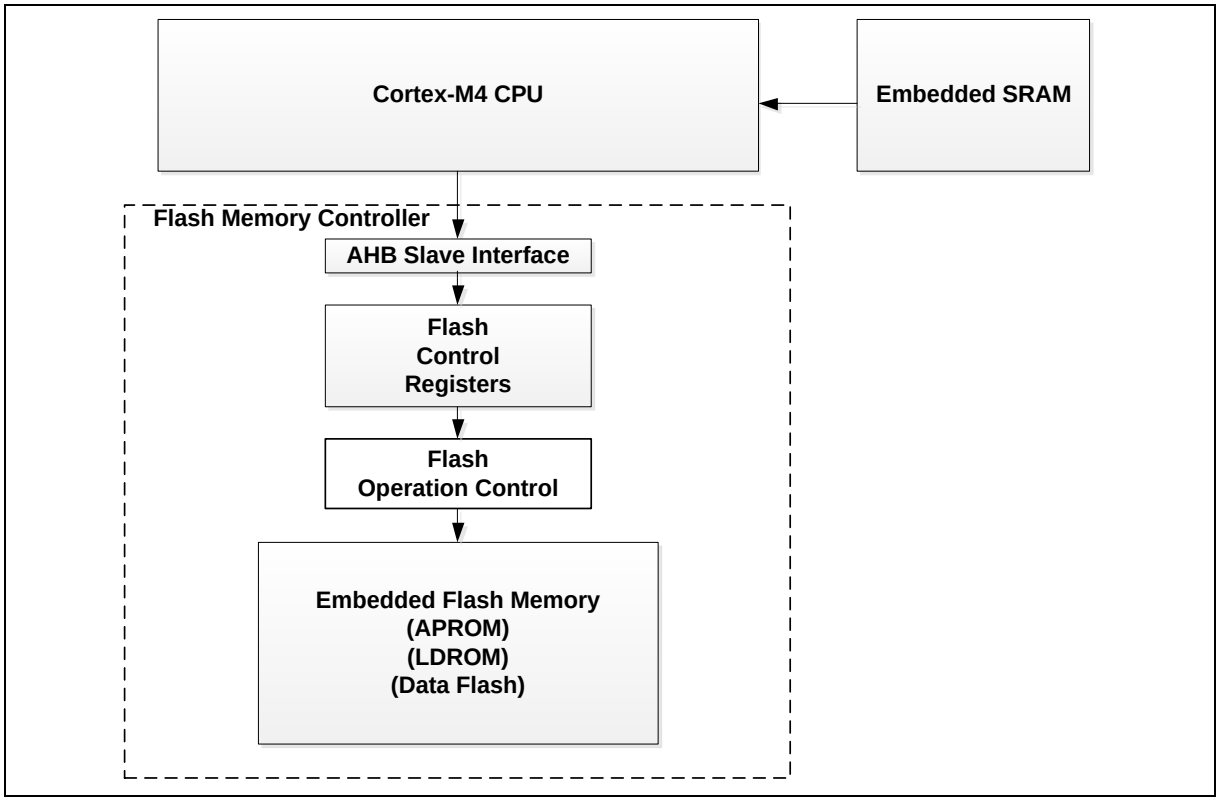


Figure 6.5-15 Firmware in SRAM for Multi-word Programming

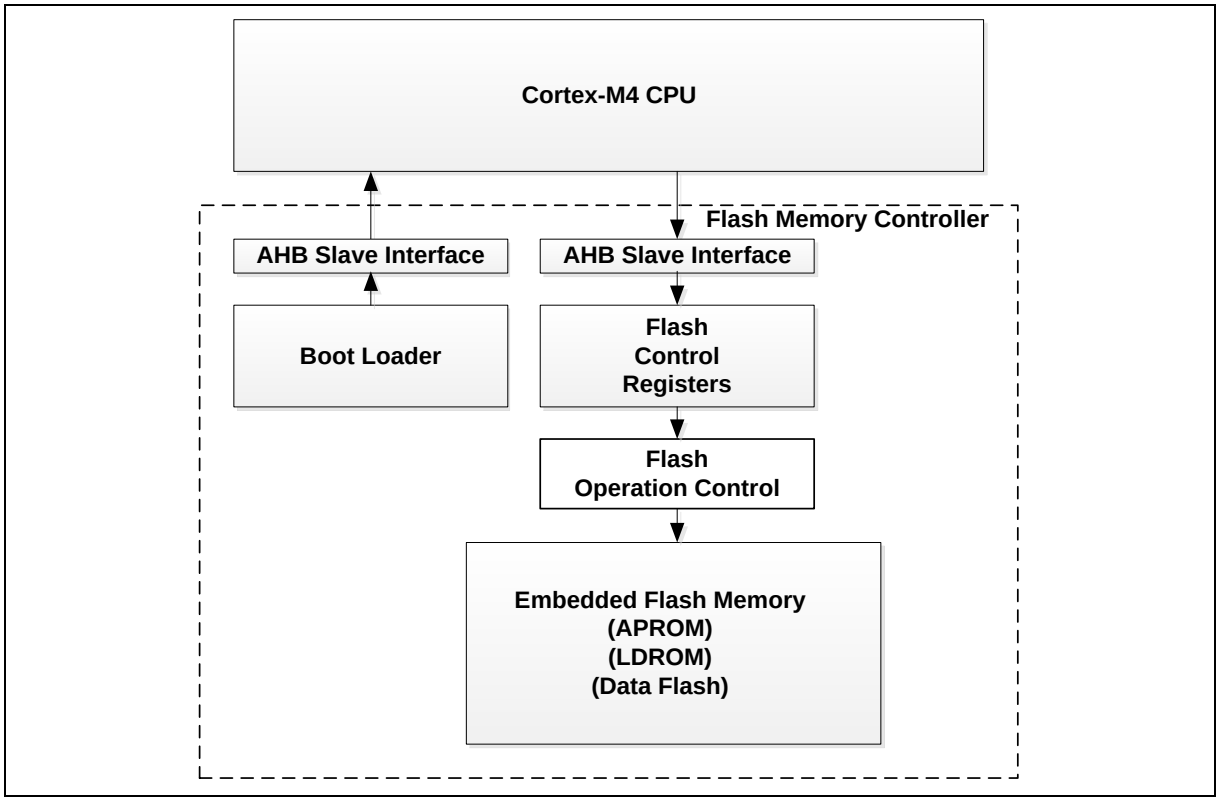


Figure 6.5-16 Firmware in Boot Loader for Multi-word Programming

The multi-word programming flow is shown in Figure 6.5-17. The starting ISP address (FMC_ISPADDR) has to be 8-byte align, FMC_ISPADDR[2:0] should be 0. ISPDAT0(FMC_MPDAT0) is the data word of the offset 0x0, ISPDAT1(FMC_MPDAT1) is the second word (offset 0x4), ISPDAT2(FMC_MPDAT2) is the third word (offset 0x8), and ISPDAT3(FMC_MPDAT3) is forth word (offset 0xC). If the starting ISP address FMC_ISPADDR [3] is 0, the 1st data word should put on ISPDAT0, and 2nd word is ISPDAT1, 3rd word is ISPDAT2, and 4th word is ISPDAT3. If the starting ISP address FMC_ISPADDR [3] is 1, the 1st data word should put on ISPDAT2, and 2nd word is ISPDAT3, 3rd word is ISPDAT0, and 4th word is ISPDAT1. The maximum programming size is 512 bytes and align to 512-byte address. While FMC controller performs multi-word programming operation, CPU needs to monitor the buffer status D3~D0(FMC_MPSTS[7:4]) and MPBUSY (FMC_MPSTS[0]) to wait the buffer empty ((D1,D0)=00, or (D3,D2)=00), and then CPU needs to update the next programming data (ISPDAT0, ISPDAT1, ISPDAT2 and ISPDAT3) in time. Otherwise, FMC controller will exit multi-word programming operation (MPBUSY (FMC_MPSTS[0]) = 0). If CPU cannot update the data in time (MPBUSY (FMC_MPSTS[0]) =0), CPU needs restart a new multi-word programming procedure to continue, FMC_MPADDR provides the last program address information. At the end of operation, CPU has to check ISPFF (FMC_MPSTS[2]) to confirm the multi-word operation successful complete.

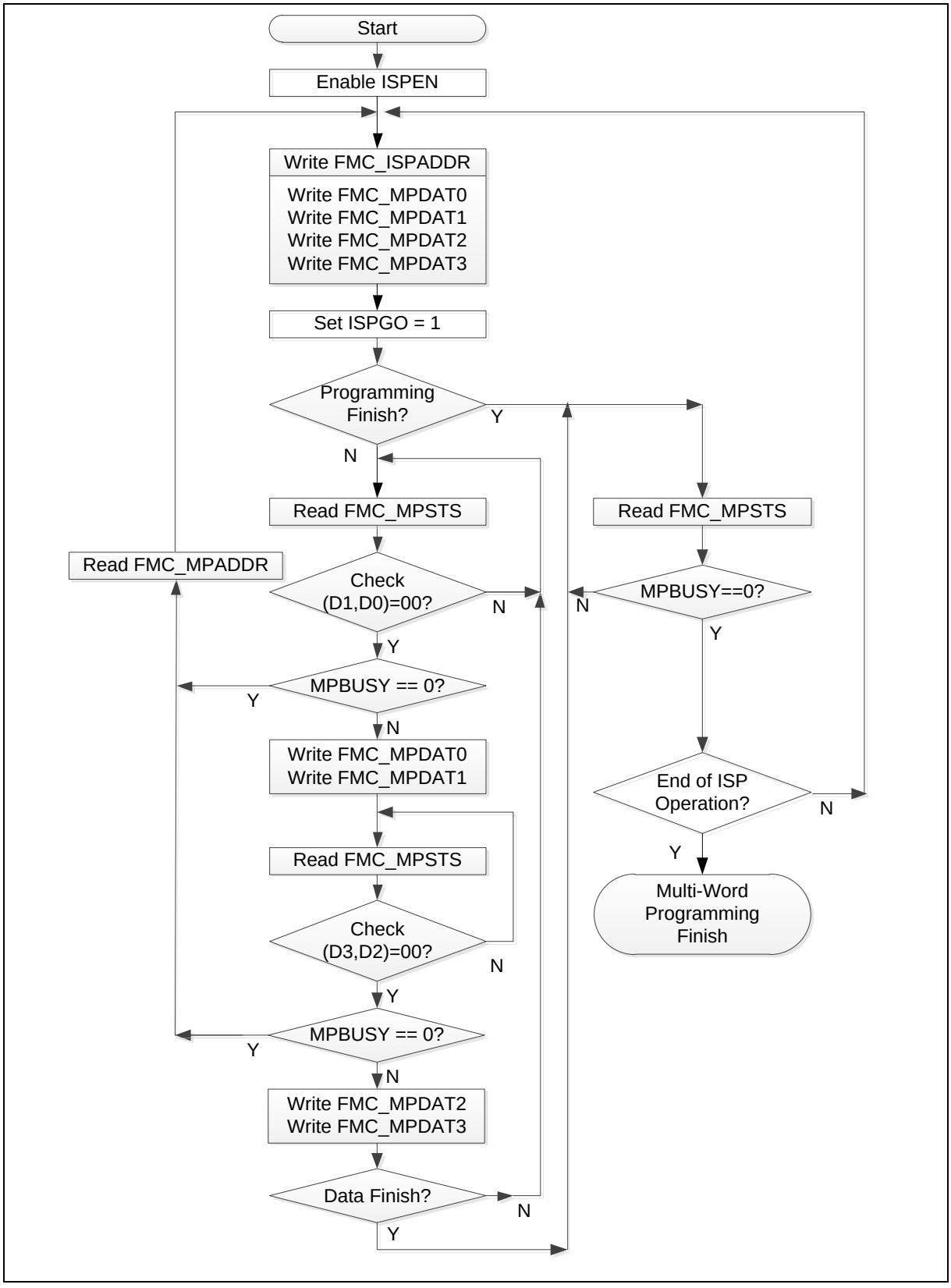


Figure 6.5-17 Multi-word Programming Flow

6.5.4.15 Fast Flash Programming Verification

In traditional Flash programming operation, the controller receives the programming trigger event then control the timing to perform the programming embedded Flash memory as show in Figure 6.5-18.

The M463/M467 series supports the fast Flash programming verification function, which provides hardware verification for Flash programming to save time of the CPU read back and comparison. When data is programmed to the embedded Flash memory, the controller asserts the Flash read operation to read data out, and performs data comparison with data in. Finally, the comparison result is saved in PGFF (FMC_ISPSTS[5]). The PGFF is set to 1 if output data is not the same as the input programming data. The flag is kept until clear by software or a new erase operation. The fast Flash programming verification flow is shown in Figure 6.5-18.

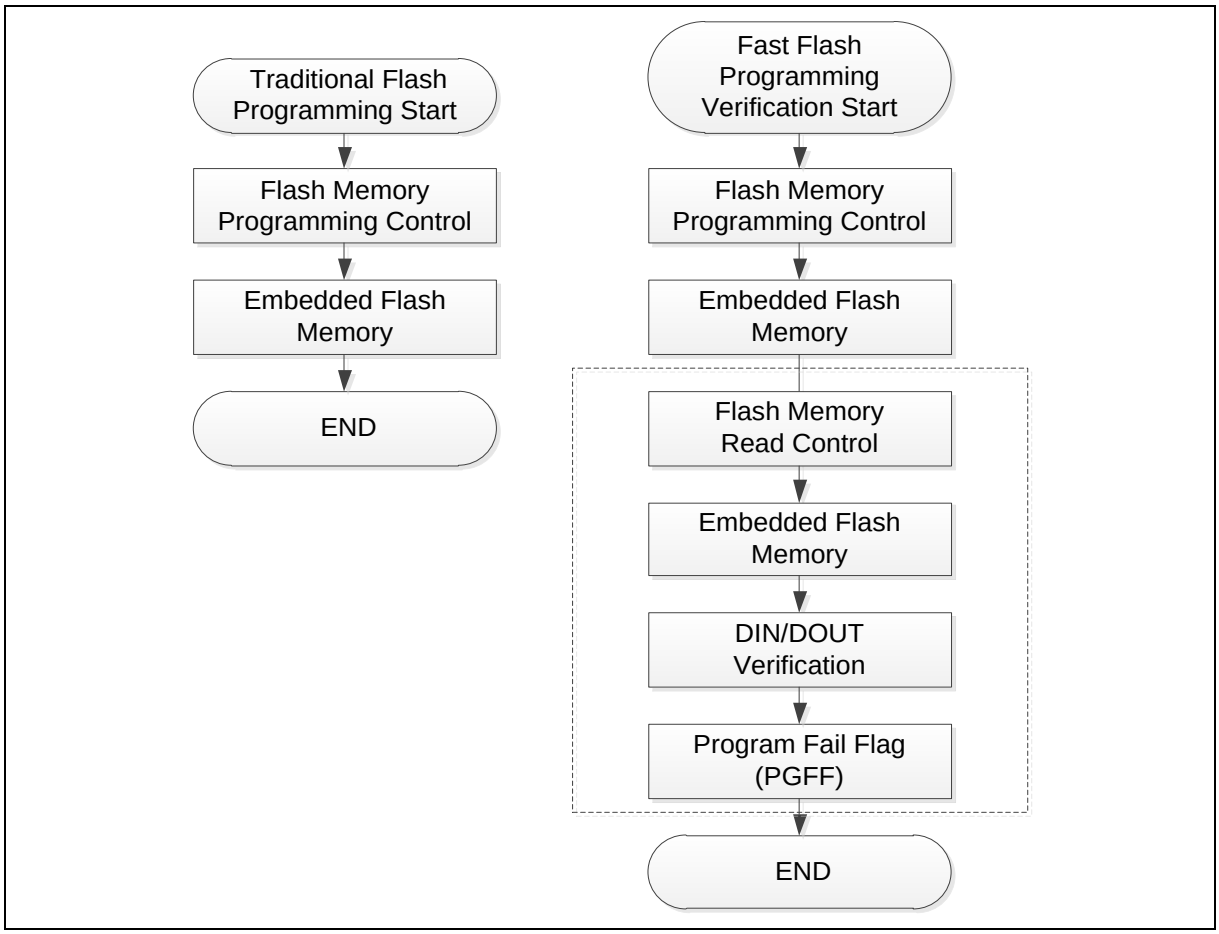


Figure 6.5-18 Fast Flash Programming Verification Flow

In traditional Flash updated operation, the Flash memory has to perform three steps to complete the Flash memory updated procedure, (1) Flash ERASE, (2) Flash PROGRAM, and (3) Flash READ back all of data to check the correction. In the M463/M467 series, it only reads FMC_ISPSTS to check PGFF flag in Step (3) without reading data back to confirm. Figure 6.5-19 compares traditional programming verification flow and fast programming verification flow.

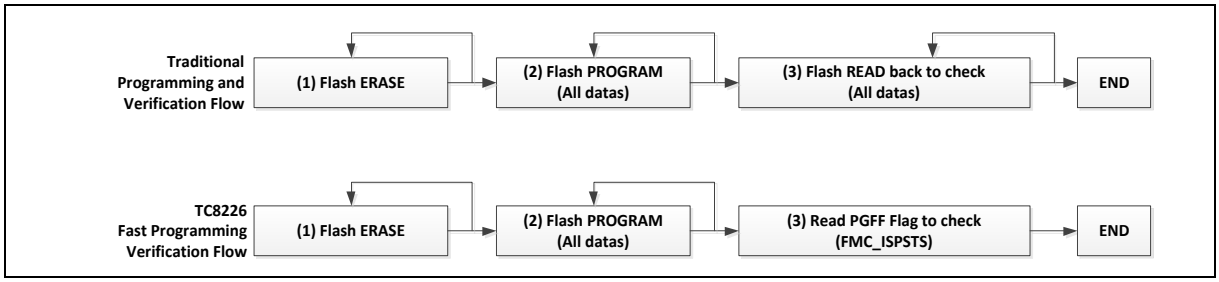


Figure 6.5-19 Verification Flow

The fast Flash programming verification function is released for 32-bit programming and 64-bit programming operation, but multi-word programming operation is not suitable due to the embedded Flash HV (High Voltage) of continue programming.

6.5.4.16 CRC32 Checksum Calculation

The M463/M467 series supports the CRC32 checksum calculation function to help user quickly check the memory content includes APROM, LDROM and Boot Loader. The CRC32 polynomial is

CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

With seed = 0xFFFF_FFFF

The CRC32 checksum calculation flow is shown in Figure 6.5-20.

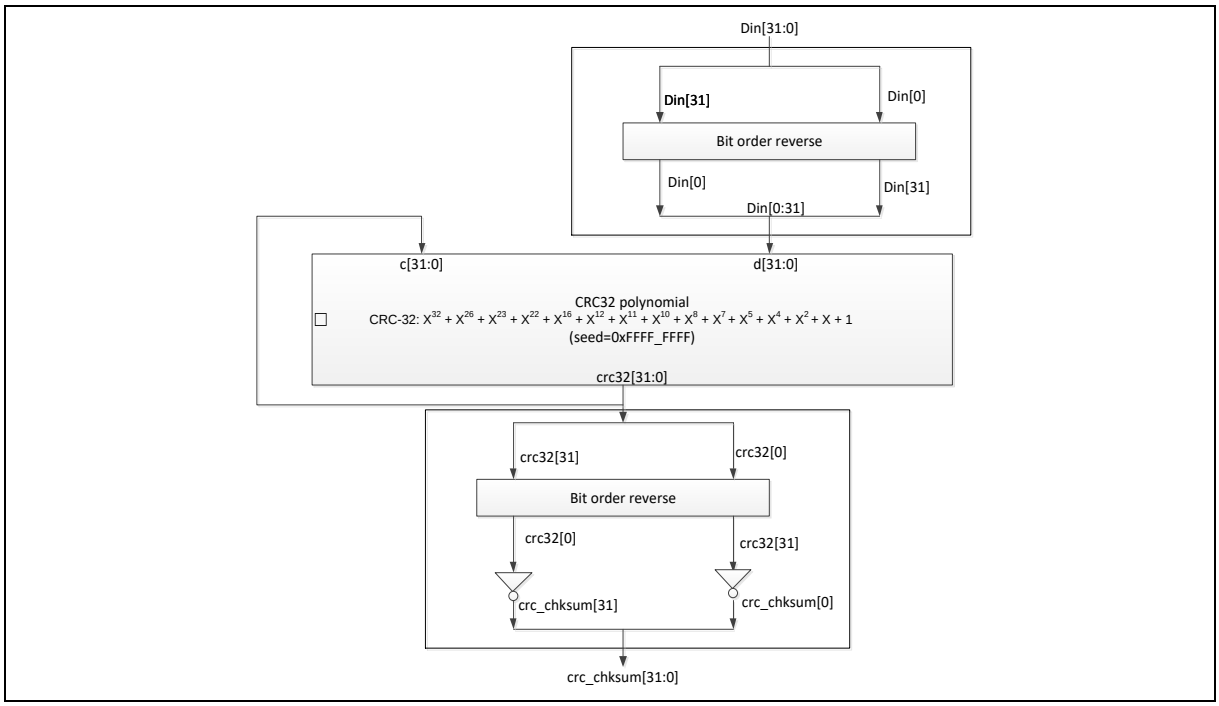


Figure 6.5-20 Flash CRC32 Checksum Calculation

Three steps complete this CRC32 checksum calculation.

1. Step 1.Perform ISP “Run Memory CRC32 Checksum” operation
2. Step 2.Perform ISP “Read Memory CRC32 Checksum” operation
3. Step 3..Read FMC_ISPDAT to get checksum.

In Step 1, user has to set the memory starting address (FMC_ISPADDR) and size (FMC_ISPDAT) to calculate. Both address and size have to be 4 Kbytes alignment, the size should be ≥ 4 Kbytes and the starting address includes APROM, LDROM and Boot Loader.

In Step 2, the FMC_ISPADDR should be kept as the same as Step 1.

In Step 3, the checksum is read from FMC_ISPDAT. If the checksum is 0x0000_0000, there is one of two conditions (1) Checksum calculation is in-progress, (2) Address and size is over device limitation

Note that the range of CRC32 checksum cannot cross bank in one operation.

6.5.4.17 Flash All One Verification

The M463/M467 series supports the Flash all one verification function to help user quickly check a memory block content blanking for APROM and LDROM after Flash erase operation.

Two or Three steps complete this Flash all one verification.

Two-step flow:

1. Step 1.Perform ISP “Run Flash All One Verification” operation
2. Step 2..Read ALLONE(FMC_ISPSTA[7])bit to get the verification result

   ALLONE : 1, all of Flash bits are 1 in verification block memory.

   ALLONE : 0, Flash bits are not all 1 in verification block memory.

Three-step flow:

1. Step 1.Perform ISP “Run Flash All One Verification” operation
2. Step 2.Perform ISP “Read Flash All One Result” operation
3. Step 3..Read FMC_ISPDAT to get the verification result.

FMC_ISPDATA : 0xA11F_FFFF, all of Flash bits are 1 in verification block memory.

FMC_ISPDATA : 0xA110_0000, Flash bits are not all 1 in verification block memory

In Step 1, user has to set the memory starting address (FMC_ISPADDR) and size (FMC_ISPDAT) to verify. Both address and size have to be 4 Kbytes alignment, the size should be ≥ 4 Kbytes and the starting address includes APROM and LDROM.

In Step 2, the FMC_ISPADDR should be kept as the same as Step 1.

Note that the range of “Run All One Verification” cannot cross bank in one operation.

6.5.4.18 APROM Function Safety Protect

FMC supports safety protect function in APROM. User can use the function to avoid user code doing accidental operation. To use the function, user needs to write the enable bit at register FMC_APPROT. Each FMC_APPROT bit protects 32 Kbytes memory.The FMC_APPROT[0] protect mermory from 0x0 to 0x7fff, FMC_APPROT[1] protects mermory from 0x8000 to 0xffff and so on. If user wants to protect 0x5000~0x9000, user needs to enable FMC_APPROT[1:0]. When APROM region is protected, user cannot do any program or page erase at protect region. But user still can do mass erase, bank erase and XOM page erase.

6.5.4.19 Lock Effect Tables

The M463/M467 series supports four kinds of protections include the Security Lock Control (i.e., LOCK in CONFIG0[1] and ALOCK in CONFIG2[7:0]). In this section, the M463/M467 series prepares some lock effect tables for user to understand the lock effects on APROM, LDROM, CONFIG and others with above four protections for CPU, ICE and ICP/Writer.

LOCK/ALOCK	**OFF**		**ON**		**OFF**		**OFF**	
KSPLOCK	OFF		OFF		ON		OFF	
APROM	R	W	R	W	R	W	R	W
Data Flash	R	W	R	W	R	W	R	W
LDROM	R	W	R	W	R	W	R	W
CFG	R	W	R	W	R	W	R	W
OTP	R	W_2	R	W_2	R	W_2	R	W_2
Secure Boot Key	-	W	-	W	-	W	-	W
Boot Loader	R	-	R	-	R	-	R	-
DID/UID/Checksum	R	-	R	-	R	-	R	-

Table 6.5-5 Lock Effect Table with Four Protections for CPU

LOCK/ALOCK	**OFF**		**ON**		**OFF**		**OFF**	
KSPBLOCK	OFF		OFF		ON		OFF	
APROM	R	W	-	-	-	-	R	W
Data Flash	R	W	-	-	-	-	R	W
LDROM	R	W	-	-	-	-	R	W
CFG	R	W	R	-	R	-	R	W
OTP	R	W_2	-	-	-	-	R	W_2
Secure Boot Key	-	W	-	-	-	-	-	W
Boot Loader	-	-	-	-	-	-	-	-
DID/UID/Checksum	R	-	R	-	R	-	R	-

Table 6.5-6 Lock Effect Table with Four Protections for ICE/ ICP/Writer

Note:

1. The symbol “R” means readable and the symbol “W” means writable.
2. The symbol “X” means no matter what value is.
3. The symbol “W_2” means OTP is writable only when the corresponding “LOCK BIT” is 0xFFFF_FFFF.
4. MassErase (i.e. whole chip erase) can be executed any time and any address, except for OTP.
5. When any protection conflicts with another, FMC will adopt the most stringent one.
6. Boot loader is un-readable for ICE and ICP/Writer.

6.5.5 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
FMC Base Address **FMC_BA = 0x4000_C000**				
FMC_ISPCTL	FMC_BA+0x00	R/W	ISP Control Register	0x0000_0000
FMC_ISPADDR	FMC_BA+0x04	R/W	ISP Address Register	0x0000_0000
FMC_ISPDAT	FMC_BA+0x08	R/W	ISP Data Register	0x0000_0000
FMC_ISPCMD	FMC_BA+0x0C	R/W	ISP Command Register	0x0000_0000
FMC_ISPTRG	FMC_BA+0x10	R/W	ISP Trigger Control Register	0x0000_0000
FMC_DFBA	FMC_BA+0x14	R	Data Flash Base Address	0xXXXX_XXXX
FMC_FTCTL	FMC_BA+0x18	R/W	Flash Access Time Control Register	0x0000_0000
FMC_ISPSTS	FMC_BA+0x40	R/W	ISP Status Register	0x0000_0000
FMC_CYCCTL	FMC_BA+0x4C	R/W	Flash Access Cycle Control Register	0x0000_0001
FMC_MPDAT0	FMC_BA+0x80	R/W	ISP Data0 Register	0x0000_0000
FMC_MPDAT1	FMC_BA+0x84	R/W	ISP Data1 Register	0x0000_0000
FMC_MPDAT2	FMC_BA+0x88	R/W	ISP Data2 Register	0x0000_0000
FMC_MPDAT3	FMC_BA+0x8C	R/W	ISP Data3 Register	0x0000_0000
FMC_MPSTS	FMC_BA+0xC0	R	ISP Multi-program Status Register	0x0000_0000
FMC_MPADDR	FMC_BA+0xC4	R	ISP Multi-program Address Register	0x0000_0000
FMC_XOMR0STS	FMC_BA+0xD0	R	XOM Region 0 Status Register	0x0FF0_00FF
FMC_XOMR1STS	FMC_BA+0xD4	R	XOM Region 1 Status Register	0x0FF0_00FF
FMC_XOMR2STS	FMC_BA+0xD8	R	XOM Region 2 Status Register	0x0FF0_00FF
FMC_XOMR3STS	FMC_BA+0xDC	R	XOM Region 3 Status Register	0x0FF0_00FF
FMC_XOMSTS	FMC_BA+0xE0	R	XOM Status Register	0x0000_0000
FMC_APPROT	FMC_BA+0x110	R/W	APROM Protect Register	0x0000_0000

6.5.6 Register Description

ISP Control Register (FMC_ISPCTL)

Register	Offset	R/W	Description	Reset Value
FMC_ISPCTL	FMC_BA+0x00	R/W	ISP Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							**INTEN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**BL**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**ISPFF**	**LDUEN**	**CFGUEN**	**APUEN**	**Reserved**		**ISPEN**

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**INTEN**	**Secure ISP INT Enable Bit (Write Protect)** 0= ISP INT Disabled. 1= ISP INT Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register. Before using INT, user needs to clear the INTFLAG(FMC_ISPSTS[24]) make sure INT happen at correct time.
[23:17]	**Reserved**	Reserved.
[16]	**BL**	**Boot Loader Booting (Write Protect)** This bit is initiated with the inversed value of MBS (CONFIG0[5]). Any reset, except CPU reset (CPU is 1) or system reset (SYS), BL will be reloaded. This bit is used to check chip boot from Boot Loader or not. User should keep original value of this bit when updating FMC_ISPCTL register. 0 = Boot from APROM or LDROM. 1 = Boot from Boot Loader. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[15:8]	**Reserved**	Reserved.
[7]	**Reserved**	Reserved.

[6]	**ISPFF**	**ISP Fail Flag (Write Protect)** This bit is set by hardware when a triggered ISP meets any of the following conditions: This bit needs to be cleared by writing 1 to it. ● APROM writes to itself if APUEN is set to 0. ● LDROM writes to itself if LDUEN is set to 0. ● CONFIG is erased/programmed if CFGUEN is set to 0. ● Erase or Program command at brown-out detected ● Destination address is illegal, such as over an available range. ● Invalid ISP commands ● Page Erase command at LOCK mode with ICE connection ● Invalid operations (except for chip erase) with ICE connection if KSPLOCK is not 0x5A ● Read any content of boot loader with ICE connection ● The base and size of new XOM regsions is wrong, overlap or writed twice ● The input setting of XOM page erase function is wrong ● The active XOM regsion is accessed (except for chip erase, page erase, chksum and read CID/DID) ● The XOM setting page is accessed (except for chip erase, word program and read) ● Violate the load code read protection ● OTP is erased, multi-word programed or 64-bit read ● Checksum or Flash All One Verification is not executed in their valid range ● Block erase or bank erase is not executed in APROM **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**LDUEN**	**LDROM Update Enable Bit (Write Protect)** 0 = LDROM cannot be updated. 1 = LDROM can be updated. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**CFGUEN**	**CONFIG Update Enable Bit (Write Protect)** 0 = CONFIG cannot be updated. 1 = CONFIG can be updated. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**APUEN**	**APROM Update Enable Bit (Write Protect)** 0 = APROM cannot be updated when the chip runs in APROM. 1 = APROM can be updated when the chip runs in APROM. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2:1]	**Reserved**	Reserved.
[0]	**ISPEN**	**ISP Enable Bit (Write Protect)** ISP function enable bit. Set this bit to enable ISP function. 0 = ISP function Disabled. 1 = ISP function Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

ISP Address (FMC_ISPADDR)

Register	Offset	R/W	Description	Reset Value
FMC_ISPADDR	FMC_BA+0x04	R/W	ISP Address Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPADDR							

Bits	Description	
[31:0]	**ISPADDR**	**ISP Address** The M463/M467 series is equipped with embedded Flash. ISPADDR[1:0] must be kept 00 for ISP 32-bit operation. ISPADDR[2:0] must be kept 000 for ISP 64-bit operation. For CRC32 Checksum Calculation command, this field is the Flash starting address for checksum calculation, 4 Kbytes alignment is necessary for CRC32 checksum calculation. For Flash32-bit Program, ISP address needs word alignment (4-byte). For Flash 64-bit Program, ISP address needs double word alignment (8-byte).

ISP Data Register (FMC_ISPDAT)

Register	Offset	R/W	Description	Reset Value
FMC_ISPDAT	FMC_BA+0x08	R/W	ISP Data Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT							

Bits	Description	
[31:0]	**ISPDAT**	**ISP Data** Write data to this register before ISP program operation. Read data from this register after ISP read operation. When ISPFF (FMC_ISPCTL[6]) is 1, ISPDAT = 0xffff_ffff. For Run CRC32 Checksum Calculation command, ISPDAT is the memory size (byte) and 4 Kbytes alignment. For ISP Read CRC32 Checksum command, ISPDAT is the checksum result. If ISPDAT = 0x0000_0000, it means that (1) the checksum calculation is in progress, or (2) the memory range for checksum calculation is incorrect.

ISP Command Register (FMC_ISPCMD)

Register	Offset	R/W	Description	Reset Value
FMC_ISPCMD	FMC_BA+0x0C	R/W	ISP Command Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**CMD**						

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:0]	**CMD**	**ISP Command** ISP command table is shown below: 0x00= FLASH Read. 0x04= Read Unique ID. 0x08= Read Flash All-One Result. 0x0B= Read Company ID. 0x0C= Read Device ID. 0x0D= Read Checksum. 0x21= FLASH 32-bit Program. 0x22= FLASH Page Erase. Erase any page in two banks, except for OTP. 0x23= FLASH Bank Erase. Erase all pages of APROM in BANK0 or BANK1. 0x25= FLASH Block Erase. Erase four pages alignment of APROM in BANK0 or BANK1. 0x27= FLASH Multi-Word Program. 0x28= Run Flash All-One Verification. 0x2D= Run Checksum Calculation. 0x2E= Vector Remap. 0x40= FLASH 64-bit Read. 0x61= FLASH 64-bit Program. The other commands are invalid.

ISP Trigger Control Register (FMC_ISPTRG)

Register	Offset	R/W	Description	Reset Value
FMC_ISPTRG	FMC_BA+0x10	R/W	ISP Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**ISPGO**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**ISPGO**	**ISP Start Trigger (Write Protect)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. 0 = ISP operation is finished. 1 = ISP is progressed. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Data Flash Base Address Register (FMC_DFBA)

Register	Offset	R/W	Description	Reset Value
FMC_DFBA	FMC_BA+0x14	R	Data Flash Base Address	0xXXXX_XXXX

31	30	29	28	27	26	25	24
DFBA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DFBA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DFBA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DFBA							

Bits	Description	
[31:0]	**DFBA**	**Data Flash Base Address** This register indicates Data Flash start address. It is a read only register. The Data Flash is shared with APROM. the content of this register is loaded from CONFIG1 This register is valid when DFEN (CONFIG0[0]) = 0.

Flash Access Time Control Register (FMC_FTCTL)

Register	Offset	R/W	Description	Reset Value
FMC_FTCTL	FMC_BA+0x18	R/W	Flash Access Time Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**CACHEINV**	**Reserved**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:7]	**Reserved**	Reserved.
[9]	**CACHEINV**	**Flash Cache Invalidation (Write Protect)** 0 = Flash Cache Invalidation finished (default). 1 = Flash Cache Invalidation. **Note 1:** Write 1 to start cache invalidation. The value will be changed to 0 once the process finishes. **Note 2:** This bit is write-protected. Refer to the SYS_REGLCTL register.
[8:0]	**Reserved**	Reserved.

ISP Status Register (FMC_ISPSTS)

Register	Offset	R/W	Description	Reset Value
FMC_ISPSTS	FMC_BA+0x40	R/W	ISP Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	FBS	Reserved	ISPCERR	Reserved			INTFLAG
23	22	21	20	19	18	17	16
VECMAP							
15	14	13	12	11	10	9	8
VECMAP							Reserved
7	6	5	4	3	2	1	0
ALLONE	ISPFF	PGFF	Reserved		CBS		ISPBUSY

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**FBS**	**Flash Bank Selection** This bit indicate which bank is selected to boot. 0 = Booting from BANK0. 1 = Booting from BANK1.
[29]	**Reserved**	Reserved.
[28]	**ISPCERR**	**ISP Conflict Error** This bit shows when FMC is doing ISP operation. User cannot access FMC_ISP_ADDR,FMC_ISPDAT,FMC_ISPCMD,FMC_ISPTRG. It would cause ISPFF.
[27:25]	**Reserved**	Reserved.
[24]	**INTFLAG**	**ISP Interrupt Flag** 0 = ISP Not Finished. 1 = ISP done or ISPFF set. **Note:** This function needs to be enabled by FMC_ISPCTRL[24].
[23:9]	**VECMAP**	**Vector Page Mapping Address (Read Only)** All access to 0x0000_0000~0x0000_01FF is remapped to the Flash memory address {VECMAP[14:0], 9'h000} ~ {VECMAP[14:0], 9'h1FF}
[8]	**Reserved**	Reserved.
[7]	**ALLONE**	**Flash All-one Verification Flag** This bit is set by hardware if all of Flash bits are 1, and clear if Flash bits are not all 1 after "Run Flash All-One Verification" complete; this bit also can be clear by writing 1 0 = All of Flash bits are 1 after "Run Flash All-One Verification" complete. 1 = Flash bits are not all 1 after "Run Flash All-One Verification" complete.

[6]	**ISPFF**	**ISP Fail Flag (Write Protect)** This bit is the mirror of ISPFF (FMC_ISPCTL[6]), it needs to be cleared by writing 1 to FMC_ISPCTL[6] or FMC_ISPSTS[6]. This bit is set by hardware when a triggered ISP meets any of the following conditions: ● APROM writes to itself if APUEN is set to 0. ● LDROM writes to itself if LDUEN is set to 0. ● CONFIG is erased/programmed if CFGUEN is set to 0. ● Erase or Program command at brown-out detected ● Destination address is illegal, such as over an available range. ● Invalid ISP commands ● Page Erase command at LOCK mode with ICE connection ● Invalid operations (except for chip erase) with ICE connection if KSPLOCK is not 0x5A ● Read any content of boot loader with ICE connection ● The base and size of new XOM regsions is wrong, overlap or writed twice ● The input setting of XOM page erase function is wrong ● The active XOM regsion is accessed (except for chip erase, page erase, chksum and read CID/DID) ● The XOM setting page is accessed (except for chip erase, word program and read) ● Violate the load code read protection ● OTP is erased, multi-word programed or 64-bit read ● Checksum or Flash All One Verification is not executed in their valid range ● Block erase or bank erase is not executed in APROM **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**PGFF**	**Flash Program with Fast Verification Flag (Read Only)** This bit is set if data is mismatched at ISP programming verification. This bit is clear by performing ISP Flash erase or ISP read CID operation 0 = Flash Program is success. 1 = Flash Program is fail. Program data is different with data in the Flash memory
[4:3]	**Reserved**	Reserved.
[2:1]	**CBS**	**Boot Selection of CONFIG (Read Only)** This bit is initiated with the CBS (CONFIG0[7:6]) after any reset is happened except CPU reset (CPU is 1) or system reset (SYS) is happened. The following function is valid when MBS (FMC_ISPSTS[3])= 1. 00 = LDROM with IAP mode. 01 = LDROM without IAP mode. 10 = APROM with IAP mode. 11 = APROM without IAP mode.
[0]	**ISPBUSY**	**ISP Busy Flag (Read Only)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. This bit is the mirror of ISPGO(FMC_ISPTRG[0]). 0 = ISP operation is finished. 1 = ISP is progressed.

Flash Access Cyce Control Register (FMC_CYCCTL)

Register	Offset	R/W	Description	Reset Value
FMC_CYCCTL	FMC_BA+0x4C	R/W	Flash Access Cycle Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CYCLE			

Bits	Description	
[31:3]	**Reserved**	Reserved.
[3:0]	**CYCLE**	**Flash Access Cycle Control (Write Protect)** This register is updated by user before HCLK running in different frequency. 0001 = CPU access with one wait cycle if cache miss; Flash access cycle is 1;. The HCLK working frequency range range is<25 MHz 0010 = CPU access with two wait cycles if cache miss; Flash access cycle is 2;. The optimized HCLK working frequency range is 26~50 MHz 0011 = CPU access with three wait cycles if cache miss; Flash access cycle is 3;. The optimized HCLK working frequency range is 51~75 MHz 0100 = CPU access with four wait cycles if cache miss; Flash access cycle is 4;. The optimized HCLK working frequency range is76~100 MHz 0101 = CPU access with five wait cycles if cache miss; Flash access cycle is 5;. The optimized HCLK working frequency range is 101~125 MHz 0110 = CPU access with six wait cycles if cache miss; Flash access cycle is 6;. The optimized HCLK working frequency range is 126~150 MHz 0111 = CPU access with seven wait cycles if cache miss; Flash access cycle is 7;. The optimized HCLK working frequency range is 151~175 MHz 1000 = CPU access with eight wait cycles if cache miss; Flash access cycle is 8;. The optimized HCLK working frequency range is >176 MHz **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

ISP Data 0 Register (FMC_MPDAT0)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT0	FMC_BA+0x80	R/W	ISP Data0 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT0							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT0							

Bits	Description	
[31:0]	**ISPDAT0**	**ISP Data 0** This register is the first 32-bit data for 32-bit/64-bit/multi-word programming, and it is also the mirror of FMC_ISPDAT, both registers keep the same data.

ISP Data 1 Register (FMC_MPDAT1)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT1	FMC_BA+0x84	R/W	ISP Data1 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT1							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT1							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT1							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT1							

Bits	Description	
[31:0]	**ISPDAT1**	**ISP Data 1** This register is the second 32-bit data for 64-bit/multi-word programming.

ISP Data 2 Register (FMC_MPDAT2)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT2	FMC_BA+0x88	R/W	ISP Data2 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT2							
23	22	21	20	19	18	17	16
ISPDAT2							
15	14	13	12	11	10	9	8
ISPDAT2							
7	6	5	4	3	2	1	0
ISPDAT2							

Bits	Description	
[31:0]	ISPDAT2	**ISP Data 2** This register is the third 32-bit data for multi-word programming.

ISP Data 3 Register (FMC_MPDAT3)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT3	FMC_BA+0x8C	R/W	ISP Data3 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT3							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT3							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT3							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT3							

Bits	Description	
[31:0]	**ISPDAT3**	**ISP Data 3** This register is the fourth 32-bit data for multi-word programming.

ISP Multi-program Status Register (FMC_MPSTS)

Register	Offset	R/W	Description	Reset Value
FMC_MPSTS	FMC_BA+0xC0	R	ISP Multi-program Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
D3	**D2**	**D1**	**D0**	**Reserved**	**ISPFF**	**PPGO**	**MPBUSY**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**D3**	**ISP DATA 3 Flag (Read Only)** This bit is set when FMC_MPDAT3 is written and auto-clear to 0 when the FMC_MPDAT3 data is programmed to Flash complete. 0 = FMC_MPDAT3 register is empty, or program to Flash complete. 1 = FMC_MPDAT3 register has been written, and not program to Flash complete.
[6]	**D2**	**ISP DATA 2 Flag (Read Only)** This bit is set when FMC_MPDAT2 is written and auto-clear to 0 when the FMC_MPDAT2 data is programmed to Flash complete. 0 = FMC_MPDAT2 register is empty, or program to Flash complete. 1 = FMC_MPDAT2 register has been written, and not program to Flash complete.
[5]	**D1**	**ISP DATA 1 Flag (Read Only)** This bit is set when FMC_MPDAT1 is written and auto-clear to 0 when the FMC_MPDAT1 data is programmed to Flash complete. 0 = FMC_MPDAT1 register is empty, or program to Flash complete. 1 = FMC_MPDAT1 register has been written, and not program to Flash complete.
[4]	**D0**	**ISP DATA 0 Flag (Read Only)** This bit is set when FMC_MPDAT0 is written and auto-clear to 0 when the FMC_MPDAT0 data is programmed to Flash complete. 0 = FMC_MPDAT0 register is empty, or program to Flash complete. 1 = FMC_MPDAT0 register has been written, and not program to Flash complete.
[3]	**Reserved**	Reserved.

[2]	**ISPFF**	**ISP Fail Flag (Read Only)** This bit is the mirror of ISPFF (FMC_ISPCTL[6]). It needs to be cleared by writing 1 to FMC_ISPCTL[6] or FMC_ISPSTS[6]. This bit is set by hardware when a triggered ISP meets any of the following conditions: ● APROM writes to itself if APUEN is set to 0. ● LDROM writes to itself if LDUEN is set to 0. ● CONFIG is erased/programmed if CFGUEN is set to 0. ● Erase or Program command at brown-out detected ● Destination address is illegal, such as over an available range. ● Invalid ISP commands ● Page Erase command at LOCK mode with ICE connection ● Read any content of boot loader with ICE connection ● The base and size of new XOM regsions is wrong, overlap or writed twice ● The input setting of XOM page erase function is wrong ● The active XOM regsion is accessed (except for chip erase, page erase, chksum and read CID/DID) ● The XOM setting page is accessed (except for chip erase, word program and read) ● Violate the load code read protection ● OTP is erased, multi-word programed or 64-bit read ● Checksum or Flash All One Verification is not executed in their valid range ● Block erase or bank erase is not executed in APROM
[1]	**PPGO**	**ISP Multi-program Status (Read Only)** 0 = ISP multi-word program operation is not active. 1 = ISP multi-word program operation is in progress.
[0]	**MPBUSY**	**ISP Multi-word Program Busy Flag (Read Only)** Write 1 to start ISP Multi-Word program operation and this bit will be cleared to 0 by hardware automatically when ISP Multi-Word program operation is finished. This bit is the mirror of ISPGO(FMC_ISPTRG[0]). 0 = ISP Multi-Word program operation is finished. 1 = ISP Multi-Word program operation is progressed.

ISP Multi-word Program Address Register (FMC_MPADDR)

Register	Offset	R/W	Description	Reset Value
FMC_MPADDR	FMC_BA+0xC4	R	ISP Multi-program Address Register	0x0000_0000

31	30	29	28	27	26	25	24
MPADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
MPADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MPADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MPADDR							

Bits	Description	
[31:0]	**MPADDR**	**ISP Multi-word Program Address** MPADDR is the address of ISP multi-word program operation when ISPGO flag is 1. MPADDR will keep the final ISP address when ISP multi-word program is complete.

XOM Region0 Status Register (FMC_XOMR0STS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMR0STS	FMC_BA+0xD0	R	XOM Region 0 Status Register	0x0FF0_00FF

31	30	29	28	27	26	25	24
BASE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BASE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BASE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SIZE							

Bits	Description	
[31:8]	**BASE**	**XOM Region 0 Base Address** BASE is the base address of XOM Region 0.
[7:0]	**SIZE**	**XOM Region 0 Size** SIZE is the page number of XOM Region 0.

XOM Region1 Status Register (FMC_XOMR1STS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMR1STS	FMC_BA+0xD4	R	XOM Region 1 Status Register	0x0FF0_00FF

31	30	29	28	27	26	25	24
BASE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BASE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BASE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SIZE							

Bits	Description	
[31:8]	**BASE**	**XOM Region 1 Base Address** BASE is the base address of XOM Region 1.
[7:0]	**SIZE**	**XOM Region 1 Size** SIZE is the page number of XOM Region 1.

XOM Region2 Status Register (FMC_XOMR2STS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMR2STS	FMC_BA+0xD8	R	XOM Region 2 Status Register	0x0FF0_00FF

31	30	29	28	27	26	25	24
BASE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BASE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BASE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SIZE							

Bits	Description	
[31:8]	**BASE**	**XOM Region 2 Base Address** BASE is the base address of XOM Region 2.
[7:0]	**SIZE**	**XOM Region 2 Size** SIZE is the page number of XOM Region 2.

XOM Region3 Status Register (FMC_XOMR3STS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMR3STS	FMC_BA+0xDC	R	XOM Region 3 Status Register	0x0FF0_00FF

31	30	29	28	27	26	25	24
BASE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BASE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BASE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SIZE							

Bits	Description	
[31:8]	**BASE**	**XOM Region 3 Base Address** BASE is the base address of XOM Region 3.
[7:0]	**SIZE**	**XOM Region 3 Size** SIZE is the page number of XOM Region 3.

XOM Status Register (FMC_XOMSTS)

Register	Offset	R/W	Description	Reset Value
FMC_XOMSTS	FMC_BA+0xE0	R	XOM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			XOMPEF	XOMR3ON	XOMR2ON	XOMR1ON	XOMR0ON

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**XOMPEF**	**XOM Page Erase Function Fail** XOM page erase function status. If XOMPEF is set to 1, user needs to erase XOM region again. 0 = Sucess. 1 = Fail.
[3]	**XOMR3ON**	**XOM Region 3 On** XOM Region 3 active status. 0 = No active. 1 = XOM region 3 is active.
[2]	**XOMR2ON**	**XOM Region 2 On** XOM Region 2 active status. 0 = No active. 1 = XOM region 2 is active.
[1]	**XOMR1ON**	**XOM Region 1 On** XOM Region 1 active status. 0 = No active. 1 = XOM region 1 is active.
[0]	**XOMR0ON**	**XOM Region 0 On** XOM Region 0 active status. 0 = No active. 1 = XOM region 0 is active.

APROM Protect Register (FMC_APPROT)

Register	Offset	R/W	Description	Reset Value
FMC_APPROT	FMC_BA+0x110	R/W	APROM Protect Register	0x0000_0000

31	30	29	28	27	26	25	24
APPROEN31	APPROEN30	APPROEN29	APPROEN28	APPROEN27	APPROEN26	APPROEN25	APPROEN24
23	22	21	20	19	18	17	16
APPROEN23	APPROEN22	APPROEN21	APPROEN20	APPROEN19	APPROEN18	APPROEN17	APPROEN16
15	14	13	12	11	10	9	8
APPROEN15	APPROEN14	APPROEN13	APPROEN12	APPROEN11	APPROEN10	APPROEN9	APPROEN8
7	6	5	4	3	2	1	0
APPROEN7	APPROEN6	APPROEN5	APPROEN4	APPROEN03	APPROEN02	APPROEN1	APPROEN0

Bits	Description	
[n] n=0,1..31	**APPROENn**	**APROM Proect Enable** This bit indicates which APROM region is protected. 0 = APROM region n is not protected. 1 = APROM region n is protected. **Note:** APROM protect region is 0x0 + n*(0x8000) to 0x7fff + n*(0x8000)

6.6 External Bus Interface (EBI)

6.6.1 Overview

This chip is equipped with an external bus interface (EBI) for external device use. To save the connections between an external device and a chip, EBI is operating at address bus and data bus multiplex mode. The EBI supports three chip selects that can connect three external devices with different timing setting requirements.

6.6.2 Features

- Supports up to three memory banks
- Supports dedicated external chip select pin with polarity control for each bank
- Supports accessible space up to 1 Mbytes for each bank, actually external addressable space is dependent on package pin out
- Supports 8-/16-bit data width
- Supports byte write in 16-bit data width mode
- Supports address bus and data bus multiplex mode
- Supports address bus and data bus separate mode
- Supports Timing parameters individual adjustment for each memory block
- Supports LCD interface i80 mode
- Supports PDMA mode
- Supports variable external bus base clock (MCLK) which based on HCLK
- Supports configurable idle cycle for different access condition: Idle of Write command finish (W2X) and Idle of Read-to-Read (R2R)

6.6.3 Block Diagram

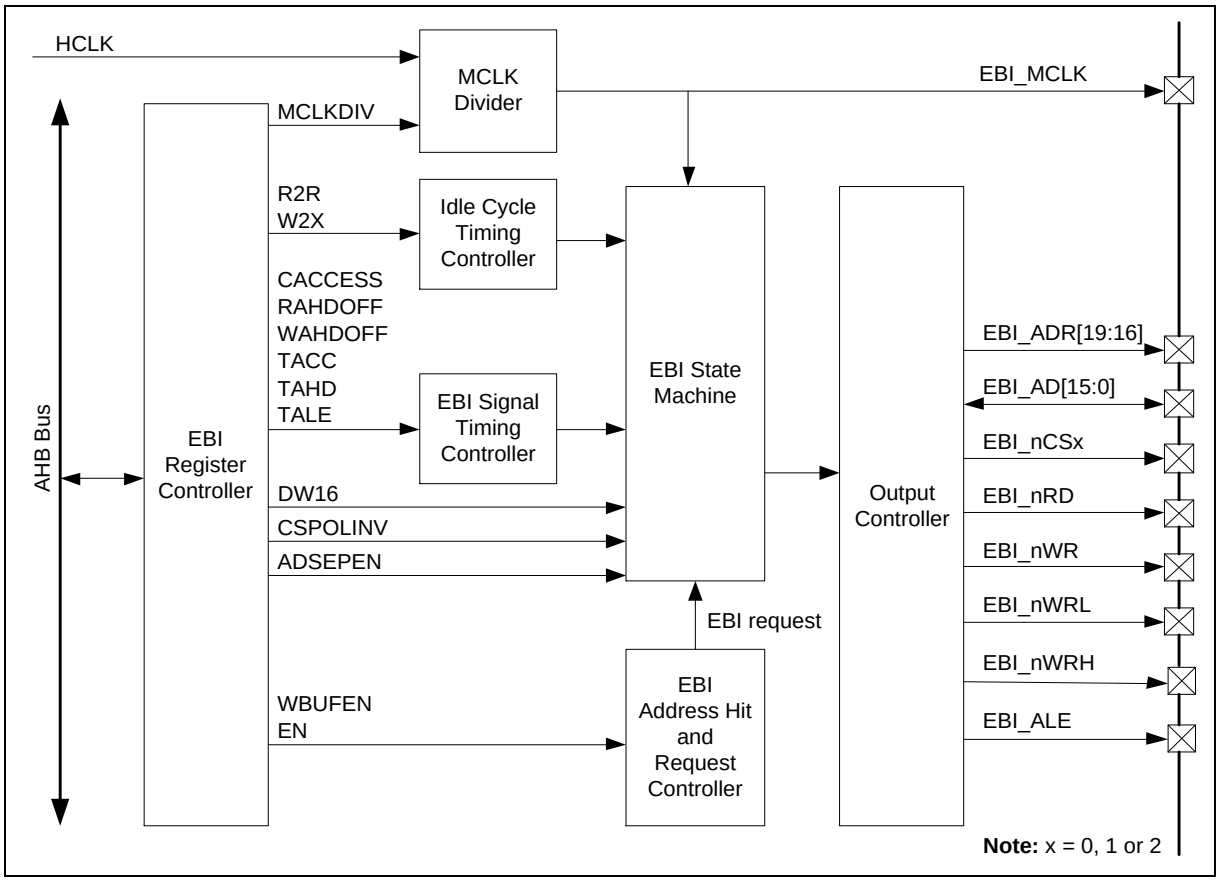


Figure 6.6-1 EBI Block Diagram

6.6.4 Basic Configuration

- Clock Source Configuration
  - Enable EBI controller clock in EBICKEN (CLK_AHBCLK0[3]).
- Reset Configuration
  - Reset EBI controller in EBIRST (SYS_IPRST0[3]).

6.6.5 Functional Description

6.6.5.1 EBI Area and Address Hit

The EBI mapping address is located at 0x6000_0000 ~ 0x602F_FFFF and the total memory space is 3 Mbytes. When system request address hits EBI's memory space, the corresponding EBI chip select signal is assert and EBI state machine operates.

Chip Select	Address Mapping
EBI_nCS0	0x6000_0000 ~ 0x600F_FFFF
EBI_nCS1	0x6010_0000 ~ 0x601F_FFFF
EBI_nCS2	0x6020_0000 ~ 0x602F_FFFF

Table 6.6-1 EBI Address Mapping

To map the whole EBI memory space, it requires 20-bit address for 8-bit data width device and 19-bit address for 16-bit data width device. For package that output less than 20-bit address, EBI will map device to mirror space. For example, the package with 18-bit EBI address, EBI will mapped external device (for Bank0/EBI_nCS0) to 0x6000_0000 ~ 0x6003_FFFF, 0x6004_0000 ~ 0x6007_FFFF, 0x6008_0000 ~ 0x600B_FFFF and 0x600C_0000 ~ 0x600F_FFFF simultaneously.

6.6.5.2 EBI Data Width Connection - Address Bus and Data Bus Multiplex Mode

The EBI supports the device whose address bus and data bus are multiplexed. For the external device with separated address and data bus, the connection to device needs additional latch device to latch the address. In this case, the pin EBI_ALE is connected to the latch device to latch the address value. Pin EBI_AD is the input of the latch device, and the output of the latch device is connected to the address of external device.

For 16-bit device, the EBI_AD[15:0] is shared by address and 16-bit data, and EBI_ADR[18:16] is dedicated for address and could be connected to 16-bit device directly. The EBI_ADR[19] will be ignored when EBI data width is set as 16-bit width. For 8-bit device, only EBI_AD[7:0] is shared by address and 8-bit data, EBI_AD[15:8] and EBI_ADR[19:16] are dedicated for address and could be connected to 8-bit device directly. Figure 6.6-2 shows the connection of 16-bit data width device and Figure 6.6-3 shows the connection of 8-bit data width device.

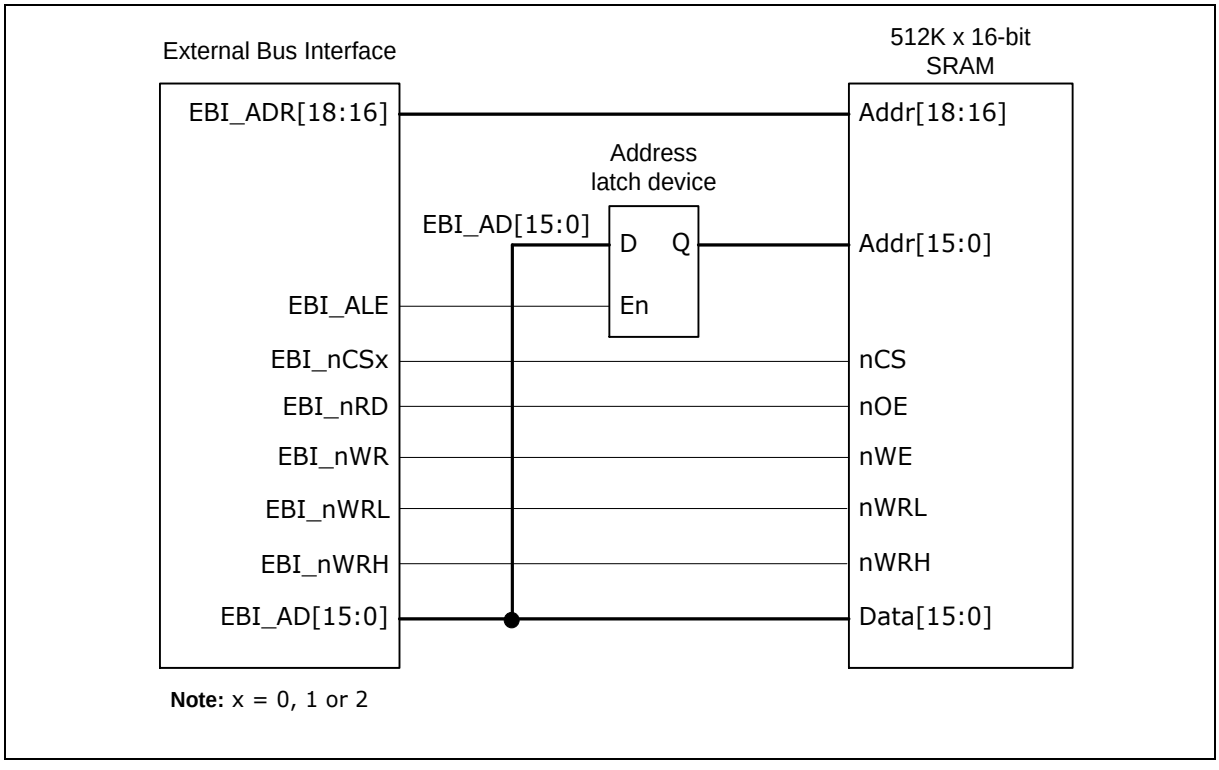


Figure 6.6-2 Connection of 16-bit EBI Data Width with 16-bit Device

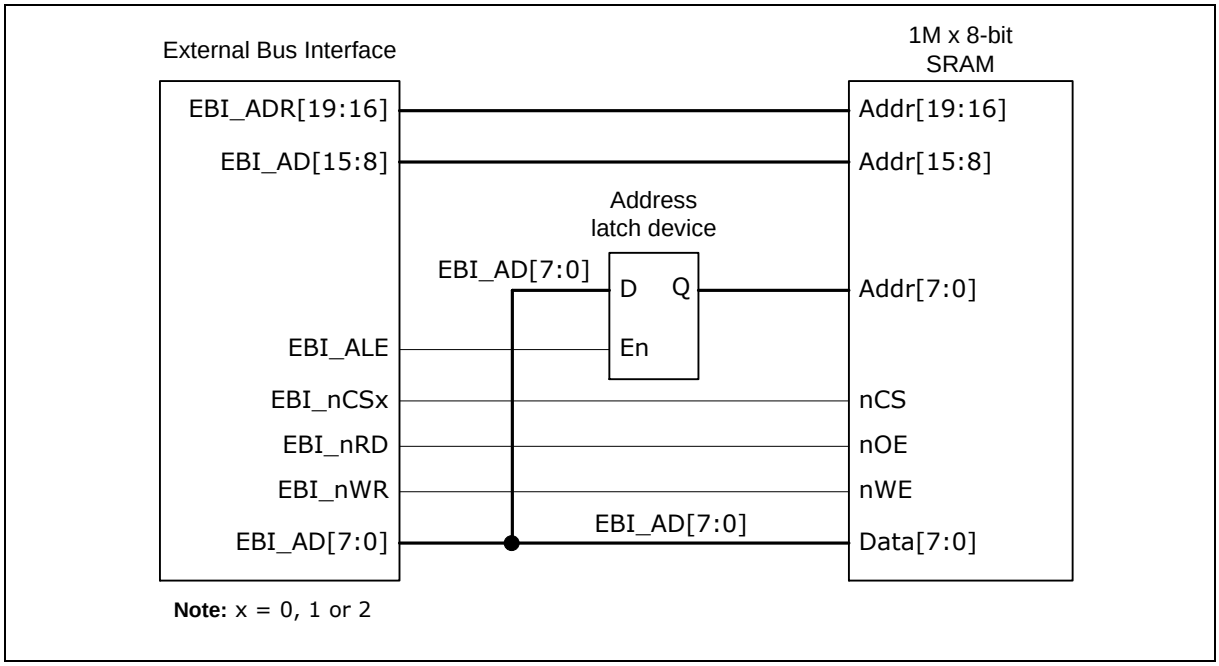


Figure 6.6-3 Connection of 8-bit EBI Data Width with 8-bit Device

When system access data width is larger than EBI data width, the EBI controller will finish a system access command by operating EBI access more than once. For example, if system requests a 32-bit data through EBI device, the EBI controller will operate accessing four times when setting EBI data width with 8-bit.

6.6.5.3 EBI Data Width Connection - Address Bus and Data Bus Separate Mode

The EBI supports address and data bus separate mode. User can enable this mode by setting ADSEPEN (EBI_CTLx[3]). When separate mode is enabled, EBI_AD is dedicated for data bus and connected directly to device data bus, EBI_ADR is dedicated for address bus.

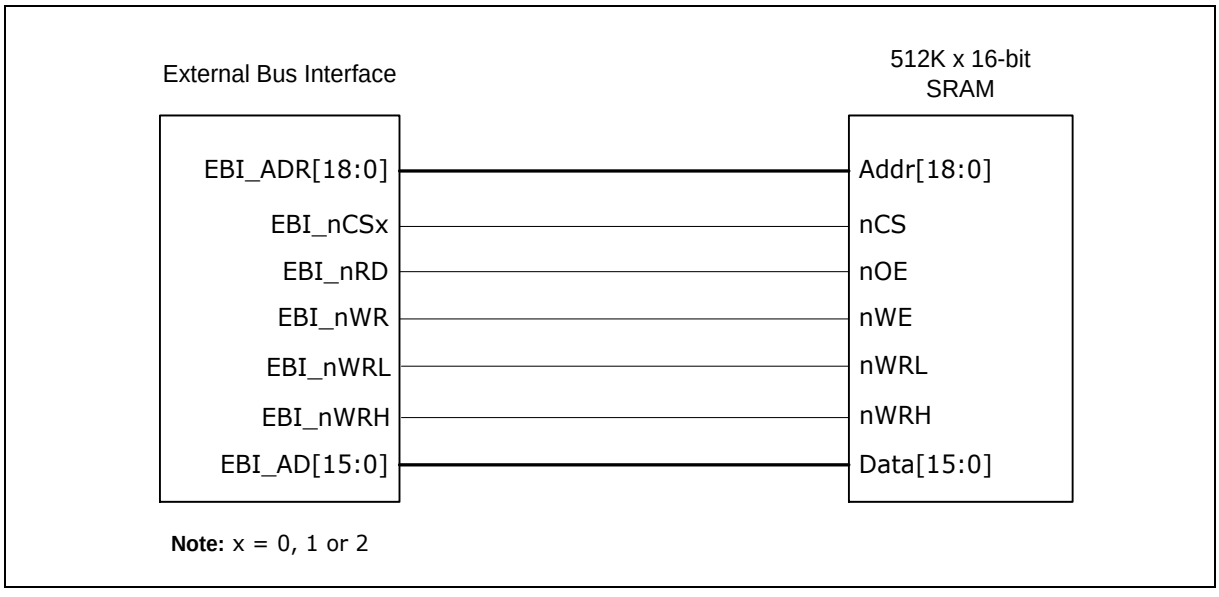


Figure 6.6-4 Connection of 16-bit EBI Data Width with 16-bit Device in Separate mode

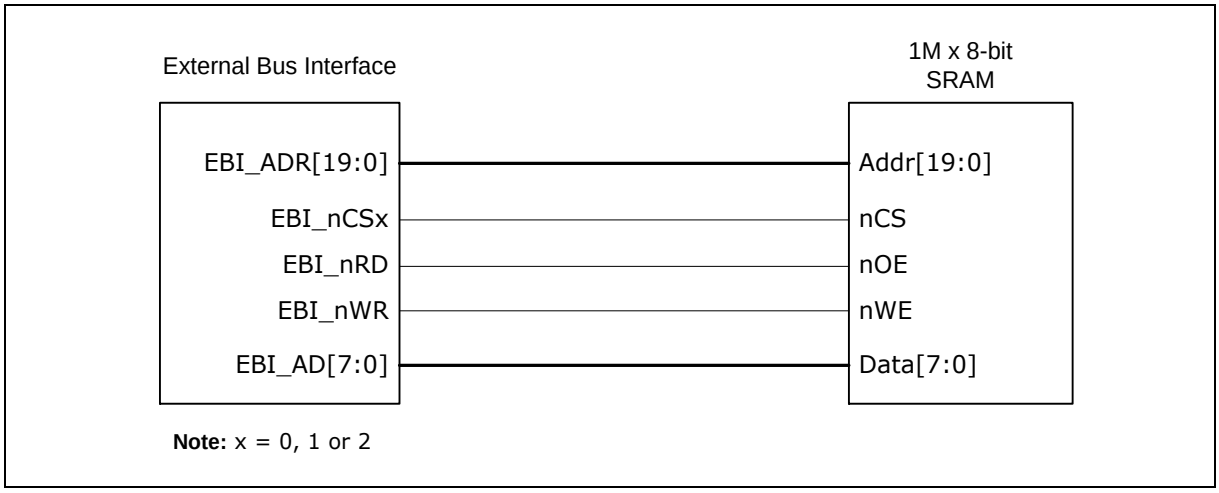


Figure 6.6-5 Connection of 8-bit EBI Data Width with 8-bit Device in Separate mode

6.6.5.4 EBI Operating Control

MCLK Control

In the chip, all EBI signals will be synchronized by EBI_MCLK when EBI is operating. When chip connects to the external device with slower operating frequency, the EBI_MCLK can divide most to HCLK/128 by setting MCLKDIV (EBI_CTLx[10:8]). Therefore, chip can be suitable for a wide frequency range of EBI device. EBI signals are all synchronized by negative edge of EBI_MCLK.

Operation and Access Timing Control

At the start of EBI access, chip select (EBI_nCS0, EBI_nCS1 and EBI_nCS2) asserts to low and wait one EBI_MCLK for address setup time (tASU) for address stable. Then EBI_ALE asserts to high after address is stable and keeps for a period of time (tALE) for address latch. After latch address, EBI_ALE asserts to low and wait one EBI_MCLK for latch hold time (tLHD) and another one EBI_MCLK cycle (tA2D) that is inserted behind address hold time to be the bus turn-around time for address change to data. Then EBI_nRD asserts to low when read access or EBI_nWR asserts to low when write access. Then EBI_nRD or EBI_nWR asserts to high after keeps access time (tACC) for reading output stable or writing finish. After that, EBI signals keep for data access hold time (tAHD) and chip select asserts to high, address is released by current access control.

The EBI controller provides a flexible timing control for different external device. In EBI timing control, tASU, tLHD and tA2D are fixed to 1 EBI_MCLK cycle, tAHD can modulate to 1~8 EBI_MCLK cycles by setting TAHD (EBI_TCTLx[10:8]), tACC can modulate to 1~32 EBI_MCLK cycles by setting TACC (EBI_TCTLx[7:3]), and tALE can modulate to 1~8 EBI_MCLK cycles by setting TALE (EBI_CTL0[18:16]). Some external device can support zero data access hold time accessing, the EBI controller can skipped tAHD to increase access speed by setting WAHDOFF (EBI_TCTLx[23]) and RAHDOFF (EBI_TCTLx[22]).

For each chip select, the EBI provides individual register with timing control except that tALE can only be controlled by EBI_CTL0.

Parameter	Value	Unit	Description
tASU	1	MCLK	Address Latch Setup Time.
tALE	1 ~ 8	MCLK	ALE High Period. Controlled by TALE (EBI_CTL0[18:16]).
tLHD	1	MCLK	Address Latch Hold Time.

tA2D	1	MCLK	Address To Data Delay (Bus Turn-Around Time).
tACC	1 ~ 32	MCLK	Data Access Time. Controlled by TACC (EBI_TCTLx[7:3]).
tAHD	1 ~ 8	MCLK	Data Access Hold Time. Controlled by TAHD (EBI_TCTLx[10:8]).
IDLE	0 ~ 15	MCLK	Idle Cycle. Controlled by R2R (EBI_TCTLx[27:24]) and W2X (EBI_TCTLx[15:12]).

Table 6.6-2 Timing Control Parameter

Figure 6.6-6 shows an example of setting 16-bit data width. In this example, EBI_AD bus is used for being address [15:0] and data [15:0]. When EBI_ALE assert to high, EBI_AD is address output. After address is latched, EBI_ALE asserts to low and the EBI_AD bus change to high impedance to wait device output data in read access operation, or it is used for being write data output.

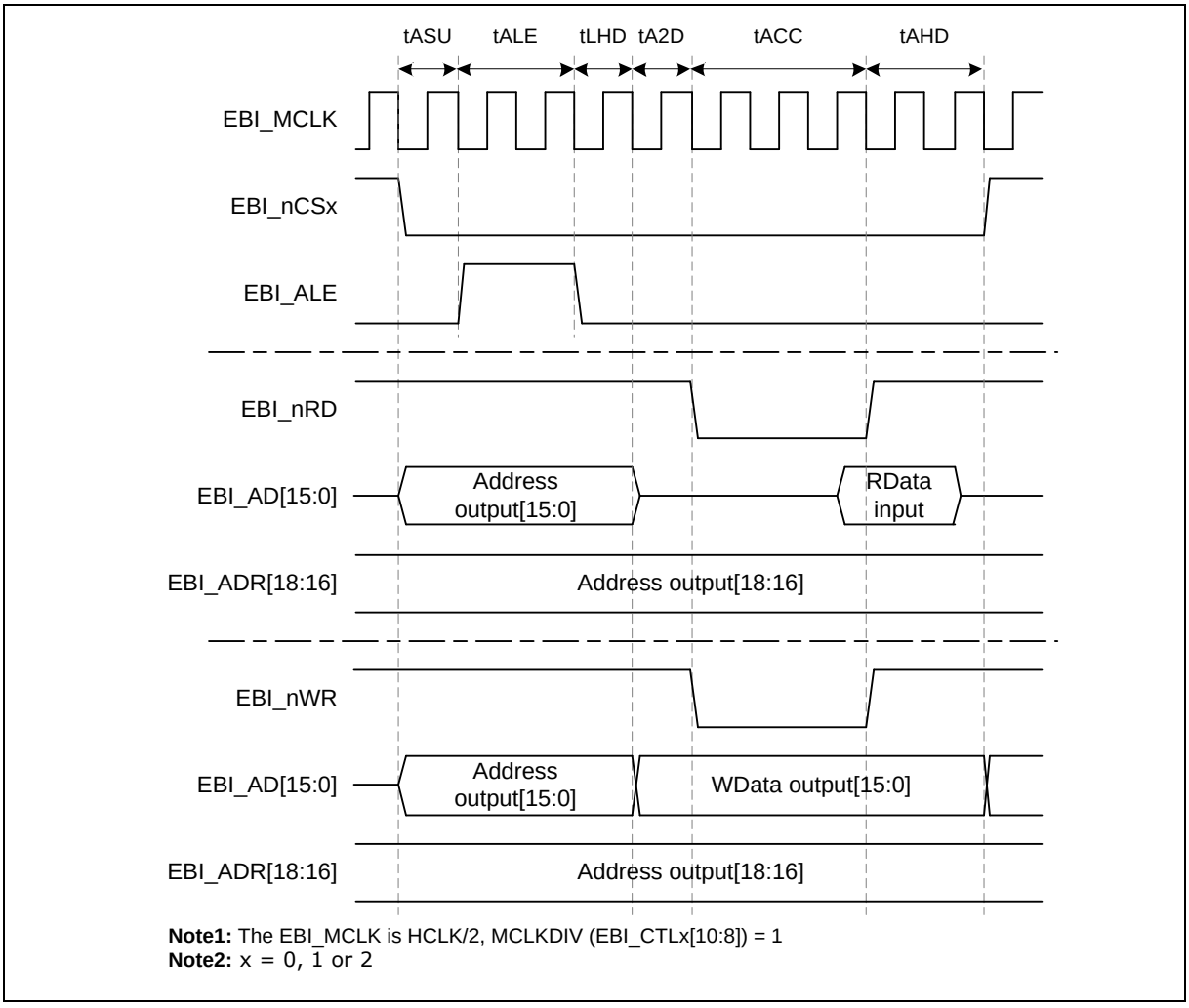
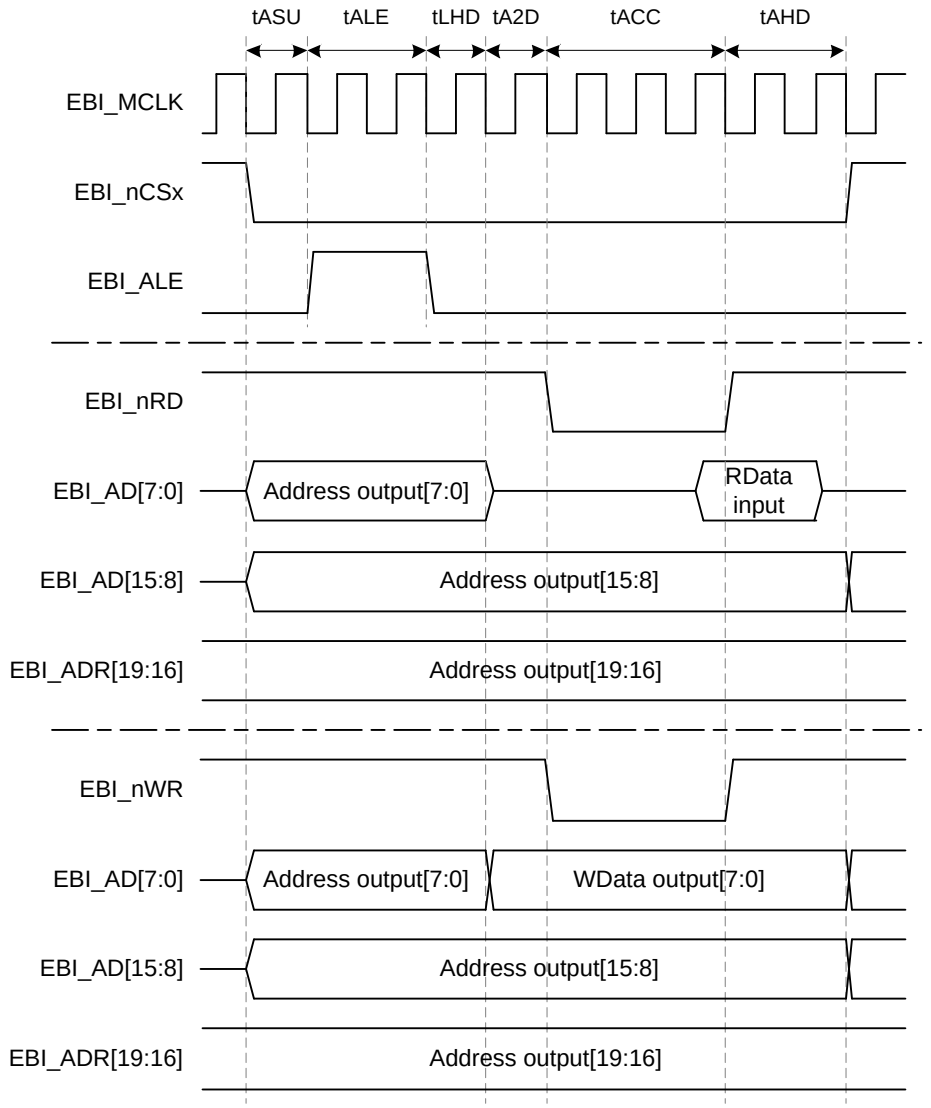


Figure 6.6-6 Timing Control Waveform for 16-bit Data Width

Figure 6.6-7 shows an example of setting 8-bit data width. The difference between 8-bit and 16-bit data width is EBI_AD[15:8]. In 8-bit data width setting, EBI_AD[15:8] is always Address [15:8] output so that external latch needs only 8-bit width.

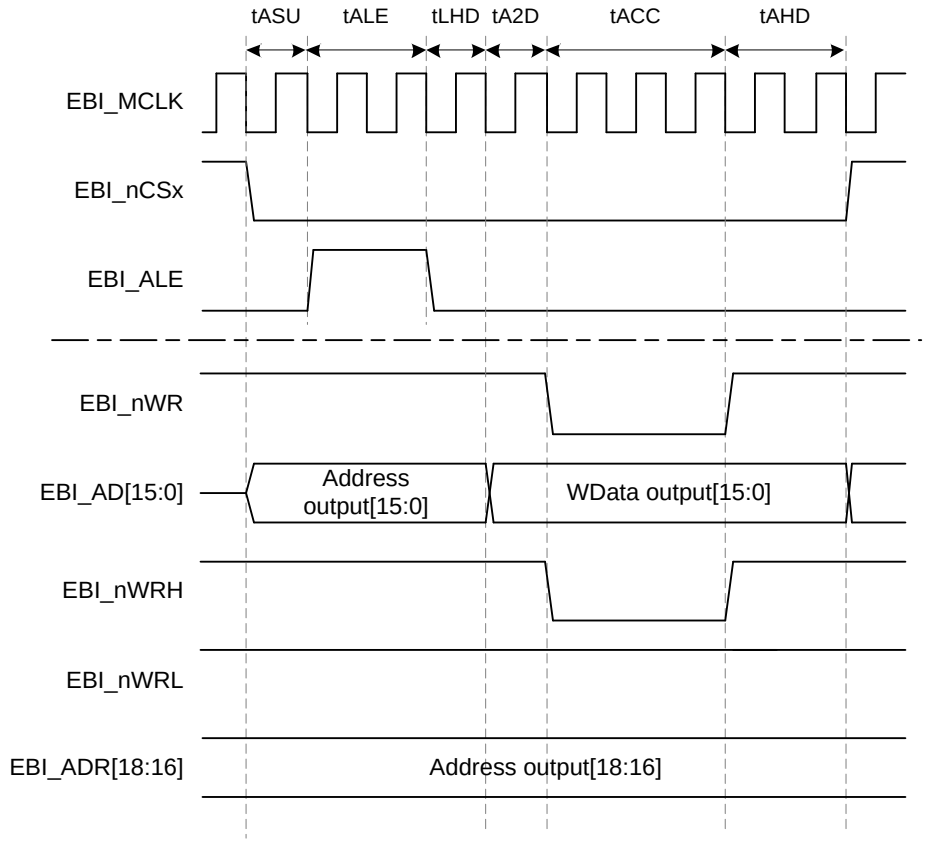


Note1: The EBI_MCLK is HCLK/2, MCLKDIV (EBI_CTLx[10:8]) = 1
Note2: x = 0, 1 or 2

Figure 6.6-7 Timing Control Waveform for 8-bit Data Width

Byte Access

The EBI supports byte access when connected to 16-bit device. The pin EBI_nWRH and EBI_nWRL assertion indicate high byte enable and low byte enable in 16-bit data bus. Figure 6.6-8 shows the write operation of 8-bit width data in EBI_AD[15:8] with EBI_nWRH assertion.



Note1: The EBI_MCLK is HCLK/2, MCLKDIV (EBI_CTLx[10:8]) = 1
Note2: x = 0, 1 or 2
Note3: WData[15:8] is valid with EBI_nWRH assertion

Figure 6.6-8 Timing Control Waveform for Byte Write in 16-bit Data Mode

Insert Idle Cycle

When EBI accesses continuously, there may occur bus conflict if the device access time is much slow with system operating. The EBI controller supplies additional idle cycle to solve this problem. During idle cycle, all control signals of EBI are inactive. Figure 6.6-9 shows idle cycles.

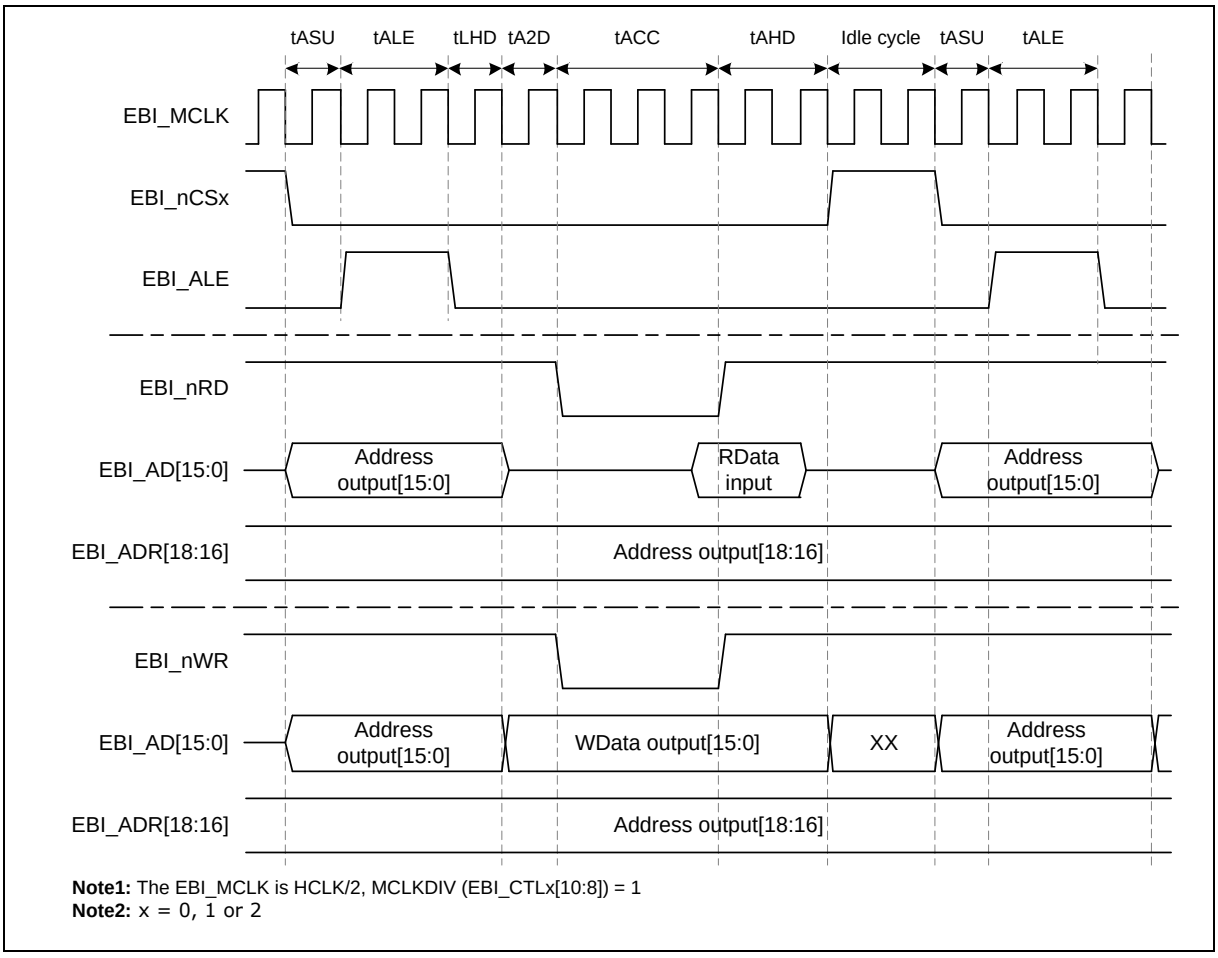


Figure 6.6-9 Timing Control Waveform for Insert Idle Cycle

There are two conditions that EBI can insert idle cycle by timing control:

1. After write access (W2X idle cycle)
2. After read access and before next read access (R2R idle cycle)

By setting W2X (EBI_TCTLx[15:12]), and R2R (EBI_TCTLx[27:24]), the time of idle cycle can be specified from 0~15 EBI_MCLK.

Chip Select Polarity Control

The EBI supports chip select polarity control for connecting to variable external device. When CSPOLINV (EBI_CTLx[2]) is set to 0, the chip select pins (EBI_nCSx) works as low active behavior. It means the external device can be access under EBI_nCSx at low state. When CSPOLINV (EBI_CTLx[2]) is set to 1, the chip select pin (EBI_nCS) works as high active behavior. It means the external device can be access under EBI_nCSx at high state.

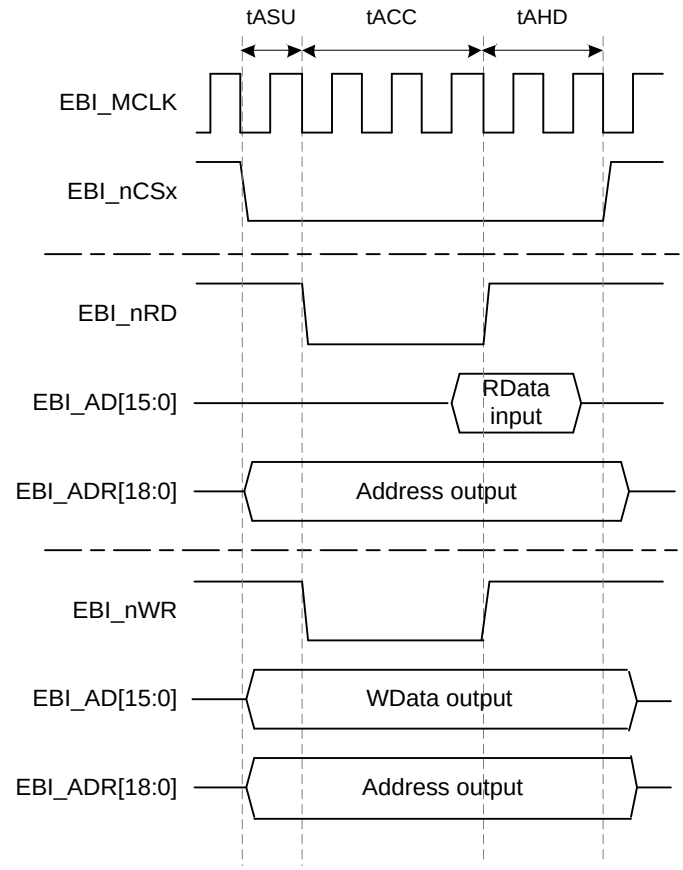
Write Buffer

When user writes data to an external device through EBI bus, the EBI controller will start processing the write action immediately and the CPU is held until current EBI write action is finished. User can enable write buffer function to improve CPU and EBI access performance. When EBI write buffer function is enabled, the CPU can continuously execute other instruction during EBI controller process the write action to external device. There is one exception condition for this case. If CPU executes another data access through EBI bus when EBI process write action, the CPU will be held.

User can enable write buffer by setting WBUFEN (EBI_CTL0[24]).

Address Data Separate Mode

When EBI is set as separate mode, the tALE, tLHD, tA2D cycles are ignored. EBI_AD and EBI_ADR are dedicated for data and address bus separately.



Note1: The EBI_MCLK is HCLK/2, MCLKDIV (EBI_CTLx[10:8]) = 1
Note2: x = 0, 1 or 2

Figure 6.6-10 Timing Control Waveform for 16-bit Data Width for Separate Mode

Continuous Data Access Mode

The EBI supports continuous data access mode for the device which needs faster data access and do not need address control interface. User can enable this mode by setting CACCESS (EBI_CTLx[4]) for each bank. When EBI is set as continuous data access mode, the tASU, tALE, tLHD cycles are ignored and EBI can access data continuously within one read or write command. There will be dummy cycle

between each access command. The timing waveform is shown as Figure 6.6-11.

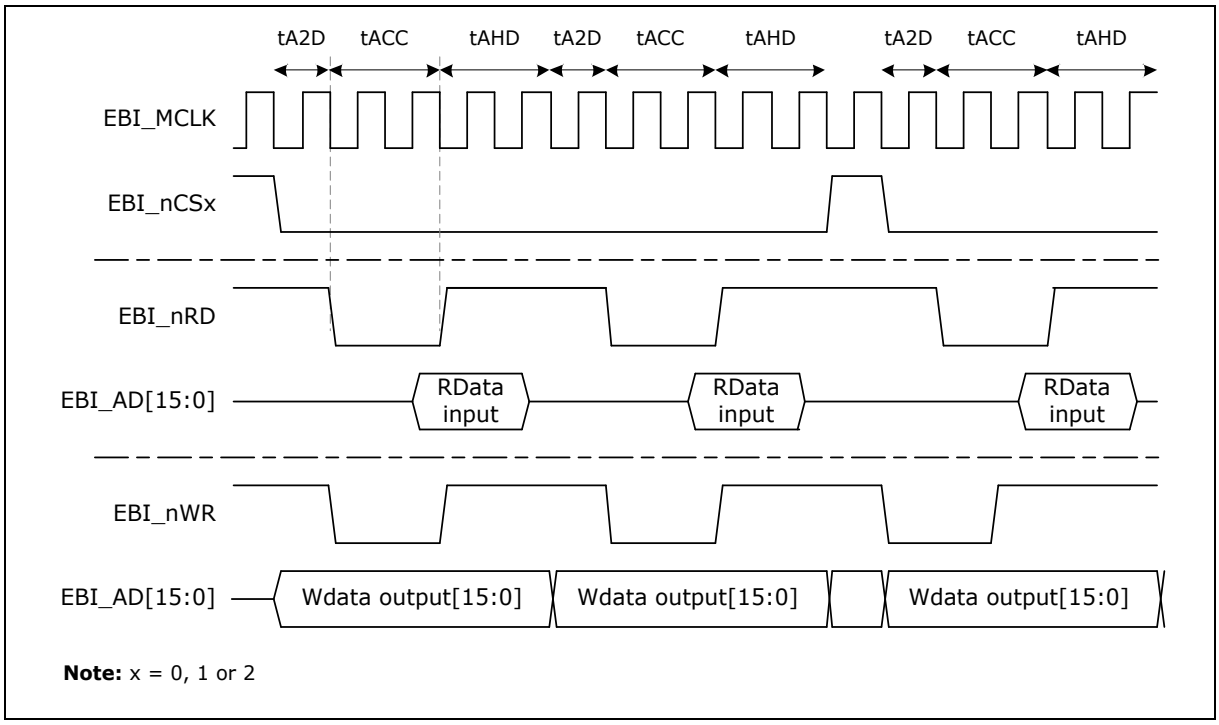


Figure 6.6-11 Timing Control Waveform for Continuous Data Access Mode

6.6.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EBI Base Address: **EBI_BA = 0x4001_0000**				
EBI_CTL0	EBI_BA+0x00	R/W	External Bus Interface Bank0 Control Register	0x0000_0000
EBI_TCTL0	EBI_BA+0x04	R/W	External Bus Interface Bank0 Timing Control Register	0x0000_0000
EBI_CTL1	EBI_BA+0x10	R/W	External Bus Interface Bank1 Control Register	0x0000_0000
EBI_TCTL1	EBI_BA+0x14	R/W	External Bus Interface Bank1 Timing Control Register	0x0000_0000
EBI_CTL2	EBI_BA+0x20	R/W	External Bus Interface Bank2 Control Register	0x0000_0000
EBI_TCTL2	EBI_BA+0x24	R/W	External Bus Interface Bank2 Timing Control Register	0x0000_0000

6.6.7 Register Description

External Bus Interface Control Register (EBI_CTLx)

Register	Offset	R/W	Description	Reset Value
EBI_CTL0	EBI_BA+0x00	R/W	External Bus Interface Bank0 Control Register	0x0000_0000
EBI_CTL1	EBI_BA+0x10	R/W	External Bus Interface Bank1 Control Register	0x0000_0000
EBI_CTL2	EBI_BA+0x20	R/W	External Bus Interface Bank2 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reversed							WBUFEN
23	22	21	20	19	18	17	16
Reversed					TALE		
15	14	13	12	11	10	9	8
Reversed					MCLKDIV		
7	6	5	4	3	2	1	0
Reversed			CACCESS	ADSEPEN	CSPOLINV	DW16	EN

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**WBUFEN**	**EBI Write Buffer Enable Bit** 0 = EBI write buffer Disabled. 1 = EBI write buffer Enabled. **Note:** This bit is only available in EBI_CTL0 register.
[23:19]	**Reserved**	Reserved.
[18:16]	**TALE**	**Extend Time of ALE** The EBI_ALE high pulse period (tALE) to latch the address can be controlled by TALE. tALE = (TALE + 1) * EBI_MCLK. **Note:** This field is only available in EBI_CTL0 register.
[15:11]	**Reserved**	Reserved.
[10:8]	**MCLKDIV**	**External Output Clock Divider** The frequency of EBI output clock (MCLK) is controlled by MCLKDIV as follow: 000 = HCLK/1. 001 = HCLK/2. 010 = HCLK/4. 011 = HCLK/8. 100 = HCLK/16. 101 = HCLK/32. 110 = HCLK/64. 111 = HCLK/128.
[7:5]	**Reserved**	Reserved.

[4]	**CACCESS**	**Continuous Data Access Mode** When continuous access mode enabled, the tASU, tALE and tLHD cycles are bypass for continuous data transfer request. 0 = Continuous data access mode Disabled. 1 = Continuous data access mode Enabled.
[3]	**ADSEPEN**	**EBI Address/Data Bus Separate Mode Enable Bit** 0 = Address/Data Bus Separate Mode Disabled. 1 = Address/Data Bus Separate Mode Enabled.
[2]	**CSPOLINV**	**Chip Select Pin Polar Inverse** This bit defines the active level of EBI chip select pin (EBI_nCS). 0 = Chip select pin (EBI_nCS) is active low. 1 = Chip select pin (EBI_nCS) is active high.
[1]	**DW16**	**EBI Data Width 16-bit Select** This bit defines if the EBI data width is 8-bit or 16-bit. 0 = EBI data width is 8-bit. 1 = EBI data width is 16-bit.
[0]	**EN**	**EBI Enable Bit** This bit is the functional enable bit for EBI. 0 = EBI function Disabled. 1 = EBI function Enabled.

External Bus Interface Timing Control Register (EBI_TCTLx)

Register	Offset	R/W	Description	Reset Value
EBI_TCTL0	EBI_BA+0x04	R/W	External Bus Interface Bank0 Timing Control Register	0x0000_0000
EBI_TCTL1	EBI_BA+0x14	R/W	External Bus Interface Bank1 Timing Control Register	0x0000_0000
EBI_TCTL2	EBI_BA+0x24	R/W	External Bus Interface Bank2 Timing Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				R2R			
23	22	21	20	19	18	17	16
WAHDOFF	RAHDOFF	Reserved					
15	14	13	12	11	10	9	8
W2X				Reversed	TAHD		
7	6	5	4	3	2	1	0
TACC					Reserved		

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**R2R**	**Idle Cycle Between Read-to-read** This field defines the number of R2R idle cycle. R2R idle cycle = (R2R * EBI_MCLK). When read action is finished and the next action is going to read, R2R idle cycle is inserted and EBI_nCS return to idle state.
[23]	**WAHDOFF**	**Access Hold Time Disable Control When Write** 0 = Data Access Hold Time (tAHD) during EBI writing Enabled. 1 = Data Access Hold Time (tAHD) during EBI writing Disabled.
[22]	**RAHDOFF**	**Access Hold Time Disable Control When Read** 0 = Data Access Hold Time (tAHD) during EBI reading Enabled. 1 = Data Access Hold Time (tAHD) during EBI reading Disabled.
[21:16]	**Reserved**	Reserved.
[15:12]	**W2X**	**Idle Cycle After Write** This field defines the number of W2X idle cycle. W2X idle cycle = (W2X * EBI_MCLK). When write action is finished, W2X idle cycle is inserted and EBI_nCS return to idle state.
[11]	**Reserved**	Reserved.
[10:8]	**TAHD**	**EBI Data Access Hold Time** TAHD defines data access hold time (tAHD). tAHD = (TAHD + 1) * EBI_MCLK.
[7:3]	**TACC**	**EBI Data Access Time** TACC defines data access time (tACC).

		tACC = (TACC + 1) * EBI_MCLK.
[2:0]	**Reserved**	Reserved.

6.7 HyperBus Interface Controller (HBI)

6.7.1 Overview

HyperBus is a low signal count, Double Data Rate (DDR) interface, which achieves high speed read and write throughout. The DDR protocol transfers two data bytes per clock cycle on the DQ input/output signals. A read or write transaction on HyperBus consists of a series of 16-bit wide, one clock cycle data transfer at the HyperRAM memory.

There is one APB and one AHB interfaces in this controller. The APB interface is used to configure the setting registers and short word access. The AHB interface is used to direct access HyperRAM up to 32 Mbytes.

6.7.2 Features

- HyperRAM 2.0
  - HyperRAM 1.0 supported
- Supports 32 Mbytes address memory space access
- Up to 90Mhz HyperBus clock rate
  - Maximum 1440M Bits/s data rate
- Configurable reset timing control
- Configurable chip select timing control for HyperRAM exit from HS (Hybrid Sleep) or DPD (deep power down) mode

6.7.3 Block Diagram

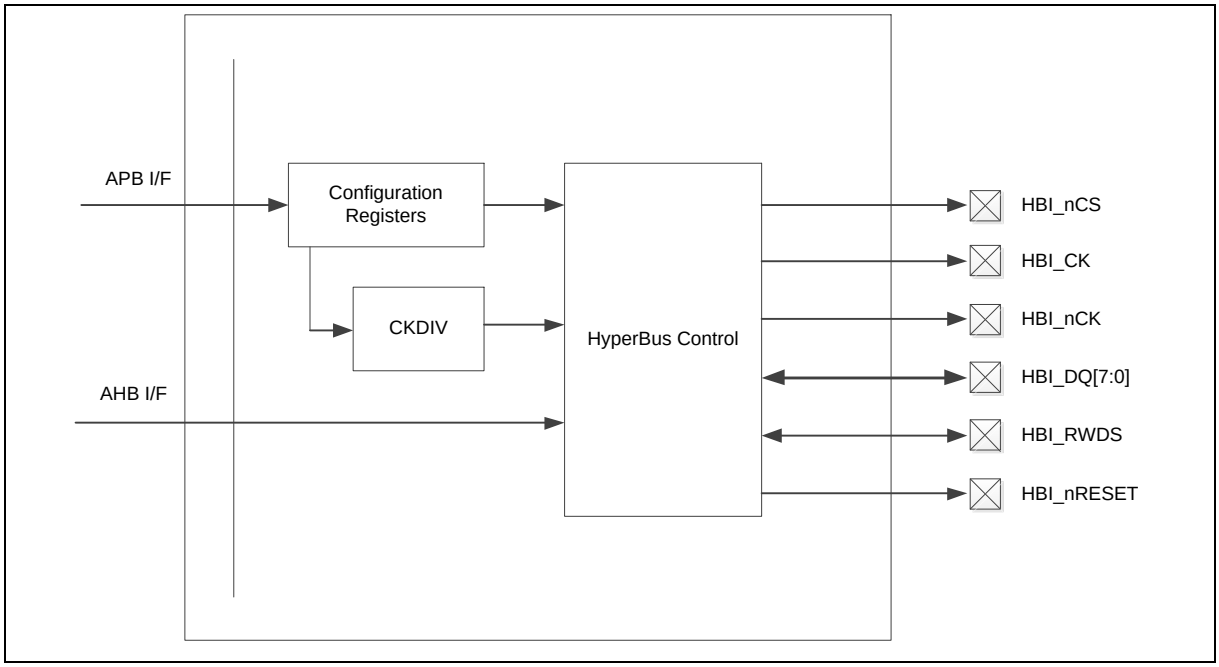


Figure 6.7-1 HyperBus Interface Controller Block Diagram

6.7.4 Basic Configuration

- Clock Source Configuration
  - Enable HBI clock in HBICKEN (CLK_APB2[8]).

- Reset Configuration
  - Reset HBI controller in HBIRST (SYS_IPRST0[18]).

6.7.4.1 HBI Pin Description

Name	BITs	I/O Type	Pin Description
HBI_nCS	1	Output	Chip Select Bus transactions are initiated with a High to Low transition. Bus transactions are terminated with a Low to High transition.
HBI_CK/HBI_nCK	2	Output	Differential Clock: Command, address, and data information is output with respect to the crossing of the HBI_CK and HBI_nCK signals. Single Ended Clock: HBI_nCK is not used, only a single ended HBI_CK is used.
HBI_DQ	8	Input/Output	Data Input/ Ouput Command, Address, and Data information is transferred on these signals during Read and Write transactions.
HBI_RWDS	1	Input/Output	Read Write Data Strobe During the Command/Address portion of all bus transactions RWDS is a slave output and indicates whether additional initial latency is required. Slave output during read data transfer, data is edge aligned with RWDS. Slave input during data transfer in write transactions to function as a data mask.
HBI_nRESET	1	Output	Hardware Reset When Low the slave device will self-initialize and return to the Standby state.

6.7.5 Functional Description

6.7.5.1 HyperBus Interface

In HyperRAM, command, address, and data information are transferred over the eight HBI_DQ[7:0] signals. The clock (HBI_CK, HBI_nCK) is used for information capture by a HyperBus slave device when receiving command, address, or data on the HBI_DQ signals. Command or Address values are center aligned with clock transitions.

Every transaction begins with the assertion of HBI_nCS and Command-Address (CA) signals, followed by the start of clock transitions to transfer six CA bytes, followed by initial access time and either read or write data transfers, until HBI_nCS is de-asserted.

Read and write transactions require two clock cycles to define the target row address and burst type, then an initial access time of t_{ACC}. During the CA part of a transaction, the memory will indicate whether an additional latency for a required refresh time ((t_{RFH}, refer to HyperRAM specification,) is added to the initial time; the ACCT (HBI_CONFIG[5:2]) bit is used to define the initial access time. During the CA period the third clock cycle will specify the target word address within the target row. During a read (or write) transaction, after the initial data value has been output (or input), additional data can be read from (or written to) the row on subsequent clock cycles in either a wrapped or linear sequenced (Refer to HYPERAM specification's CR0[2]). The controller can provide a sustained data rate of 384 Mbytes/s (1 byte (8 bit data bus) * 2 (data clock edges) * 192 MHz = 384 Mbytes/s). The Figure 6.7-2 shows the transaction timing of HyperRAM (Read Command).

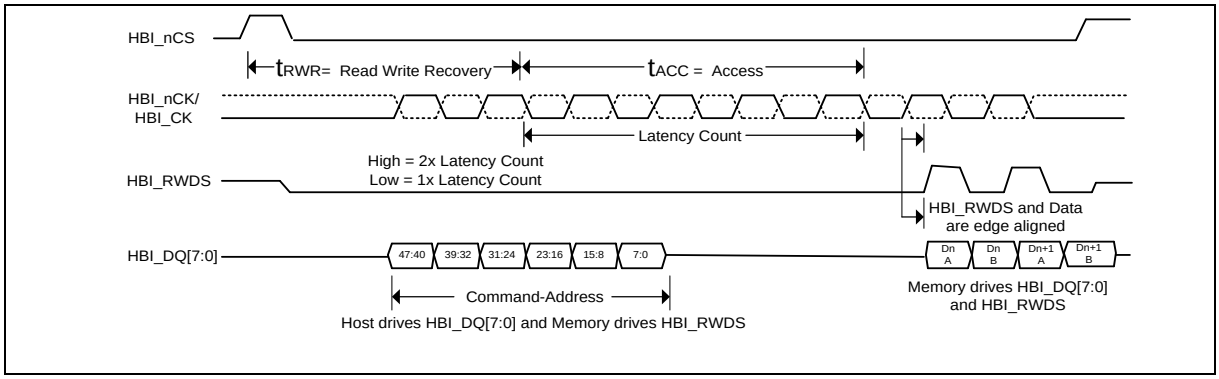


Figure 6.7-2 HyperRAM Read Transaction, Single Initial Latency Count

The Read/Write Data Strobe (HBI_RWDS) is a bidirectional signal that indicates:

- When data will start to transfer from a HyperRAM device to the master device in read transactions (initial read latency).
- When data is being transferred from a HyperRAM device to the master device during read transactions (as a source synchronous read data strobe)
- When data may start to transfer from the master device to a HyperRAM device in write transactions (initial write latency)
- Data masking during write data transfers

During the CA transfer portion of a read or write transaction, HBI_RWDS acts as an output from a HyperRAM device to indicate whether additional initial access latency is needed in the transaction.

During read data transfers, HBI_RWDS is a read data strobe with data values edge aligned with the transitions of HBI_RWDS.

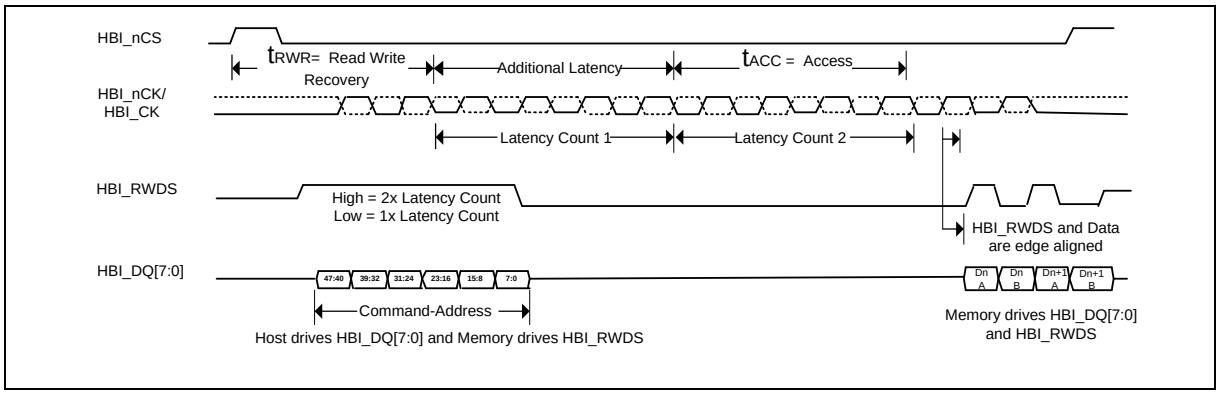


Figure 6.7-3 Read Transaction, Additional Latency Count

During write data transfers, RWDS indicates whether each data byte transfer is masked with RWDS High (invalid and prevented from changing the byte location in a memory) or not masked with RWDS Low (valid and written to a memory). Data masking may be used by the host to byte align write data within a memory or to enable merging of multiple non-word aligned writes in a single burst write. During write transactions, data is center aligned with clock transitions.

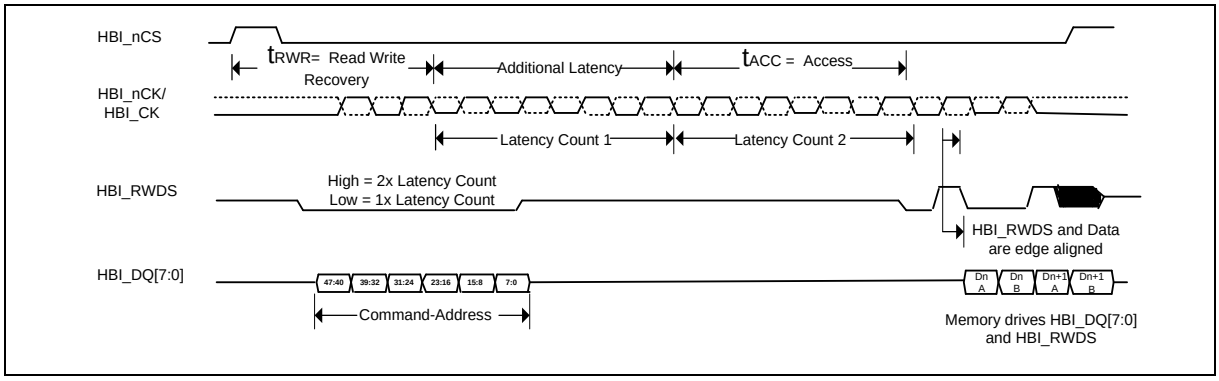


Figure 6.7-4 Write Transaction, Single Initial Latency Count

- Command / Address Bit Assignments

All HyperRAM bus transactions can be classified as either read or write. A bus transaction is started with HBI_nCS going Low with clock in idle state (HBI_CK=Low and HBI_nCK=High). The first three clock cycles transfer three words of Command/Address (CA0, CA1, CA2) information to define the transaction characteristics. The Command/Address words are presented with DDR timing, using the first six clock edges. The following characteristics are defined by the Command/Address information:

- Read or Write transaction
- Address Space: memory array space or register space
  - Register space is used to access Device Identification (ID) registers and Configuration Registers (CR) that identify the device characteristics and determine the slave specific behavior of read and write transfers on the HyperBus interface.
- Whether a transaction will use a linear or wrapped burst sequence
- The target row (and half-page) address (upper order address)
- The target column (word within half-page) address (lower order address)

Note: For detailed information, refer to HyperRAM specification.

- Data Placement for HyperBus Transaction
  - Read Transaction

Data is edge aligned with the HBI_RWDS serving as a READ data strobe during read transaction. Data is always transferred in full words increment. Word address will be increased in each clock cycle. Byte A is between HBI_RWDS rising and falling edges and is followed by Bytes B between HBI_RWDS falling and rising edges, of each word. The data bits in each byte are always in high to low order with bit 7 on HBI_DQ[7] and bit 0 on HBI_DQ[0].

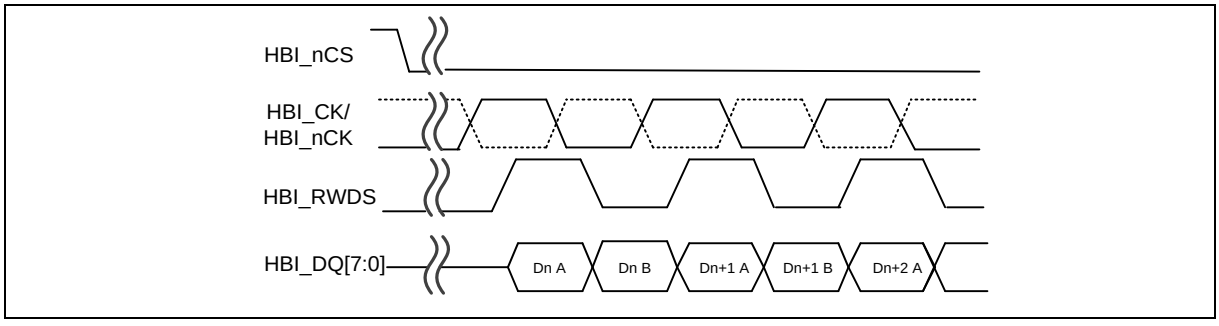


Figure 6.7-5 Data Placement during a Read Transaction

- Write Transaction

Data is center aligned with the clock during a WRITE transaction. HBI_RWDS function as a data mask during write data transfers with initial latency. Masking of the first and last byte is shown to illustrate an unaligned 3 bytes write data. HBI_RWDS is not driven by the master during write data transfer with zero initial latency. Full data words are always written in this case. HBI_RWDS may be driven LOW by the slave.

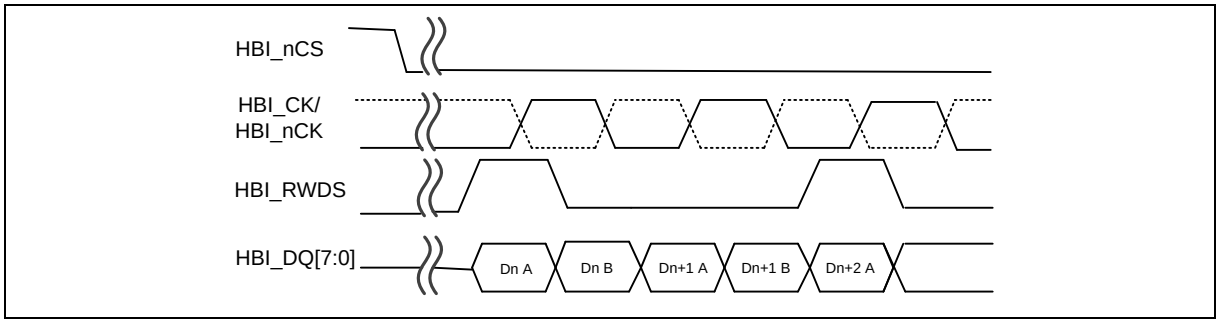


Figure 6.7-6 Data Placement during a Write Transaction

6.7.5.2 HyperBus Operation/Status

The HyperBus Interface control is used to implement each operation mode according the setting value of HYPCMD (HBI_CMD[3:0]). The operation mode including

- Reset HyperRAM
- Exit from Hybrid sleep (HS) and deep Power-down mode (DPD)
- Read/Write HyperRAM register
- Read 1~2 words from HyperRAM
- Write 1~4 bytes to HyperRAM

When an operation is done, the read value of HYPCMD automatically return to 4'b0000 otherwise it is still busy in the current transaction.

Programming example:

1. **Reset HyperRAM**

   Step 1: Configure the HPYCMD (HBI_CMD[3:0]) = 4'b0001
   Step 2: Polling the HPYCMD = 4'b0000

2. **Write 32 bits data into offset 0x100**

   Step 1: Polling the HPYCMD (HBI_CMD[3:0]) = 4'b0000
   Step 2: Set the HBI_CONFIG = 0x00700D89
   Chip select setup time = 2.5 HCLK
   Initial access time = 7 HBI_CK cycles
   Chip select hold time = 2.5 HCLK
   Chip select idle period = 14 HCLK
   Burst group size = 32 Bits
   Data Endian = Little-Endian
   HyperBus clock rate = 1/2 HCLK
   Chip select maximum low time = 71 HCLK
   Step 3: Set the HBI_ADR = 0x100
   Step 4: Set the 32 bits data into HBI_WDATA
   Step 5: Configure the HPYCMD (HBI_CMD[3:0]) = 4'b1111
   Step 6: Polling the HPYCMD = 4'b0000

6.7.5.3 HyperBus Interface Timing

This controller provides registers with different timing parameters to ensure that the timing of access to various types of HyperRAM is correct.

- Reset HyperRAM

Write 4'b0001 to HYPCMD, the controller will send a reset pulse to HBI_nRESET pin about 20 PCLK cycles. The status will read out 4'b0000 from HPYCMD register when the reset command is done (after another 20 PCLK cycles).

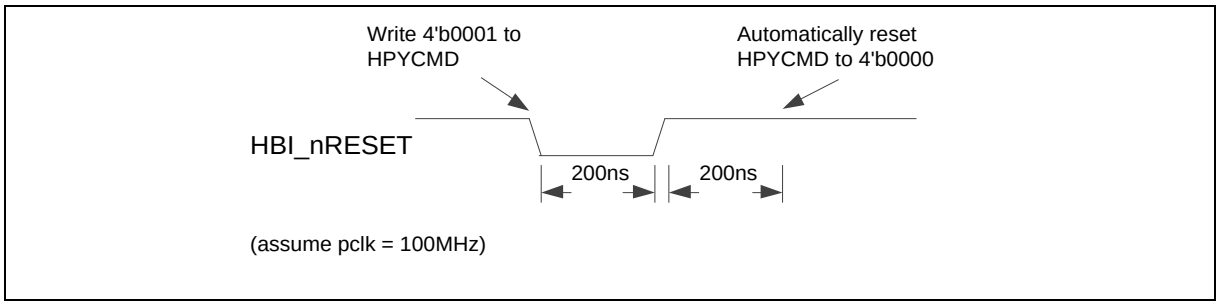


Figure 6.7-7 Chip Reset Command and Timing

- Exit from Hybrid Sleep (HS) and Deep Power Down (DPD)

When the HyperRAM device is not needed for system operation, it may be placed in a very low power consuming state called Deep Power Down (DPD), by writing 0 to bit-15 (CR0[15]) of HyperRAM Configuration Register 0. When CR0[15] is cleared to 0, the device enters the DPD state within the time of tDPDIN for each slave device (For Winbond W956D8MKY, the maximum time of tDPDIN is about 3us) and all refresh operations stop. The data in RAM is lost, (becomes invalid without refresh) during DPD state. To exit DPD, it requires to driving HBI_nCS signal to Low and then High, or a reset. Only HBI_nCS and HBI_nRESET signals are monitored during DPD mode. All register content might lose in Deep Power Down State.

Write 4'b0101 to HYPCMD, the controller will sent at least 1us low pulse (100 PCLK cycles) to HBI_nCS pin to wake-up. The status will be read out as 4'b0000 from HPYCMD register when the exit command is done (after 14900 PCLK cycles).

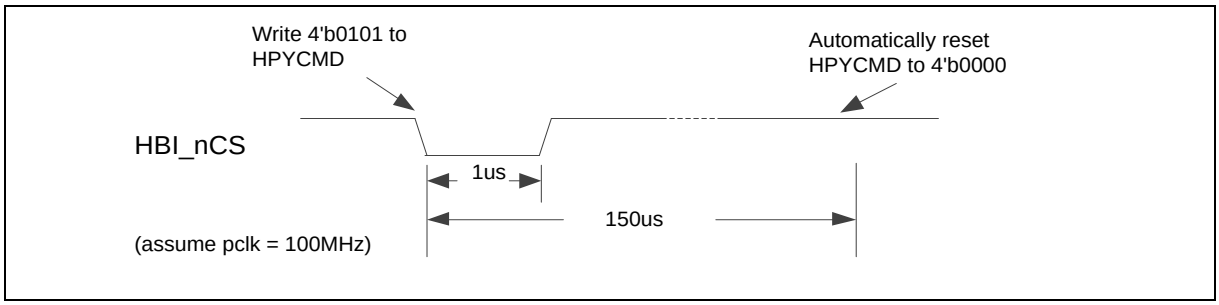


Figure 6.7-8 Exit Timing for HS and DPD

- HyperBus setup/hold time and one transaction period Control

A HyperBus transmission interface must take into account various timings, including the setup time of HBI_nCS at the beginning of HBI_CK, the hold time after HBI_nCK ends, and the initial access time during transmission. These timing variables can all be satisfied by HBI_CONFIG register settings.

Setup Time

The CSS (HBI_CONFIG[1:0]) is used to define the setup time between the HBI_nCS and the first rising edge of HBI_CK.

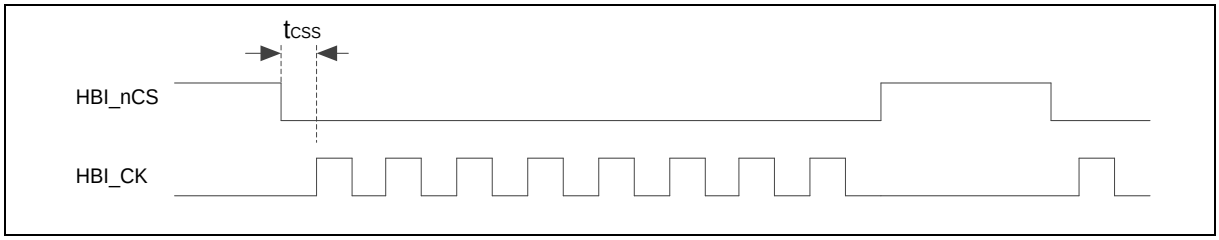


Figure 6.7-9 Chip Select Setup Time to Next HBI_CK Rising Edge

Initial Access Time

Memory Space read and write transactions or Register Space read transactions require some access time to open the row selected by the CA (command/address period). The number of access time needed to be satisfied for the slave device (refer to HyperRAM's tACC specification) depends on the HyperBus frequency and can be vary from 3 to 7 clocks. The value in ACCTS (HBI_CONFIG[5:2]) selects the number of clocks for initial access time. The default value is 7 clocks, allowing for operation up to a maximum frequency of 192 MHz prior to the host system setting a lower initial latency value that may be more optimal for the system.

HBI_RWDS may be High or Low during the CA period. The HBI_RWDS signal goes High during the CA to indicate that an additional initial latency is being inserted to allow a refresh operation to complete before opening the selected row. (Refer to Figure 6.7-2)

The value of ACC (HBI_CONFIG[5:2]) must be set to the same value of initial latency in the HyperRAM devise's Configuration Register 0 (CR0).

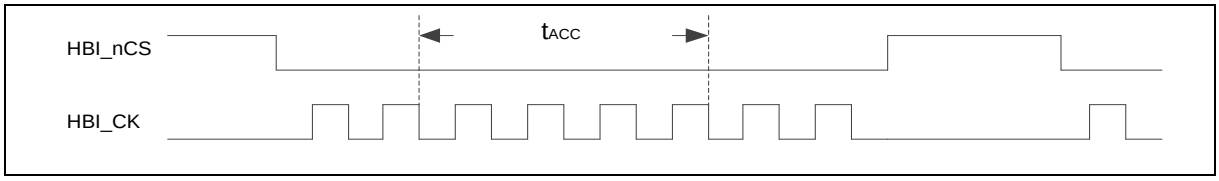


Figure 6.7-10 HyperBus Initial Access Time

Hold Time

The CSH (HBI_CONFIG[7:6]) is used to define the hold time between the HBI_nCS and the last falling edge of HBI_CK.

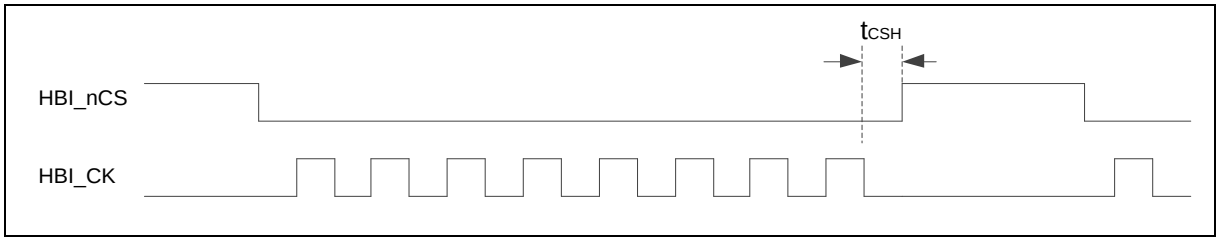


Figure 6.7-11 Chip Select Hold Time to After Last HBI_CK Falling Edge

Chip Select High between Transaction

In HyperRAM, the time period between two transactions must be confirmed. And the Read-Write recovery time of the HyperRAM specification must also be guaranteed. Therefore, CSHI (HBI_CONFIG[11:8]) is a parameter used to confirm such specifications. The Read-Write recovery time equal to the following condition.

$$t_{RWR} = t_{CSHI} + t_{CSS} + 3 \text{ HCLK cycles}$$

6.7.5.4 Memory and Data Map

Up to 32 Mbytes of memory space is supported in this controller. For HyperRAM register space, the HBADDR (HBI_ADR) shall be configured as dedicated content which is described in the register.

The data order of write is listed as Figure 6.7-12.

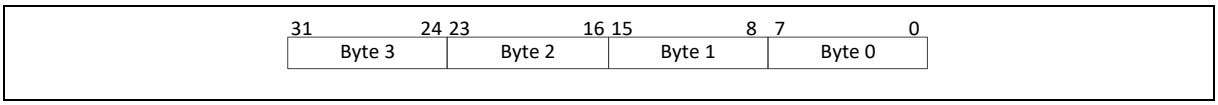


Figure 6.7-12 Data Order for Write

The data order of read is listed as Figure 6.7-13.

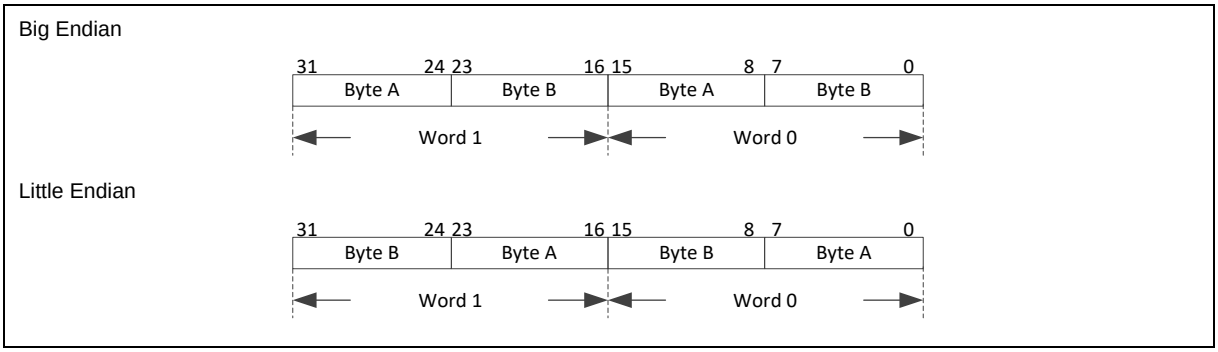


Figure 6.7-13 Data Order for Read

6.7.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write.

Register	Offset	R/W	Description	Reset Value
HBI Base Address: **HBI_BA = 0x400C_E000**				
HBI_CMD	HBI_BA+0x00	R/W	HyperBus Command and Status Register	0x0000_0000
HBI_CONFIG	HBI_BA+0x04	R/W	HyperBus Configuration Register	0x02EC_7849
HBI_ADR	HBI_BA+0x08	R/W	HyperBus Byte Address access Register	0x0000_0000
HBI_WDATA	HBI_BA+0x0C	R/W	HyperBus 32-Bits Write Data Register	0x0000_0000
HBI_RDATA	HBI_BA+0x10	R/W	HyperBus 32-Bits Read Data Register	0x0000_0000
HBI_INTEN	HBI_BA+0x14	R/W	HyperBus Interrupt Enable Register	0x0000_0000
HBI_INTSTS	HBI_BA+0x18	R/W	HyperBus Interrupt Status Register	0x0000_0000

6.7.7 Register Description

HyperBus Command and Status Register (HBI_CMD)

Register	Offset	R/W	Description	Reset Value
HBI_CMD	HBI_BA+0x00	R/W	HyperBus Command and Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**HYPCMD**			

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**HYPCMD**	**HyperBus Command and Status** **Write** 0001 = Reset HyperRAM. 0010 = Read HyperRAM register (16-Bit, Read Data[15:0]. 0101 = Exit From Hybrid Sleep and deep power down. 0111 = Write HyperRAM register (16-Bit, Write Data[15:0]. 1000 = Read 1 word (Read Data[15:0]) from HyperRAM. 1001 = Read 2 words (Read Data[31:0]) from HyperRAM. 1100 = Write 1 Byte (Write Data[7:0]) to HyperRAM. 1101 = Write 2 Bytes (Write Data[15:0]) to HyperRAM. 1110 = Write 3 Bytes (Write Data[23:0]) to HyperRAM. 1111 = Write 4 Bytes (Write Data[31:0]) to HyperRAM. Other value = reserved. **Read** 0000 = HyperBus interface is Idle. Other value = HyperBus interface is busy. **Note:** When an operation is Done, the read value automatically return to 4'b0000.

HyperBus Configuration Register (HBI_CONFIG)

Register	Offset	R/W	Description	Reset Value
HBI_CONFIG	HBI_BA+0x04	R/W	HyperBus Configuration Register	0x02EC_7849

31	30	29	28	27	26	25	24
Reserved					CSMAXLT		
23	22	21	20	19	18	17	16
CSMAXLT							
15	14	13	12	11	10	9	8
Reserved	ENDIAN	BGSIZE		CSHI			
7	6	5	4	3	2	1	0
CSH		ACCT				CSST	

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:16]	**CSMAXLT**	**Chip Select Maximum Low Time** This field indicates the maximum Low period of the chip select (CS#) in one transaction 00000000000 = 1 HCLK cycle. 00000000001 = 2 HCLK cycles. 00000000010 = 3 HCLK cycles. 00000000011 = 4 HCLK cycles. … 01011101100 = 749 HCLK cycles (3.9us @192 MHz). … 11111111110 = 2047 HCLK cycles. 11111111111 = 2048 HCLK cycles. **Note:** This field indicates the timing of HyperRAM Chip Select specification so that it has to relative the frequency of HCLK and the CLKDIV (HBI_CONFIG[15]).
[15]	**Reserved**	Reserved.
[14]	**ENDIAN**	**Endian Condition on the HyperBus Data Pipe** 0 = Little-Endian. Byte A = Bits[7:0] of a 16-Bit .....w.ord Byte B = Bits[15:8] of a 16-Bit .....w.ord 1 = Big-Endian. Byte A = Bits[15:8] of a 16-Bit .....w.ord Byte B = Bits[7:0] of a 16-Bit .....w.ord
[13:12]	**BGSIZE**	**Burst Group Size** This field indicates the burst length on the HyperBus transaction 00 = 128 Bytes. 01 = 64 Bytes. 10 = 16 Bytes. 11 = 32 Bytes.

		Note: This field must be set to the same value as “Burst Length” in HyperRAM’s Configuration Register 0.
[11:8]	**CSHI**	**Chip Select High between Transaction** This field indicates the inactive period between two HyperBus transactions 0000 = 1 HCLK cycle. 0001 = 2 HCLK cycles. 0010 = 3 HCLK cycles. 0011 = 4 HCLK cycles. … 1111 = 16 HCLK cycles. **Note:** This field must meet the HyperRAM device’s specification of tCSHI.
[7:6]	**CSH**	**Chip Select Hold Time After CK Falling Edge** This field indicates the hold time between the last CK falling edge and chip select 00 = 0.5 HCLK cycles. 01 = 1.5 HCLK cycles. 10 = 2.5 HCLK cycles. 11 = 3.5 HCLK cycles.
[5:2]	**ACCT**	**Initial Access Time** This field indicates the initial access cycles of the HyperBus transaction 0000 = 5 CK cycles. 0001 = 6 CK cycles. 0010 = 7 CK cycles. 1110 = 3 CK cycles. 1111 = 4 CK cycles. Others = Reserved. **Note:** This field must be set to the same value as “Initial Latency” in HyperRAM’s Configuration Register 0.
[1:0]	**CSST**	**Chip Select Setup Time to Next CK Rising Edge** This field indicates the setup time between the chip select and the next CK rising edge 00 = 1.5 HCLK cycles. 01 = 2.5 HCLK cycles. 10 = 3.5 HCLK cycles. 11 = 4.5 HCLK cycles.

HyperBus Byte Address Access Register (HBI_ADR)

Register	Offset	R/W	Description	Reset Value
HBI_ADR	HBI_BA+0x08	R/W	HyperBus Byte Address access Register	0x0000_0000

31	30	29	28	27	26	25	24
ADR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ADR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ADR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADR							

Bits	Description	
[31:0]	**ADR**	**HyperBus Byte Address** Memory Space Range: 0x0000_0000 ~ 0x01FF_FFFF Register Space Range: 0x0000_0000 = Identification Registe.r 0 0x0000_0002 = Identification Registe.r 1 0x0000_1000 = Configuration Registe.r 0 0x0000_1002 = Configuration Registe.r 1 **Note:** 1. It is “Byte” address, not “word” address 2. Up to 32M Bytes of memory space is supported.

HyperBus Write Data Register (HBI_WDATA)

Register	Offset	R/W	Description	Reset Value
HBI_WDATA	HBI_BA+0x0C	R/W	HyperBus 32-Bits Write Data Register	0x0000_0000

31	30	29	28	27	26	25	24
WDATA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
WDATA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
WDATA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDATA							

Bits	Description	
[31:0]	**WDATA**	**HyperBus 32-Bits Write Data** To write 1 Byte to HyperRAM, Byte 0 (Data[7:0]) is used To write 2 Bytes to HyperRAM, Byte 1~0 (Data[15:0]) is used To write 3 Bytes to HyperRAM, Byte 2~0 (Data[23:0]) is used To write 4 Bytes to HyperRAM, Byte 3~0 (Data[31:0]) is used

HyperBus Read Data Register (HBI_RDATA)

Register	Offset	R/W	Description	Reset Value
HBI_RDATA	HBI_BA+0x10	R/W	HyperBus 32-Bits Read Data Register	0x0000_0000

31	30	29	28	27	26	25	24
RDATA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RDATA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RDATA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RDATA							

Bits	Description	
[31:0]	**RDATA**	**HyperBus 32-Bits Read Data** 32-Bits Data for HyperBus Read **Note:** The data order depends on the ENDIAN (HBI_CONFIG[14]). Refer to Memory and Data Map section for detailed information.

HyperBus Interrupt Enable Register (HBI_INTEN)

Register	Offset	R/W	Description	Reset Value
HBI_INTEN	HBI_BA+0x14	R/W	HyperBus Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**OPINTEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**OPINTEN**	**HyperBus Operation Done Interrupt Enable** 0 = Operation done interrupt Disabled. 1 = Operation done interrupt Enabled.

HyperBus Interrupt Status Register (HBI_INTSTS)

Register	Offset	R/W	Description	Reset Value
HBI_INTSTS	HBI_BA+0x18	R/W	HyperBus Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**OPDONE**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**OPDONE**	**HyperBus Operation Done Interrupt** 0 = HyperBus operation is busy. 1 = HyperBus operation is done.

6.8 General Purpose I/O (GPIO)

6.8.1 Overview

This chip has up to 146 General Purpose I/O pins to be shared with other function pins depending on the chip configuration. These 146 pins are arranged in 10 ports named as PA, PB, PC, PD, PE, PF, PG, PH, PI and PJ. PA, PB, PE, PG and PH has 16 pins on port. PC, PD has 15 pins on port. PF has 12 pins on port. PI has 10 pins on port. PJ has 14 pins on port. Each of the 146 pins is independent and has the corresponding register bits to control the pin mode function and data.

The I/O type of each of I/O pins can be configured by software individually as Input, Push-pull output, Open-drain output or Quasi-bidirectional mode. After the chip is reset, the I/O mode of all pins are depending on CIOINI (CONFIG0[10]). Please refer to the Datasheet for detailed pin operation voltage information about V_{DD}, V_{DDIO} and V_{BAT} electrical characteristics.

6.8.2 Features

- Four I/O modes:
  - Quasi-bidirectional mode
  - Push-Pull Output mode
  - Open-Drain Output mode
  - Input only with high impendence mode
- TTL/Schmitt trigger input selectable
- I/O pin can be configured as interrupt source with edge/level setting
- Supports High Slew Rate I/O mode
- Configurable default I/O mode of all pins after reset by CIOINI (CONFIG0[10]) setting
  - CIOINI = 0, all GPIO pins in Quasi-bidirectional mode after chip reset
  - CIOINI = 1, all GPIO pins in input mode after chip reset
- Supports independent pull-up and pull-down control
- Enabling the pin interrupt function will also enable the wake-up function
- Supports EINT0~7 edge detect and trigger function

6.8.3 Block Diagram

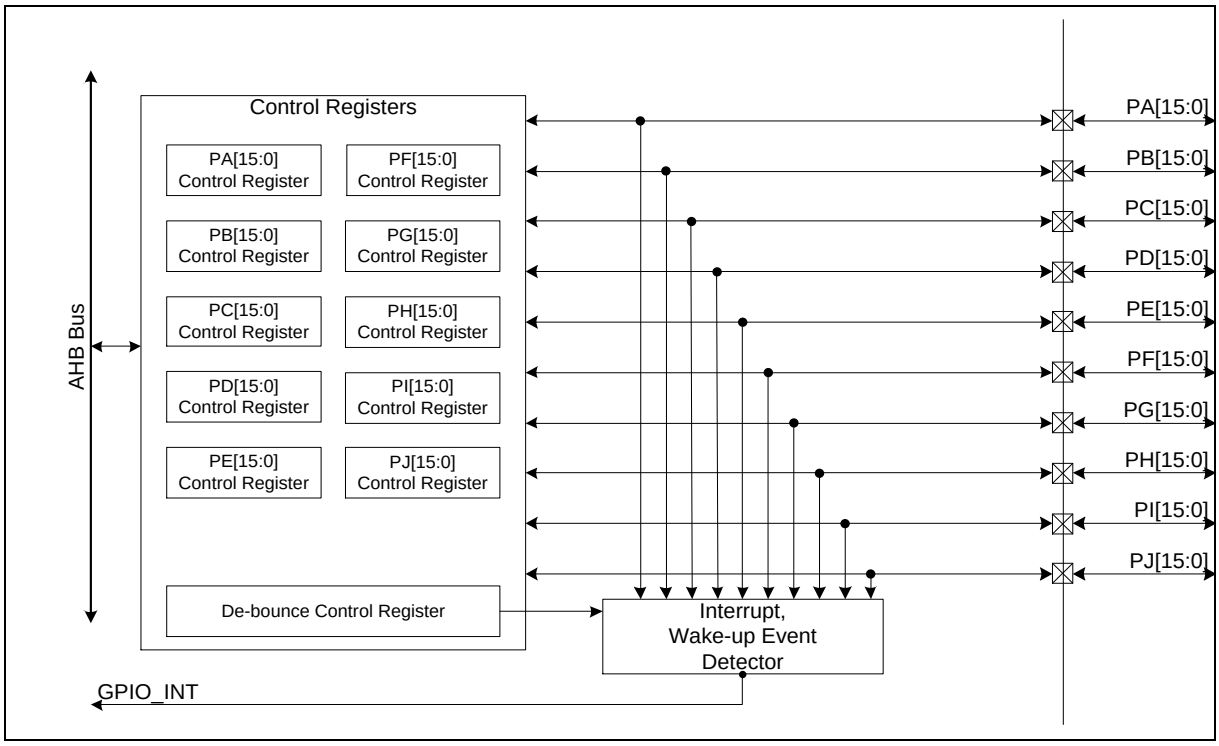


Figure 6.8-1 GPIO Controller Block Diagram

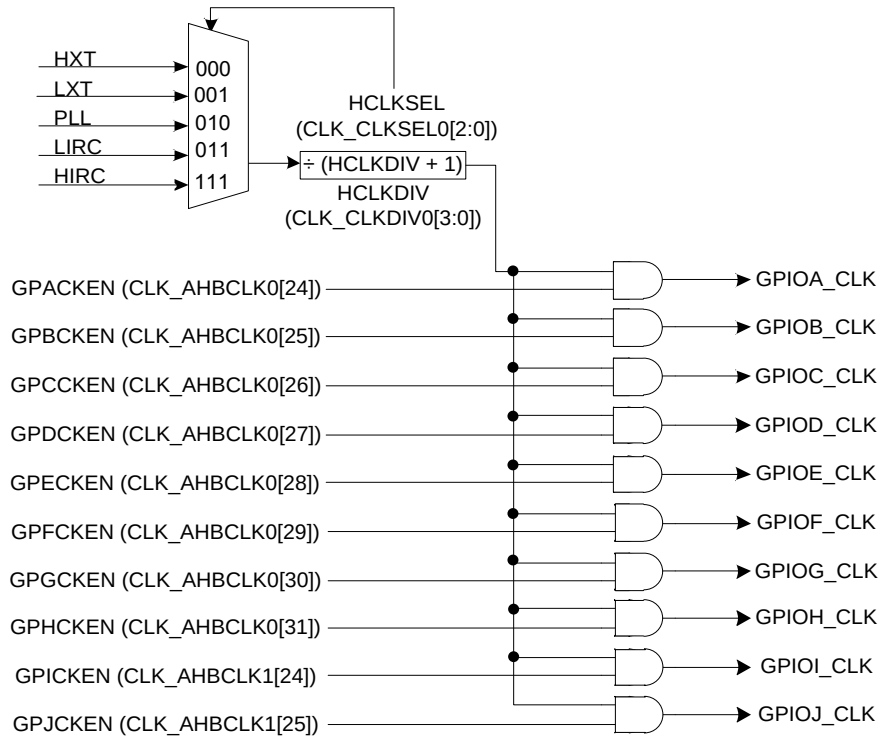


Figure 6.8-2 GPIO Clock Control Diagram

Note: The PC.15/PD.15/PF.12/PF.13/PF.14/PF.15/PI.0/PI.1/ PI.2/PI.3/ PI.4/PI.5/PJ.14/PJ.15 pins are not available.

6.8.4 Basic Configuration

- Clock Source Selection
  - Enable GPIO peripheral clock in CLK_AHBCLK0[31:24] and CLK_AHBCLK1[25:24].
- Reset configuration
  - Reset GPIO in GPIORST SYS_IPRST1[1].

6.8.5 Functional Description

6.8.5.1 Input Mode

Set MODEn (Px_MODE[2n+1:2n]) to 00 as the Px.n pin is in Input mode and the I/O pin is in tri-state (high impedance) without output drive capability. The PIN (Px_PIN[n]) value reflects the status of the

corresponding port pins.

6.8.5.2 Push-pull Output Mode

Figure 6.8-2 shows the diagram of Push-pull Output Mode. Set MODEn (Px_MODE[2n+1:2n]) to 01 as Px.n pin is in Push-pull Output mode and the I/O pin supports digital output function with source/sink current capability. The bit value in the corresponding DOUT (Px_DOUT[n]) is driven on the pin.

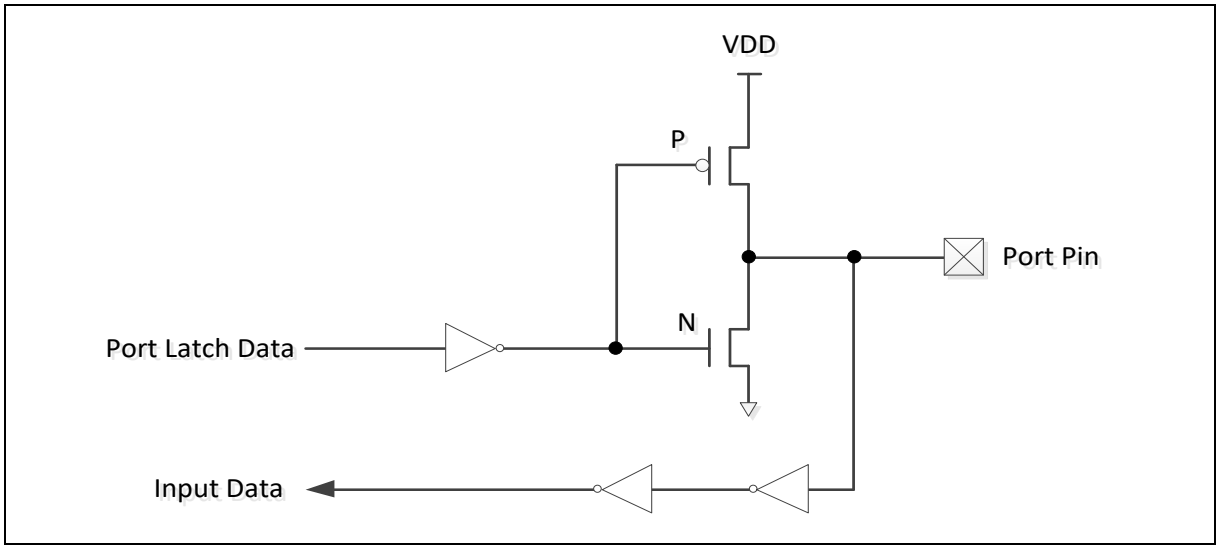


Figure 6.8-2 Push-Pull Output

6.8.5.3 Open-drain Mode

Figure 6.8-3 shows the diagram of Open-drain Mode. Set MODEn (Px_MODE[2n+1:2n]) to 10 and the Px.n pin is in Open-drain mode and the digital output function of I/O pin supports only sink current capability, an external pull-up resistor is needed for driving high state. If the bit value in the corresponding DOUT (Px_DOUT[n]) bit is 0, the pin drives a low output on the pin. If the bit value in the corresponding DOUT (Px_DOUT[n]) bit is 1, the pin output drives high that is controlled by external pull high resistor.

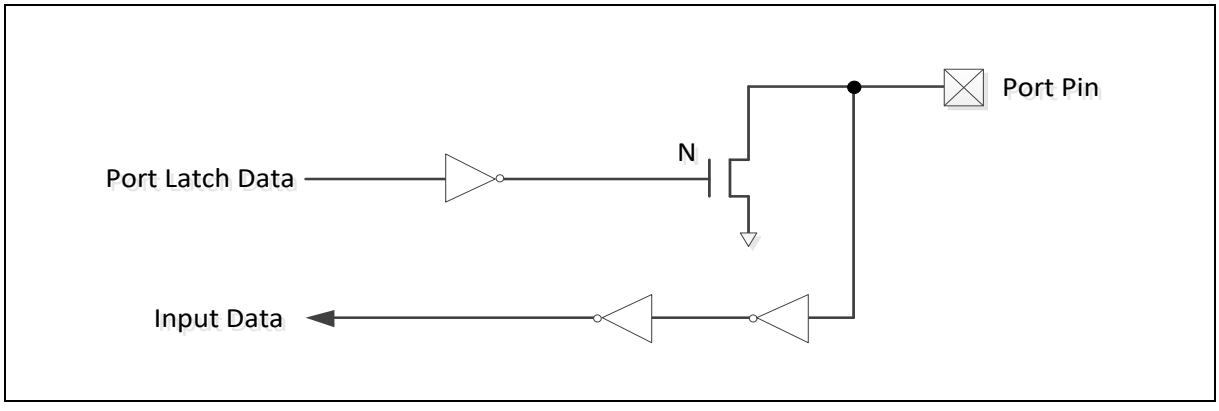


Figure 6.8-3 Open-Drain Output

6.8.5.4 Quasi-bidirectional Mode

Figure 6.8-4 shows the diagram of Quasi-bidirectional Mode. Set MODEn (Px_MODE[2n+1:2n]) to 11 as the Px.n pin is in Quasi-bidirectional mode and the I/O pin supports digital output and input function at the same time but the source current is only up to hundreds uA. Before the digital input function is performed the corresponding DOUT (Px_DOUT[n]) bit must be set to 1. The quasi-bidirectional output is common on the 80C51 and most of its derivatives. If the bit value in the corresponding DOUT

(Px_DOUT[n]) bit is 0, the pin drives a low output on the pin. If the bit value in the corresponding DOUT (Px_DOUT[n]) bit is 1, the pin will check the pin value. If pin value is high, no action takes. If pin state is low, the pin will drive strong high with 2 clock cycles on the pin and then disable the strong output drive. Meanwhile, the pin status is controlled by internal pull-up resistor. Note that the source current capability in quasi-bidirectional mode, please refer to the Datasheet for detailed information. .

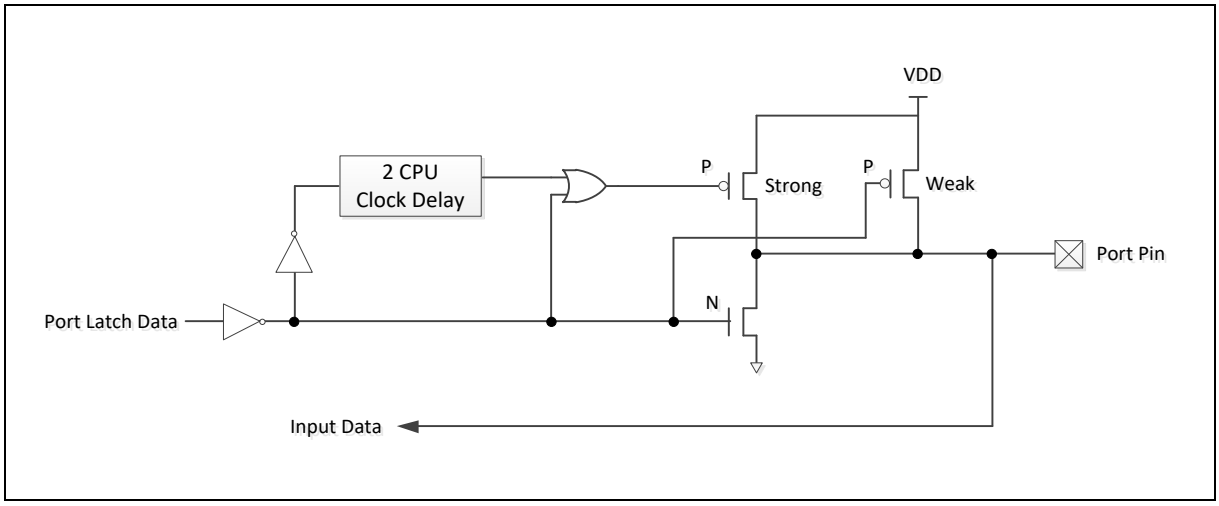


Figure 6.8-4 Quasi-Bidirectional I/O Mode

6.8.5.5 GPIO Interrupt and Wake-up Function

Each GPIO pin can be set as chip interrupt source by setting correlative RHIEN (Px_INTEN[n+16])/ FLIEN (Px_INTEN[n]) bit and TYPE (Px_INTTYPE[n]). There are five types of interrupt condition can be selected: low level trigger, high level trigger, falling edge trigger, rising edge trigger and both rising and falling edge trigger. The GPIO can also be the chip wake-up source when chip enters Idle/Power-down mode. The setting of wake-up trigger condition is the same as GPIO interrupt trigger.

6.8.5.6 GPIO De-bounce Function

GPIO de-bounce function can be used to sample interrupt input for each GPIO pin and prevent unexpected interrupt happened which caused by noise. GPIO de-bounce function only supports edge detection trigger type, but it is not supported in Power-down mode. For edge trigger condition, there are three types of interrupt condition can be selected for de-bounce function: falling edge trigger, rising edge trigger and both rising and falling edge trigger by setting correlative RHIEN (Px_INTEN[n+16])/ FLIEN (Px_INTEN[n]) bit and TYPE (Px_INTTYPE[n]). If user wants to use de-bounce function, de-bounce enable control register Px_DBEN must be set for corresponding GPIO pin. The de-bounce clock source can be HCLK or LIRC (10 kHz) by setting DBCLKSRC (Px_DBCTL[4]) register. And DBCLKSEL (Px_DBCTL[3:0]) register can control sampling cycle period.

Figure 6.8-5 shows GPIO rising edge trigger interrupt. The interval of time between the two valid sample signal is determined by DBCLKSRC (Px_DBCTL[4]) and DBCLKSEL (Px_DBCTL[3:0]). Each valid data from GPIO pin need to be sample twice. For rising edge setting, if pin status is low before setting DBEN (Px_DBEN), interrupt will happen when generating a pin high valid data. But, if pin status is high before setting DBEN (Px_DBEN), interrupt will happen when generating a pin low valid data first, and then generating a pin high valid data. For falling edge trigger, Figure 6.8-6 shows the situation is opposite to rising edge trigger.

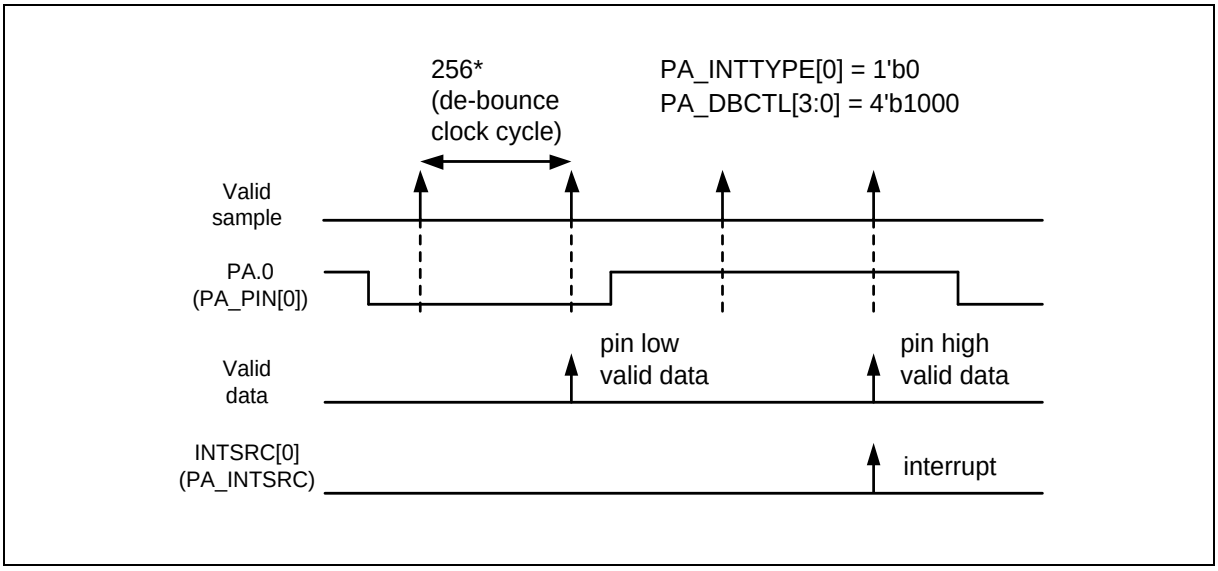


Figure 6.8-5 GPIO Rising Edge Trigger Interrupt

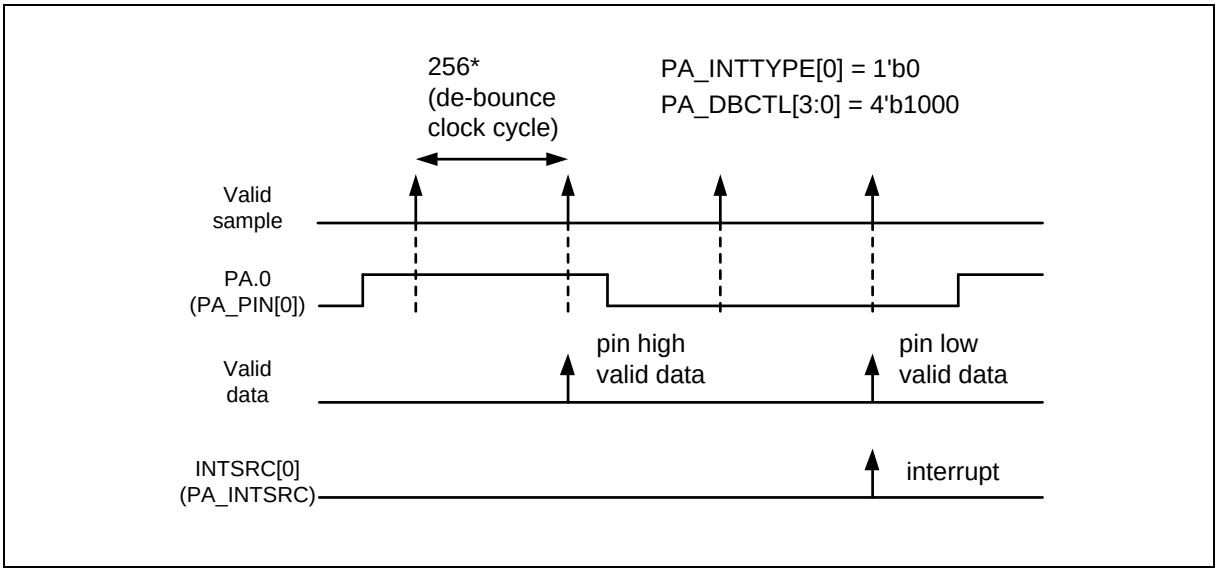


Figure 6.8-6 GPIO Falling Edge Trigger Interrupt

6.8.5.7 INT Edge Detect and Trigger Function

There are many INT0~7 GPIO pins that can be selected by multi-function. Each INT pin will be filtered by a 3-bit noise filter. User can enable the noise filter function by NFEN (INTn_INNF[0], n=0~7) bit, and noise filter sampling clock can be selected by setting NFSEL (INTn_INNF[6:4], n=0~7) bits to fit different noise properties. Moreover, by setting the NFCNT (INTn_INNF[10:8], n=0~7) bits, user can define by how many sampling clock cycles a filter will recognize the effective edge of the INT pin signal.

Each INT pin can be detected by setting EDETCTLn (INT_EDETCTL[2n+1:2n]) and EDIFn (INT_EDSTS[n], n=0~7) will be set if INT pin matches condition. Interrupt will happen if user enables EDIENn (INT_EDINTEN[n], n=0~7) and INT pin match condition.

In addition to detect function, it can also trigger EPWM and TIMER (PWM mode) if INT pin matches condition. More detailed setting is in EPWM or TIMER (PWM mode) “External Pin Event Trigger” section.

6.8.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
GPIO Base Address: **GPIO_BA = 0x4000_4000**				
PA_MODE	GPIO_BA+0x000	R/W	PA I/O Mode Control	0xXXXX_XXXX
PA_DINOFF	GPIO_BA+0x004	R/W	PA Digital Input Path Disable Control	0x0000_0000
PA_DOUT	GPIO_BA+0x008	R/W	PA Data Output Value	0x0000_FFFF
PA_DATMSK	GPIO_BA+0x00C	R/W	PA Data Output Write Mask	0x0000_0000
PA_PIN	GPIO_BA+0x010	R	PA Pin Value	0x0000_XXXX
PA_DBEN	GPIO_BA+0x014	R/W	PA De-bounce Enable Control Register	0x0000_0000
PA_INTTYPE	GPIO_BA+0x018	R/W	PA Interrupt Trigger Type Control	0x0000_0000
PA_INTEN	GPIO_BA+0x01C	R/W	PA Interrupt Enable Control Register	0x0000_0000
PA_INTSRC	GPIO_BA+0x020	R/W	PA Interrupt Source Flag	0x0000_XXXX
PA_SMTEN	GPIO_BA+0x024	R/W	PA Input Schmitt Trigger Enable Register	0x0000_0000
PA_SLEWCTL	GPIO_BA+0x028	R/W	PA High Slew Rate Control Register	0x0000_0000
PA_PUSEL	GPIO_BA+0x030	R/W	PA Pull-up and Pull-down Selection Register	0x0000_0000
PA_DBCTL	GPIO_BA+0x034	R/W	PA Interrupt De-bounce Control Register	0x0000_0020
PB_MODE	GPIO_BA+0x040	R/W	PB I/O Mode Control	0xXXXX_XXXX
PB_DINOFF	GPIO_BA+0x044	R/W	PB Digital Input Path Disable Control	0x0000_0000
PB_DOUT	GPIO_BA+0x048	R/W	PB Data Output Value	0x0000_FFFF
PB_DATMSK	GPIO_BA+0x04C	R/W	PB Data Output Write Mask	0x0000_0000
PB_PIN	GPIO_BA+0x050	R	PB Pin Value	0x0000_XXXX
PB_DBEN	GPIO_BA+0x054	R/W	PB De-bounce Enable Control Register	0x0000_0000
PB_INTTYPE	GPIO_BA+0x058	R/W	PB Interrupt Trigger Type Control	0x0000_0000
PB_INTEN	GPIO_BA+0x05C	R/W	PB Interrupt Enable Control Register	0x0000_0000
PB_INTSRC	GPIO_BA+0x060	R/W	PB Interrupt Source Flag	0x0000_XXXX
PB_SMTEN	GPIO_BA+0x064	R/W	PB Input Schmitt Trigger Enable Register	0x0000_0000
PB_SLEWCTL	GPIO_BA+0x068	R/W	PB High Slew Rate Control Register	0x0000_0000
PB_PUSEL	GPIO_BA+0x070	R/W	PB Pull-up and Pull-down Selection Register	0x0000_0000
PB_DBCTL	GPIO_BA+0x074	R/W	PB Interrupt De-bounce Control Register	0x0000_0020
PC_MODE	GPIO_BA+0x080	R/W	PC I/O Mode Control	0xXXXX_XXXX

PC_DINOFF	GPIO_BA+0x084	R/W	PC Digital Input Path Disable Control	0x0000_0000
PC_DOUT	GPIO_BA+0x088	R/W	PC Data Output Value	0x0000_7FFF
PC_DATMSK	GPIO_BA+0x08C	R/W	PC Data Output Write Mask	0x0000_0000
PC_PIN	GPIO_BA+0x090	R	PC Pin Value	0x0000_XXXX
PC_DBEN	GPIO_BA+0x094	R/W	PC De-bounce Enable Control Register	0x0000_0000
PC_INTTYPE	GPIO_BA+0x098	R/W	PC Interrupt Trigger Type Control	0x0000_0000
PC_INTEN	GPIO_BA+0x09C	R/W	PC Interrupt Enable Control Register	0x0000_0000
PC_INTSRC	GPIO_BA+0x0A0	R/W	PC Interrupt Source Flag	0x0000_XXXX
PC_SMTEN	GPIO_BA+0x0A4	R/W	PC Input Schmitt Trigger Enable Register	0x0000_0000
PC_SLEWCTL	GPIO_BA+0x0A8	R/W	PC High Slew Rate Control Register	0x0000_0000
PC_PUSEL	GPIO_BA+0x0B0	R/W	PC Pull-up and Pull-down Selection Register	0x0000_0000
PC_DBCTL	GPIO_BA+0x0B4	R/W	PC Interrupt De-bounce Control Register	0x0000_0020
PD_MODE	GPIO_BA+0x0C0	R/W	PD I/O Mode Control	0xXXXX_XXXX
PD_DINOFF	GPIO_BA+0x0C4	R/W	PD Digital Input Path Disable Control	0x0000_0000
PD_DOUT	GPIO_BA+0x0C8	R/W	PD Data Output Value	0x0000_7FFF
PD_DATMSK	GPIO_BA+0x0CC	R/W	PD Data Output Write Mask	0x0000_0000
PD_PIN	GPIO_BA+0x0D0	R	PD Pin Value	0x0000_XXXX
PD_DBEN	GPIO_BA+0x0D4	R/W	PD De-bounce Enable Control Register	0x0000_0000
PD_INTTYPE	GPIO_BA+0x0D8	R/W	PD Interrupt Trigger Type Control	0x0000_0000
PD_INTEN	GPIO_BA+0x0DC	R/W	PD Interrupt Enable Control Register	0x0000_0000
PD_INTSRC	GPIO_BA+0x0E0	R/W	PD Interrupt Source Flag	0x0000_XXXX
PD_SMTEN	GPIO_BA+0x0E4	R/W	PD Input Schmitt Trigger Enable Register	0x0000_0000
PD_SLEWCTL	GPIO_BA+0x0E8	R/W	PD High Slew Rate Control Register	0x0000_0000
PD_PUSEL	GPIO_BA+0x0F0	R/W	PD Pull-up and Pull-down Selection Register	0x0000_0000
PD_DBCTL	GPIO_BA+0x0F4	R/W	PD Interrupt De-bounce Control Register	0x0000_0020
PE_MODE	GPIO_BA+0x100	R/W	PE I/O Mode Control	0xXXXX_XXXX
PE_DINOFF	GPIO_BA+0x104	R/W	PE Digital Input Path Disable Control	0x0000_0000
PE_DOUT	GPIO_BA+0x108	R/W	PE Data Output Value	0x0000_FFFF
PE_DATMSK	GPIO_BA+0x10C	R/W	PE Data Output Write Mask	0x0000_0000
PE_PIN	GPIO_BA+0x110	R	PE Pin Value	0x0000_XXXX
PE_DBEN	GPIO_BA+0x114	R/W	PE De-bounce Enable Control Register	0x0000_0000

PE_INTTYPE	GPIO_BA+0x118	R/W	PE Interrupt Trigger Type Control	0x0000_0000
PE_INTEN	GPIO_BA+0x11C	R/W	PE Interrupt Enable Control Register	0x0000_0000
PE_INTSRC	GPIO_BA+0x120	R/W	PE Interrupt Source Flag	0x0000_XXXX
PE_SMTEN	GPIO_BA+0x124	R/W	PE Input Schmitt Trigger Enable Register	0x0000_0000
PE_SLEWCTL	GPIO_BA+0x128	R/W	PE High Slew Rate Control Register	0x0000_0000
PE_PUSEL	GPIO_BA+0x130	R/W	PE Pull-up and Pull-down Selection Register	0x0000_0000
PE_DBCTL	GPIO_BA+0x134	R/W	PE Interrupt De-bounce Control Register	0x0000_0020
PF_MODE	GPIO_BA+0x140	R/W	PF I/O Mode Control	0xXXXX_XXXX
PF_DINOFF	GPIO_BA+0x144	R/W	PF Digital Input Path Disable Control	0x0000_0000
PF_DOUT	GPIO_BA+0x148	R/W	PF Data Output Value	0x0000_0FFF
PF_DATMSK	GPIO_BA+0x14C	R/W	PF Data Output Write Mask	0x0000_0000
PF_PIN	GPIO_BA+0x150	R	PF Pin Value	0x0000_XXXX
PF_DBEN	GPIO_BA+0x154	R/W	PF De-bounce Enable Control Register	0x0000_0000
PF_INTTYPE	GPIO_BA+0x158	R/W	PF Interrupt Trigger Type Control	0x0000_0000
PF_INTEN	GPIO_BA+0x15C	R/W	PF Interrupt Enable Control Register	0x0000_0000
PF_INTSRC	GPIO_BA+0x160	R/W	PF Interrupt Source Flag	0x0000_XXXX
PF_SMTEN	GPIO_BA+0x164	R/W	PF Input Schmitt Trigger Enable Register	0x0000_0000
PF_SLEWCTL	GPIO_BA+0x168	R/W	PF High Slew Rate Control Register	0x0000_0000
PF_PUSEL	GPIO_BA+0x170	R/W	PF Pull-up and Pull-down Selection Register	0x0000_0000
PF_DBCTL	GPIO_BA+0x174	R/W	PF Interrupt De-bounce Control Register	0x0000_0020
PG_MODE	GPIO_BA+0x180	R/W	PG I/O Mode Control	0xXXXX_XXXX
PG_DINOFF	GPIO_BA+0x184	R/W	PG Digital Input Path Disable Control	0x0000_0000
PG_DOUT	GPIO_BA+0x188	R/W	PG Data Output Value	0x0000_FFFF
PG_DATMSK	GPIO_BA+0x18C	R/W	PG Data Output Write Mask	0x0000_0000
PG_PIN	GPIO_BA+0x190	R	PG Pin Value	0x0000_XXXX
PG_DBEN	GPIO_BA+0x194	R/W	PG De-bounce Enable Control Register	0x0000_0000
PG_INTTYPE	GPIO_BA+0x198	R/W	PG Interrupt Trigger Type Control	0x0000_0000
PG_INTEN	GPIO_BA+0x19C	R/W	PG Interrupt Enable Control Register	0x0000_0000
PG_INTSRC	GPIO_BA+0x1A0	R/W	PG Interrupt Source Flag	0x0000_XXXX
PG_SMTEN	GPIO_BA+0x1A4	R/W	PG Input Schmitt Trigger Enable Register	0x0000_0000
PG_SLEWCTL	GPIO_BA+0x1A8	R/W	PG High Slew Rate Control Register	0x0000_0000

PG_PUSEL	GPIO_BA+0x1B0	R/W	PG Pull-up and Pull-down Selection Register	0x0000_0000
PG_DBCTL	GPIO_BA+0x1B4	R/W	PG Interrupt De-bounce Control Register	0x0000_0020
PH_MODE	GPIO_BA+0x1C0	R/W	PH I/O Mode Control	0xXXXX_XXXX
PH_DINOFF	GPIO_BA+0x1C4	R/W	PH Digital Input Path Disable Control	0x0000_0000
PH_DOUT	GPIO_BA+0x1C8	R/W	PH Data Output Value	0x0000_FFFF
PH_DATMSK	GPIO_BA+0x1CC	R/W	PH Data Output Write Mask	0x0000_0000
PH_PIN	GPIO_BA+0x1D0	R	PH Pin Value	0x0000_XXXX
PH_DBEN	GPIO_BA+0x1D4	R/W	PH De-bounce Enable Control Register	0x0000_0000
PH_INTTYPE	GPIO_BA+0x1D8	R/W	PH Interrupt Trigger Type Control	0x0000_0000
PH_INTEN	GPIO_BA+0x1DC	R/W	PH Interrupt Enable Control Register	0x0000_0000
PH_INTSRC	GPIO_BA+0x1E0	R/W	PH Interrupt Source Flag	0x0000_XXXX
PH_SMTEN	GPIO_BA+0x1E4	R/W	PH Input Schmitt Trigger Enable Register	0x0000_0000
PH_SLEWCTL	GPIO_BA+0x1E8	R/W	PH High Slew Rate Control Register	0x0000_0000
PH_PUSEL	GPIO_BA+0x1F0	R/W	PH Pull-up and Pull-down Selection Register	0x0000_0000
PH_DBCTL	GPIO_BA+0x1F4	R/W	PH Interrupt De-bounce Control Register	0x0000_0020
PI_MODE	GPIO_BA+0x200	R/W	PI I/O Mode Control	0xXXXX_XXXX
PI_DINOFF	GPIO_BA+0x204	R/W	PI Digital Input Path Disable Control	0x0000_0000
PI_DOUT	GPIO_BA+0x208	R/W	PI Data Output Value	0x0000_FFC0
PI_DATMSK	GPIO_BA+0x20C	R/W	PI Data Output Write Mask	0x0000_0000
PI_PIN	GPIO_BA+0x210	R	PI Pin Value	0x0000_XXXX
PI_DBEN	GPIO_BA+0x214	R/W	PI De-bounce Enable Control Register	0x0000_0000
PI_INTTYPE	GPIO_BA+0x218	R/W	PI Interrupt Trigger Type Control	0x0000_0000
PI_INTEN	GPIO_BA+0x21C	R/W	PI Interrupt Enable Control Register	0x0000_0000
PI_INTSRC	GPIO_BA+0x220	R/W	PI Interrupt Source Flag	0x0000_XXXX
PI_SMTEN	GPIO_BA+0x224	R/W	PI Input Schmitt Trigger Enable Register	0x0000_0000
PI_SLEWCTL	GPIO_BA+0x228	R/W	PI High Slew Rate Control Register	0x0000_0000
PI_PUSEL	GPIO_BA+0x230	R/W	PI Pull-up and Pull-down Selection Register	0x0000_0000
PI_DBCTL	GPIO_BA+0x234	R/W	PI Interrupt De-bounce Control Register	0x0000_0020
PJ_MODE	GPIO_BA+0x240	R/W	PJ I/O Mode Control	0xXXXX_XXXX
PJ_DINOFF	GPIO_BA+0x244	R/W	PJ Digital Input Path Disable Control	0x0000_0000
PJ_DOUT	GPIO_BA+0x248	R/W	PJ Data Output Value	0x0000_3FFF

PJ_DATMSK	GPIO_BA+0x24C	R/W	PJ Data Output Write Mask	0x0000_0000
PJ_PIN	GPIO_BA+0x250	R	PJ Pin Value	0x0000_XXXX
PJ_DBEN	GPIO_BA+0x254	R/W	PJ De-bounce Enable Control Register	0x0000_0000
PJ_INTTYPE	GPIO_BA+0x258	R/W	PJ Interrupt Trigger Type Control	0x0000_0000
PJ_INTEN	GPIO_BA+0x25C	R/W	PJ Interrupt Enable Control Register	0x0000_0000
PJ_INTSRC	GPIO_BA+0x260	R/W	PJ Interrupt Source Flag	0x0000_XXXX
PJ_SMTEN	GPIO_BA+0x264	R/W	PJ Input Schmitt Trigger Enable Register	0x0000_0000
PJ_SLEWCTL	GPIO_BA+0x268	R/W	PJ High Slew Rate Control Register	0x0000_0000
PJ_PUSEL	GPIO_BA+0x270	R/W	PJ Pull-up and Pull-down Selection Register	0x0000_0000
PJ_DBCTL	GPIO_BA+0x274	R/W	PJ Interrupt De-bounce Control Register	0x0000_0020
INT0_INNF	GPIO_BA+0x450	R/W	INT0 Input Noise Filter Register	0x0000_0000
INT1_INNF	GPIO_BA+0x454	R/W	INT1 Input Noise Filter Register	0x0000_0000
INT2_INNF	GPIO_BA+0x458	R/W	INT2 Input Noise Filter Register	0x0000_0000
INT3_INNF	GPIO_BA+0x45C	R/W	INT3 Input Noise Filter Register	0x0000_0000
INT4_INNF	GPIO_BA+0x460	R/W	INT4 Input Noise Filter Register	0x0000_0000
INT5_INNF	GPIO_BA+0x464	R/W	INT5 Input Noise Filter Register	0x0000_0000
INT6_INNF	GPIO_BA+0x468	R/W	INT6 Input Noise Filter Register	0x0000_0000
INT7_INNF	GPIO_BA+0x46C	R/W	INT7 Input Noise Filter Register	0x0000_0000
INT_EDETCTL	GPIO_BA+0x490	R/W	INT Edge Detect Control Register	0x0000_0000
INT_EDINTEN	GPIO_BA+0x498	R/W	INT Edge Detect Interrupt Enable Control Register	0x0000_0000
INT_EDSTS	GPIO_BA+0x49C	R/W	INT Edge Detect Interrupt Flag Register	0x0000_0000
PAn_PDIO n=0,1..15	GPIO_BA+0x800+(0x04 * n)	R/W	GPIO PA.n Pin Data Input/Output Register	0x0000_000X
PBn_PDIO n=0,1..15	GPIO_BA+0x840+(0x04 * n)	R/W	GPIO PB.n Pin Data Input/Output Register	0x0000_000X
PCn_PDIO n=0,1..14	GPIO_BA+0x880+(0x04 * n)	R/W	GPIO PC.n Pin Data Input/Output Register	0x0000_000X
PDn_PDIO n=0,1..14	GPIO_BA+0x8C0+(0x04 * n)	R/W	GPIO PD.n Pin Data Input/Output Register	0x0000_000X
PEn_PDIO n=0,1..15	GPIO_BA+0x900+(0x04 * n)	R/W	GPIO PE.n Pin Data Input/Output Register	0x0000_000X
PFn_PDIO n=0,1..11	GPIO_BA+0x940+(0x04 * n)	R/W	GPIO PF.n Pin Data Input/Output Register	0x0000_000X
PGn_PDIO	GPIO_BA+0x980+(0x04 * n)	R/W	GPIO PG.n Pin Data Input/Output Register	0x0000_000X

n=0,1..15				
PHn_PDIO **n=0,1..15**	GPIO_BA+0x9C0+(0x04 * n)	R/W	GPIO PH.n Pin Data Input/Output Register	0x0000_000X
PIn_PDIO **n=6,7..15**	GPIO_BA+0xA00+(0x04 * n)	R/W	GPIO PI.n Pin Data Input/Output Register	0x0000_000X
PJn_PDIO **n=0,1..13**	GPIO_BA+0xA40+(0x04 * n)	R/W	GPIO PJ.n Pin Data Input/Output Register	0x0000_000X

6.8.7 Register Description

Port A-J I/O Mode Control (Px_MODE)

Register	Offset	R/W	Description	Reset Value
PA_MODE	GPIO_BA+0x000	R/W	PA I/O Mode Control	0xXXXX_XXXX
PB_MODE	GPIO_BA+0x040	R/W	PB I/O Mode Control	0xXXXX_XXXX
PC_MODE	GPIO_BA+0x080	R/W	PC I/O Mode Control	0xXXXX_XXXX
PD_MODE	GPIO_BA+0x0C0	R/W	PD I/O Mode Control	0xXXXX_XXXX
PE_MODE	GPIO_BA+0x100	R/W	PE I/O Mode Control	0xXXXX_XXXX
PF_MODE	GPIO_BA+0x140	R/W	PF I/O Mode Control	0xXXXX_XXXX
PG_MODE	GPIO_BA+0x180	R/W	PG I/O Mode Control	0xXXXX_XXXX
PH_MODE	GPIO_BA+0x1C0	R/W	PH I/O Mode Control	0xXXXX_XXXX
PI_MODE	GPIO_BA+0x200	R/W	PI I/O Mode Control	0xXXXX_XXXX
PJ_MODE	GPIO_BA+0x240	R/W	PJ I/O Mode Control	0xXXXX_XXXX

31	30	29	28	27	26	25	24
MODE15		MODE14		MODE13		MODE12	
23	22	21	20	19	18	17	16
MODE11		MODE10		MODE9		MODE8	
15	14	13	12	11	10	9	8
MODE7		MODE6		MODE5		MODE4	
7	6	5	4	3	2	1	0
MODE3		MODE2		MODE1		MODE0	

Bits	Description	
[2n+1:2n] n=0,1..15	**MODEn**	**Port A-J I/O Pin[n] Mode Control** Determine each I/O mode of Px.n pins. 00 = Px.n is in Input mode. 01 = Px.n is in Push-pull Output mode. 10 = Px.n is in Open-drain Output mode. 11 = Px.n is in Quasi-bidirectional mode. **Note 1:** The initial value of this field is defined by CIOINI (CONFIG0 [10]). If CIOINI is set to 0, the default value is 0xFFFF_FFFF and all pins will be quasi-bidirectional mode after chip powered on. If CIOINI is set to 1, the default value is 0x0000_0000 and all pins will be input mode after chip powered on. **Note 2:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J Digital Input Path Disable Control (Px_DINOFF)

Register	Offset	R/W	Description	Reset Value
PA_DINOFF	GPIO_BA+0x004	R/W	PA Digital Input Path Disable Control	0x0000_0000
PB_DINOFF	GPIO_BA+0x044	R/W	PB Digital Input Path Disable Control	0x0000_0000
PC_DINOFF	GPIO_BA+0x084	R/W	PC Digital Input Path Disable Control	0x0000_0000
PD_DINOFF	GPIO_BA+0x0C4	R/W	PD Digital Input Path Disable Control	0x0000_0000
PE_DINOFF	GPIO_BA+0x104	R/W	PE Digital Input Path Disable Control	0x0000_0000
PF_DINOFF	GPIO_BA+0x144	R/W	PF Digital Input Path Disable Control	0x0000_0000
PG_DINOFF	GPIO_BA+0x184	R/W	PG Digital Input Path Disable Control	0x0000_0000
PH_DINOFF	GPIO_BA+0x1C4	R/W	PH Digital Input Path Disable Control	0x0000_0000
PI_DINOFF	GPIO_BA+0x204	R/W	PI Digital Input Path Disable Control	0x0000_0000
PJ_DINOFF	GPIO_BA+0x244	R/W	PJ Digital Input Path Disable Control	0x0000_0000

31	30	29	28	27	26	25	24
DINOFF							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DINOFF							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[n+16] n=0,1..15	**DINOFF[n]**	**Port A-J Pin[n] Digital Input Path Disable Bit** Each of these bits is used to control if the digital input path of corresponding Px.n pin is disabled. If input is analog signal, users can disable Px.n digital input path to avoid input current leakage. 0 = Px.n digital input path Enabled. 1 = Px.n digital input path Disabled (digital input tied to low). **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.
[15:0]	**Reserved**	Reserved.

Port A-J Data Output Value (Px_DOUT)

Register	Offset	R/W	Description	Reset Value
PA_DOUT	GPIO_BA+0x008	R/W	PA Data Output Value	0x0000_FFFF
PB_DOUT	GPIO_BA+0x048	R/W	PB Data Output Value	0x0000_FFFF
PC_DOUT	GPIO_BA+0x088	R/W	PC Data Output Value	0x0000_7FFF
PD_DOUT	GPIO_BA+0x0C8	R/W	PD Data Output Value	0x0000_7FFF
PE_DOUT	GPIO_BA+0x108	R/W	PE Data Output Value	0x0000_FFFF
PF_DOUT	GPIO_BA+0x148	R/W	PF Data Output Value	0x0000_0FFF
PG_DOUT	GPIO_BA+0x188	R/W	PG Data Output Value	0x0000_FFFF
PH_DOUT	GPIO_BA+0x1C8	R/W	PH Data Output Value	0x0000_FFFF
PI_DOUT	GPIO_BA+0x208	R/W	PI Data Output Value	0x0000_FFC0
PJ_DOUT	GPIO_BA+0x248	R/W	PJ Data Output Value	0x0000_3FFF

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DOUT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DOUT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**DOUT[n]**	**Port A-J Pin[n] Output Value** Each of these bits controls the status of a Px.n pin when the Px.n is configured as Push-pull output, Open-drain output or Quasi-bidirectional mode. 0 = Px.n will drive Low if the Px.n pin is configured as Push-pull output, Open-drain output or Quasi-bidirectional mode. 1 = Px.n will drive High if the Px.n pin is configured as Push-pull output or Quasi-bidirectional mode. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J Data Output Write Mask (Px_DATMSK)

Register	Offset	R/W	Description	Reset Value
PA_DATMSK	GPIO_BA+0x00C	R/W	PA Data Output Write Mask	0x0000_0000
PB_DATMSK	GPIO_BA+0x04C	R/W	PB Data Output Write Mask	0x0000_0000
PC_DATMSK	GPIO_BA+0x08C	R/W	PC Data Output Write Mask	0x0000_0000
PD_DATMSK	GPIO_BA+0x0CC	R/W	PD Data Output Write Mask	0x0000_0000
PE_DATMSK	GPIO_BA+0x10C	R/W	PE Data Output Write Mask	0x0000_0000
PF_DATMSK	GPIO_BA+0x14C	R/W	PF Data Output Write Mask	0x0000_0000
PG_DATMSK	GPIO_BA+0x18C	R/W	PG Data Output Write Mask	0x0000_0000
PH_DATMSK	GPIO_BA+0x1CC	R/W	PH Data Output Write Mask	0x0000_0000
PI_DATMSK	GPIO_BA+0x20C	R/W	PI Data Output Write Mask	0x0000_0000
PJ_DATMSK	GPIO_BA+0x24C	R/W	PJ Data Output Write Mask	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DATMSK							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATMSK							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..15	**DATMSK[n]**	**Port A-J Pin[n] Data Output Write Mask** These bits are used to protect the corresponding DOUT (Px_DOUT[n]) bit. When the DATMSK (Px_DATMSK[n]) bit is set to 1, the corresponding DOUT (Px_DOUT[n]) bit is protected. If the write signal is masked, writing data to the protect bit is ignored. 0 = Corresponding DOUT (Px_DOUT[n]) bit can be updated. 1 = Corresponding DOUT (Px_DOUT[n]) bit protected. **Note 1:** This function only protects the corresponding DOUT (Px_DOUT[n]) bit, and will not protect the corresponding PDIO (Pxn_PDIO[0]) bit. **Note 2:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J Pin Value (Px_PIN)

Register	Offset	R/W	Description	Reset Value
PA_PIN	GPIO_BA+0x010	R	PA Pin Value	0x0000_XXXX
PB_PIN	GPIO_BA+0x050	R	PB Pin Value	0x0000_XXXX
PC_PIN	GPIO_BA+0x090	R	PC Pin Value	0x0000_XXXX
PD_PIN	GPIO_BA+0x0D0	R	PD Pin Value	0x0000_XXXX
PE_PIN	GPIO_BA+0x110	R	PE Pin Value	0x0000_XXXX
PF_PIN	GPIO_BA+0x150	R	PF Pin Value	0x0000_XXXX
PG_PIN	GPIO_BA+0x190	R	PG Pin Value	0x0000_XXXX
PH_PIN	GPIO_BA+0x1D0	R	PH Pin Value	0x0000_XXXX
PI_PIN	GPIO_BA+0x210	R	PI Pin Value	0x0000_XXXX
PJ_PIN	GPIO_BA+0x250	R	PJ Pin Value	0x0000_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PIN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PIN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**PIN[n]**	**Port A-J Pin[n] Pin Value** Each bit of the register reflects the actual status of the respective Px.n pin. If the bit is 1, it indicates the corresponding pin status is high; else the pin status is low. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J De-bounce Enable Control Register (Px_DBEN)

Register	Offset	R/W	Description	Reset Value
PA_DBEN	GPIO_BA+0x014	R/W	PA De-bounce Enable Control Register	0x0000_0000
PB_DBEN	GPIO_BA+0x054	R/W	PB De-bounce Enable Control Register	0x0000_0000
PC_DBEN	GPIO_BA+0x094	R/W	PC De-bounce Enable Control Register	0x0000_0000
PD_DBEN	GPIO_BA+0x0D4	R/W	PD De-bounce Enable Control Register	0x0000_0000
PE_DBEN	GPIO_BA+0x114	R/W	PE De-bounce Enable Control Register	0x0000_0000
PF_DBEN	GPIO_BA+0x154	R/W	PF De-bounce Enable Control Register	0x0000_0000
PG_DBEN	GPIO_BA+0x194	R/W	PG De-bounce Enable Control Register	0x0000_0000
PH_DBEN	GPIO_BA+0x1D4	R/W	PH De-bounce Enable Control Register	0x0000_0000
PI_DBEN	GPIO_BA+0x214	R/W	PI De-bounce Enable Control Register	0x0000_0000
PJ_DBEN	GPIO_BA+0x254	R/W	PJ De-bounce Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DBEN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DBEN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**DBEN[n]**	**Port A-J Pin[n] Input Signal De-bounce Enable Bit** The DBEN[n] bit is used to enable the de-bounce function for each corresponding bit. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is seen as the signal bounce and will not trigger the interrupt. The de-bounce clock source is controlled by DBCLKSRC (Px_DBCTL [4]), one de-bounce sample cycle period is controlled by DBCLKSEL (Px_DBCTL [3:0]). 0 = Px.n de-bounce function Disabled. 1 = Px.n de-bounce function Enabled. The de-bounce function is valid only for edge triggered interrupt. If the interrupt mode is level triggered, the de-bounce enable bit is ignored. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J Interrupt Type Control (Px_INTTYPE)

Register	Offset	R/W	Description	Reset Value
PA_INTTYPE	GPIO_BA+0x018	R/W	PA Interrupt Trigger Type Control	0x0000_0000
PB_INTTYPE	GPIO_BA+0x058	R/W	PB Interrupt Trigger Type Control	0x0000_0000
PC_INTTYPE	GPIO_BA+0x098	R/W	PC Interrupt Trigger Type Control	0x0000_0000
PD_INTTYPE	GPIO_BA+0x0D8	R/W	PD Interrupt Trigger Type Control	0x0000_0000
PE_INTTYPE	GPIO_BA+0x118	R/W	PE Interrupt Trigger Type Control	0x0000_0000
PF_INTTYPE	GPIO_BA+0x158	R/W	PF Interrupt Trigger Type Control	0x0000_0000
PG_INTTYPE	GPIO_BA+0x198	R/W	PG Interrupt Trigger Type Control	0x0000_0000
PH_INTTYPE	GPIO_BA+0x1D8	R/W	PH Interrupt Trigger Type Control	0x0000_0000
PI_INTTYPE	GPIO_BA+0x218	R/W	PI Interrupt Trigger Type Control	0x0000_0000
PJ_INTTYPE	GPIO_BA+0x258	R/W	PJ Interrupt Trigger Type Control	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TYPE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TYPE							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**TYPE[n]**	**Port A-J Pin[n] Edge or Level Detection Interrupt Trigger Type Control** TYPE (Px_INTTYPE[n]) bit is used to control the triggered interrupt is by level trigger or by edge trigger. If the interrupt is by edge trigger, the trigger source can be controlled by de-bounce. If the interrupt is by level trigger, the input source is sampled by one HCLK clock and generates the interrupt. 0 = Edge trigger interrupt. 1 = Level trigger interrupt. If the pin is set as the level trigger interrupt, only one level can be set on the registers RHIEN (Px_INTEN[n+16])/FLIEN (Px_INTEN[n]). If both levels to trigger interrupt are set, the setting is ignored and no interrupt will occur. The de-bounce function is valid only for edge triggered interrupt. If the interrupt mode is level triggered, the de-bounce enable bit is ignored. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J Interrupt Enable Control Register (Px_INTEN)

Register	Offset	R/W	Description	Reset Value
PA_INTEN	GPIO_BA+0x01C	R/W	PA Interrupt Enable Control Register	0x0000_0000
PB_INTEN	GPIO_BA+0x05C	R/W	PB Interrupt Enable Control Register	0x0000_0000
PC_INTEN	GPIO_BA+0x09C	R/W	PC Interrupt Enable Control Register	0x0000_0000
PD_INTEN	GPIO_BA+0x0DC	R/W	PD Interrupt Enable Control Register	0x0000_0000
PE_INTEN	GPIO_BA+0x11C	R/W	PE Interrupt Enable Control Register	0x0000_0000
PF_INTEN	GPIO_BA+0x15C	R/W	PF Interrupt Enable Control Register	0x0000_0000
PG_INTEN	GPIO_BA+0x19C	R/W	PG Interrupt Enable Control Register	0x0000_0000
PH_INTEN	GPIO_BA+0x1DC	R/W	PH Interrupt Enable Control Register	0x0000_0000
PI_INTEN	GPIO_BA+0x21C	R/W	PI Interrupt Enable Control Register	0x0000_0000
PJ_INTEN	GPIO_BA+0x25C	R/W	PJ Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
RHIEN							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RHIEN							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FLIEN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FLIEN							

Bits	Description	
[n+16] n=0,1..15	**RHIEN[n]**	**Port A-J Pin[n] Rising Edge or High Level Interrupt Trigger Type Enable Bit** The RHIEN (Px_INTEN[n+16]) bit is used to enable the interrupt for each of the corresponding input Px.n pin. Set bit to 1 also enable the pin wake-up function. When setting the RHIEN (Px_INTEN[n+16]) bit to 1 : If the interrupt is level trigger (TYPE (Px_INTTYPE[n]) bit is set to 1), the input Px.n pin will generate the interrupt while this pin state is at high level. If the interrupt is edge trigger (TYPE (Px_INTTYPE[n]) bit is set to 0), the input Px.n pin will generate the interrupt while this pin state changed from low to high. 0 = Px.n level high or low to high interrupt Disabled. 1 = Px.n level high or low to high interrupt Enabled. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.
[n] n=0,1..15	**FLIEN[n]**	**Port A-J Pin[n] Falling Edge or Low Level Interrupt Trigger Type Enable Bit** The FLIEN (Px_INTEN[n]) bit is used to enable the interrupt for each of the corresponding input Px.n pin. Set bit to 1 also enable the pin wake-up function.

		When setting the FLIEN (Px_INTEN[n]) bit to 1: If the interrupt is level trigger (TYPE (Px_INTTYPE[n]) bit is set to 1), the input Px.n pin will generate the interrupt while this pin state is at low level. If the interrupt is edge trigger (TYPE (Px_INTTYPE[n]) bit is set to 0), the input Px.n pin will generate the interrupt while this pin state changed from high to low. 0 = Px.n level low or high to low interrupt Disabled. 1 = Px.n level low or high to low interrupt Enabled. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J Interrupt Source Flag (Px_INTSRC)

Register	Offset	R/W	Description	Reset Value
PA_INTSRC	GPIO_BA+0x020	R/W	PA Interrupt Source Flag	0x0000_XXXX
PB_INTSRC	GPIO_BA+0x060	R/W	PB Interrupt Source Flag	0x0000_XXXX
PC_INTSRC	GPIO_BA+0x0A0	R/W	PC Interrupt Source Flag	0x0000_XXXX
PD_INTSRC	GPIO_BA+0x0E0	R/W	PD Interrupt Source Flag	0x0000_XXXX
PE_INTSRC	GPIO_BA+0x120	R/W	PE Interrupt Source Flag	0x0000_XXXX
PF_INTSRC	GPIO_BA+0x160	R/W	PF Interrupt Source Flag	0x0000_XXXX
PG_INTSRC	GPIO_BA+0x1A0	R/W	PG Interrupt Source Flag	0x0000_XXXX
PH_INTSRC	GPIO_BA+0x1E0	R/W	PH Interrupt Source Flag	0x0000_XXXX
PI_INTSRC	GPIO_BA+0x220	R/W	PI Interrupt Source Flag	0x0000_XXXX
PJ_INTSRC	GPIO_BA+0x260	R/W	PJ Interrupt Source Flag	0x0000_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
INTSRC							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTSRC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**INTSRC[n]**	**Port A-J Pin[n] Interrupt Source Flag** Write Operation: 0 = No action. 1 = Clear the corresponding pending interrupt. Read Operation: 0 = No interrupt at Px.n. 1 = Px.n generates an interrupt. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J Input Schmitt Trigger Enable Register (Px_SMTEN)

Register	Offset	R/W	Description	Reset Value
PA_SMTEN	GPIO_BA+0x024	R/W	PA Input Schmitt Trigger Enable Register	0x0000_0000
PB_SMTEN	GPIO_BA+0x064	R/W	PB Input Schmitt Trigger Enable Register	0x0000_0000
PC_SMTEN	GPIO_BA+0x0A4	R/W	PC Input Schmitt Trigger Enable Register	0x0000_0000
PD_SMTEN	GPIO_BA+0x0E4	R/W	PD Input Schmitt Trigger Enable Register	0x0000_0000
PE_SMTEN	GPIO_BA+0x124	R/W	PE Input Schmitt Trigger Enable Register	0x0000_0000
PF_SMTEN	GPIO_BA+0x164	R/W	PF Input Schmitt Trigger Enable Register	0x0000_0000
PG_SMTEN	GPIO_BA+0x1A4	R/W	PG Input Schmitt Trigger Enable Register	0x0000_0000
PH_SMTEN	GPIO_BA+0x1E4	R/W	PH Input Schmitt Trigger Enable Register	0x0000_0000
PI_SMTEN	GPIO_BA+0x224	R/W	PI Input Schmitt Trigger Enable Register	0x0000_0000
PJ_SMTEN	GPIO_BA+0x264	R/W	PJ Input Schmitt Trigger Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SMTEN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SMTEN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**SMTEN[n]**	**Port A-J Pin[n] Input Schmitt Trigger Enable Bit** 0 = Px.n input schmitt trigger function Disabled. 1 = Px.n input schmitt trigger function Enabled. **Note:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J High Slew Rate Control Register (Px_SLEWCTL)

Register	Offset	R/W	Description	Reset Value
PA_SLEWCTL	GPIO_BA+0x028	R/W	PA High Slew Rate Control Register	0x0000_0000
PB_SLEWCTL	GPIO_BA+0x068	R/W	PB High Slew Rate Control Register	0x0000_0000
PC_SLEWCTL	GPIO_BA+0x0A8	R/W	PC High Slew Rate Control Register	0x0000_0000
PD_SLEWCTL	GPIO_BA+0x0E8	R/W	PD High Slew Rate Control Register	0x0000_0000
PE_SLEWCTL	GPIO_BA+0x128	R/W	PE High Slew Rate Control Register	0x0000_0000
PF_SLEWCTL	GPIO_BA+0x168	R/W	PF High Slew Rate Control Register	0x0000_0000
PG_SLEWCTL	GPIO_BA+0x1A8	R/W	PG High Slew Rate Control Register	0x0000_0000
PH_SLEWCTL	GPIO_BA+0x1E8	R/W	PH High Slew Rate Control Register	0x0000_0000
PI_SLEWCTL	GPIO_BA+0x228	R/W	PI High Slew Rate Control Register	0x0000_0000
PJ_SLEWCTL	GPIO_BA+0x268	R/W	PJ High Slew Rate Control Register	0x0000_0000

31	30	29	28	27	26	25	24
HSREN15		HSREN14		HSREN13		HSREN12	
23	22	21	20	19	18	17	16
HSREN11		HSREN10		HSREN9		HSREN8	
15	14	13	12	11	10	9	8
HSREN07		HSREN6		HSREN5		HSREN4	
7	6	5	4	3	2	1	0
HSREN03		HSREN2		HSREN1		HSREN0	

Bits	Description	
[2n+1:2n] n=0,1..15	**HSRENn**	**Port A-J Pin[n] High Slew Rate Control** 00 = Px.n output with normal slew rate mode. 01 = Px.n output with high slew rate mode. 10 = Px.n output with fast slew rate mode). 11 = Reserved. **Note 1:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter. **Note 2:** Please refer to the Datasheet for detailed pin operation voltage information about V_{DD}, V_{DDIO} and V_{BAT} electrical characteristics.

Port A-J Pull-up and Pull-down Selection Register (Px_PUSEL)

Register	Offset	R/W	Description	Reset Value
PA_PUSEL	GPIO_BA+0x030	R/W	PA Pull-up and Pull-down Selection Register	0x0000_0000
PB_PUSEL	GPIO_BA+0x070	R/W	PB Pull-up and Pull-down Selection Register	0x0000_0000
PC_PUSEL	GPIO_BA+0x0B0	R/W	PC Pull-up and Pull-down Selection Register	0x0000_0000
PD_PUSEL	GPIO_BA+0x0F0	R/W	PD Pull-up and Pull-down Selection Register	0x0000_0000
PE_PUSEL	GPIO_BA+0x130	R/W	PE Pull-up and Pull-down Selection Register	0x0000_0000
PF_PUSEL	GPIO_BA+0x170	R/W	PF Pull-up and Pull-down Selection Register	0x0000_0000
PG_PUSEL	GPIO_BA+0x1B0	R/W	PG Pull-up and Pull-down Selection Register	0x0000_0000
PH_PUSEL	GPIO_BA+0x1F0	R/W	PH Pull-up and Pull-down Selection Register	0x0000_0000
PI_PUSEL	GPIO_BA+0x230	R/W	PI Pull-up and Pull-down Selection Register	0x0000_0000
PJ_PUSEL	GPIO_BA+0x270	R/W	PJ Pull-up and Pull-down Selection Register	0x0000_0000

31	30	29	28	27	26	25	24
PUSEL15		PUSEL14		PUSEL13		PUSEL12	
23	22	21	20	19	18	17	16
PUSEL11		PUSEL10		PUSEL9		PUSEL8	
15	14	13	12	11	10	9	8
PUSEL7		PUSEL6		PUSEL5		PUSEL4	
7	6	5	4	3	2	1	0
PUSEL3		PUSEL2		PUSEL1		PUSEL0	

Bits	Description	
[2n+1:2n] n=0,1..15	**PUSELn**	**Port A-J Pin[n] Pull-up and Pull-down Enable Register** Determine each I/O Pull-up/pull-down of Px.n pins. 00 = Px.n pull-up and pull-down disable. 01 = Px.n pull-up enable. 10 = Px.n pull-down enable. 11 = Reserved. **Note 1:** Basically, the pull-up control and pull-down control has following behavior limitation: The independent pull-up control register only valid when MODEn set as tri-state and open-drain mode. The independent pull-down control register only valid when MODEn set as tri-state mode. When both pull-up pull-down is set as 1 at “tri-state” mode, keep I/O in tri-state mode. **Note 2:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

Port A-J Interrupt De-bounce Control Register (Px_DBCTL)

Register	Offset	R/W	Description	Reset Value
PA_DBCTL	GPIO_BA+0x034	R/W	PA Interrupt De-bounce Control Register	0x0000_0020
PB_DBCTL	GPIO_BA+0x074	R/W	PB Interrupt De-bounce Control Register	0x0000_0020
PC_DBCTL	GPIO_BA+0x0B4	R/W	PC Interrupt De-bounce Control Register	0x0000_0020
PD_DBCTL	GPIO_BA+0x0F4	R/W	PD Interrupt De-bounce Control Register	0x0000_0020
PE_DBCTL	GPIO_BA+0x134	R/W	PE Interrupt De-bounce Control Register	0x0000_0020
PF_DBCTL	GPIO_BA+0x174	R/W	PF Interrupt De-bounce Control Register	0x0000_0020
PG_DBCTL	GPIO_BA+0x1B4	R/W	PG Interrupt De-bounce Control Register	0x0000_0020
PH_DBCTL	GPIO_BA+0x1F4	R/W	PH Interrupt De-bounce Control Register	0x0000_0020
PI_DBCTL	GPIO_BA+0x234	R/W	PI Interrupt De-bounce Control Register	0x0000_0020
PJ_DBCTL	GPIO_BA+0x274	R/W	PJ Interrupt De-bounce Control Register	0x0000_0020

31	30	29	28	27	26	25	24
DBCLKBUSY	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**ICLKON**	**DBCLKSRC**	**DBCLKSEL**			

Bits	Description	
[31]	**DBCLKBUSY**	**De-bounce Clock Switching Busy Flag (Read Only)** This bit is set when de-bounce clock source is changed by setting DBCLKSRC(Px_DBCTL[4]). And it is cleared after de-bounce clock source switching is finished. De-bounce function can work normally after de-bounce clock switch done. 0 = De-bounce clock switch done. 1 = De-bounce clock is switching.
[30:6]	**Reserved**	Reserved.

Bits	Description	
[5]	**ICLKON**	**Interrupt Clock on Mode** 0 = Edge detection circuit is active only if I/O pin corresponding RHIEN (Px_INTEN[n+16])/FLIEN (Px_INTEN[n]) bit is set to 1. If corresponding RHIEN or FLIEN is 0, the clock of I/O detect circuit is stopped and interrupt source flag(Px_INTSRC) cannot be clear also. 1 = All I/O pins edge detection circuit is always active after reset. **Note 1:** It is recommended to disable this bit to save system power if no special application concern. **Note 2:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.
[4]	**DBCLKSRC**	**De-bounce Counter Clock Source Selection** 0 = De-bounce counter clock source is the HCLK. 1 = De-bounce counter clock source is the 10 kHz internal low speed RC oscillator (LIRC).
[3:0]	**DBCLKSEL**	**De-bounce Sampling Cycle Selection** 0000 = Sample interrupt input once per 1 clocks. 0001 = Sample interrupt input once per 2 clocks. 0010 = Sample interrupt input once per 4 clocks. 0011 = Sample interrupt input once per 8 clocks. 0100 = Sample interrupt input once per 16 clocks. 0101 = Sample interrupt input once per 32 clocks. 0110 = Sample interrupt input once per 64 clocks. 0111 = Sample interrupt input once per 128 clocks. 1000 = Sample interrupt input once per 256 clocks. 1001 = Sample interrupt input once per 2*256 clocks. 1010 = Sample interrupt input once per 4*256 clocks. 1011 = Sample interrupt input once per 8*256 clocks. 1100 = Sample interrupt input once per 16*256 clocks. 1101 = Sample interrupt input once per 32*256 clocks. 1110 = Sample interrupt input once per 64*256 clocks. 1111 = Sample interrupt input once per 128*256 clocks.

INT0~7 Input Noise Filter Register (INTx_INNF)

Register	Offset	R/W	Description	Reset Value
INT0_INNF	GPIO_BA+0x450	R/W	INT0 Input Noise Filter Register	0x0000_0000
INT1_INNF	GPIO_BA+0x454	R/W	INT1 Input Noise Filter Register	0x0000_0000
INT2_INNF	GPIO_BA+0x458	R/W	INT2 Input Noise Filter Register	0x0000_0000
INT3_INNF	GPIO_BA+0x45C	R/W	INT3 Input Noise Filter Register	0x0000_0000
INT4_INNF	GPIO_BA+0x460	R/W	INT4 Input Noise Filter Register	0x0000_0000
INT5_INNF	GPIO_BA+0x464	R/W	INT5 Input Noise Filter Register	0x0000_0000
INT6_INNF	GPIO_BA+0x468	R/W	INT6 Input Noise Filter Register	0x0000_0000
INT7_INNF	GPIO_BA+0x46C	R/W	INT7 Input Noise Filter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**NFCNT**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**NFSEL**			**Reserved**			**NFEN**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:8]	**NFCNT**	**Noise Filter Count** The register bits control the filter counter to count from 0 to NFCNT.
[7]	**Reserved**	Reserved.
[6:4]	**NFSEL**	**Noise Filter Clock Selection** 000 = Filter clock is HCLK. 001 = Filter clock is HCLK/2. 010 = Filter clock is HCLK/4. 011 = Filter clock is HCLK/8. 100 = Filter clock is HCLK/16. 101 = Filter clock is HCLK/32. 110 = Filter clock is HCLK/64. 111 = Filter clock is HCLK/128.
[3:1]	**Reserved**	Reserved.
[0]	**NFEN**	**Noise Filter Enable** 0 = Noise Filter function Disabled.

		1 = Noise Filter function Enabled.

INT Edge Detect Control Register (INT_EDETCTL)

Register	Offset	R/W	Description	Reset Value
INT_EDETCTL	GPIO_BA+0x490	R/W	INT Edge Detect Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
EDETCTL7		**EDETCTL6**		**EDETCTL5**		**EDETCTL4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EDETCTL3		**EDETCTL2**		**EDETCTL1**		**EDETCTL0**	

Bits	Description	
[31:16]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..7	**EDETCTLn**	**INTn Edge Detect Control Bits** 00 = Not detect. 01 = INTn low to high detection Enable. 10 = INTn high to low detection Enable. 11 = INTn both low to high and high to low detection Enable.

INT Egde Detect Interrupt Enable Control Register (INT_EDINTEN)

Register	Offset	R/W	Description	Reset Value
INT_EDINTEN	GPIO_BA+0x498	R/W	INT Edge Detect Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
EDIEN7	EDIEN6	EDIEN5	EDIEN4	EDIEN3	EDIEN2	EDIEN1	EDIEN0

Bits	Description	
[31:8]	**Reserved**	Reseved.
[n] n=0,1..7	**EDIENn**	**INTn Edge Detect Interrupt Enable Bit** 0 = INTn Edge Detect Interrupt Disable. 1 = INTn Edge Detect Interrupt Enable.

INT Edge Detect Interrupt Flag Register (INT_EDSTS)

Register	Offset	R/W	Description	Reset Value
INT_EDSTS	GPIO_BA+0x49C	R/W	INT Edge Detect Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EDIF7	**EDIF6**	**EDIF5**	**EDIF4**	**EDIF3**	**EDIF2**	**EDIF1**	**EDIF0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..7	**EDIFn**	**INTn Edge Detect Interrupt Flag** 0 = No Edge Detection happened. 1 = Rising Edge or Falling edge has been detected. **Note:** This bit is cleared by writing 1 to it.

GPIO Px.n Pin Data Input/Output Register (Pxn_PDIO)

Register	Offset	R/W	Description	Reset Value
PAn_PDIO n=0,1..15	GPIO_BA+0x800+(0x04 * n)	R/W	GPIO PA.n Pin Data Input/Output Register	0x0000_000X
PBn_PDIO n=0,1..15	GPIO_BA+0x840+(0x04 * n)	R/W	GPIO PB.n Pin Data Input/Output Register	0x0000_000X
PCn_PDIO n=0,1..14	GPIO_BA+0x880+(0x04 * n)	R/W	GPIO PC.n Pin Data Input/Output Register	0x0000_000X
PDn_PDIO n=0,1..14	GPIO_BA+0x8C0+(0x04 * n)	R/W	GPIO PD.n Pin Data Input/Output Register	0x0000_000X
PEn_PDIO n=0,1..15	GPIO_BA+0x900+(0x04 * n)	R/W	GPIO PE.n Pin Data Input/Output Register	0x0000_000X
PFn_PDIO n=0,1..11	GPIO_BA+0x940+(0x04 * n)	R/W	GPIO PF.n Pin Data Input/Output Register	0x0000_000X
PGn_PDIO n=0,1..15	GPIO_BA+0x980+(0x04 * n)	R/W	GPIO PG.n Pin Data Input/Output Register	0x0000_000X
PHn_PDIO n=0,1..15	GPIO_BA+0x9C0+(0x04 * n)	R/W	GPIO PH.n Pin Data Input/Output Register	0x0000_000X
PIn_PDIO n=6,7..15	GPIO_BA+0xA00+(0x04 * n)	R/W	GPIO PI.n Pin Data Input/Output Register	0x0000_000X
PJn_PDIO n=0,1..13	GPIO_BA+0xA40+(0x04 * n)	R/W	GPIO PJ.n Pin Data Input/Output Register	0x0000_000X

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**PDIO**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**PDIO**	**GPIO Px.n Pin Data Input/Output** Writing this bit can control one GPIO pin output value. 0 = Corresponding GPIO pin set to low. 1 = Corresponding GPIO pin set to high. Read this register to get GPIO pin status.

		For example, writing PA0_PDIO will reflect the written value to bit DOUT (Px_DOUT[0]), reading PA0_PDIO will return the value of PIN (PA_PIN[0]). **Note 1:** The writing operation will not be affected by register DATMSK (Px_DATMSK[n]). **Note 2:** For more information about Px.n, please refer to the "PIN CONFIGURATION" chapter.

6.9 PDMA Controller (PDMA)

6.9.1 Overview

The peripheral direct memory access (PDMA) controller is used to provide high-speed data transfer. The PDMA controller can transfer data from one address to another without CPU intervention. This has the benefit of reducing the workload of CPU and keeps CPU resources free for other applications. There are two PDMA controller PDMA0 and PDMA1. PDMA controller has a total of 16 channels and each channel can perform transfer between memory and peripherals or between memory and memory.

6.9.2 Features

- Supports 16 independently configurable channels
- Supports selectable 2 level of priority (fixed priority or round-robin priority)
- Supports 2 PDMA controller PDMA0 and PDMA1
- Supports transfer data width of 8, 16, and 32 bits
- Supports source and destination address increment size can be byte, half-word, word or no increment
- Supports software and USB, UART, USCI, SPI, QSPI, ACMP, PSIO, EINT, EPWM, I^2C0, I^2C1, I^2S, Timer, ADC, and DAC request
- Supports Scatter-gather mode to perform sophisticated transfer through the use of the descriptor link list table
- Supports single and burst transfer type
- Supports time-out function on all channels
- Supports stride function from channel 0 to channel 5
- Supports enhanced stride function on channel 0 and channel1

6.9.3 Block Diagram

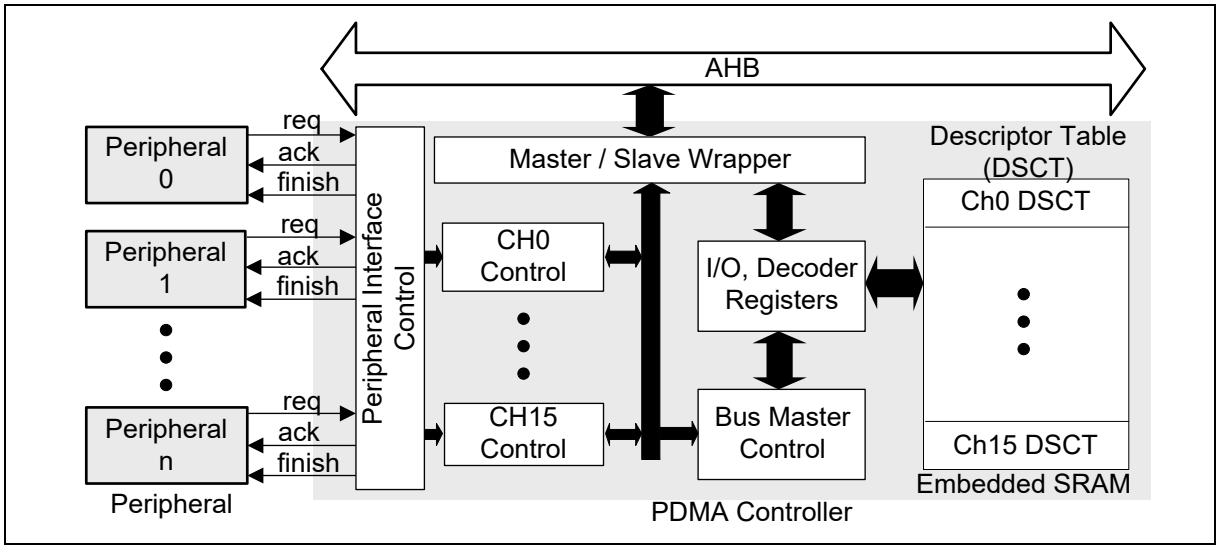


Figure 6.9-1 PDMA Controller Block Diagram

6.9.4 Basic Configuration

6.9.4.1 Basic Configuration of PDMA0

- Clock Source Configuration
  - Enable PDMA0 controller clock in PDMA0CKEN(CLK_AHBCLK [1]).
- Reset Configuration
  - Reset PDMA0 controller in PDMA0RST (SYS_IPRST0[2]).

6.9.4.2 Basic Configuration of PDMA1

- Clock Source Configuration
  - Enable PDMA1 controller clock in PDMA1CKEN (CLK_AHBCLK [18]).
- Reset Configuration
  - Reset PDMA1 controller in PDMA1RST (SYS_IPRST0[18]).

6.9.5 Functional Description

The PDMA controller transfers data from one address to another without CPU intervention. The PDMA controller supports 16 independent channels and serves only one channel at one time, as the result, PDMA controller supports two level channel priorities: fixed and round-robin priority, PDMA controller serves channel in order from highest to lowest priority channel. The PDMA controller supports two operation modes: Basic mode and Scatter-gather mode. Basic mode is used to perform one descriptor table transfer. Scatter-gather mode has more entries for each PDMA channel, and thus the PDMA controller supports sophisticated transfer through the entries. The descriptor table entry data structure contains many transfer information including the transfer source address, transfer destination address, transfer count, burst size, transfer type and operation mode. Figure 6.9-2 shows the diagram of descriptor table (DSCT) data structure.

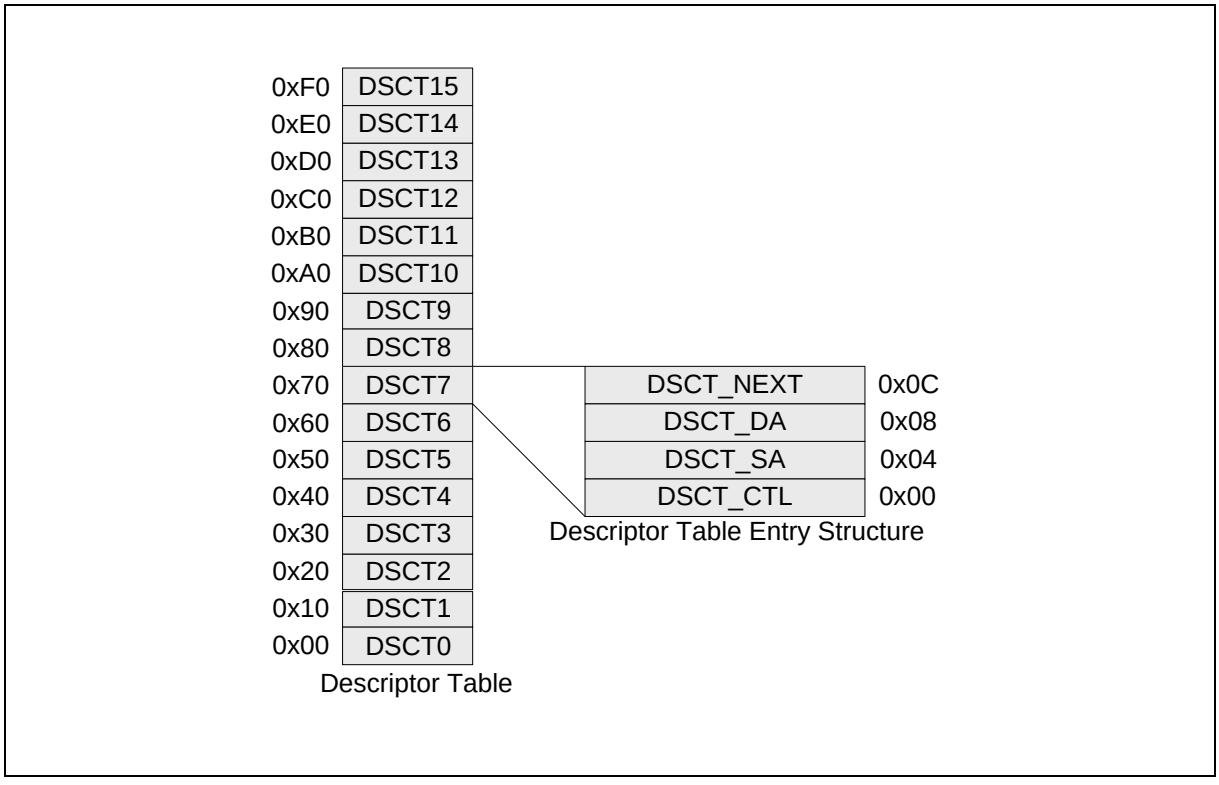


Figure 6.9-2 Descriptor Table Entry Structure

The PDMA controller also supports single and burst transfer type and the request source can be from

software or peripheral request, transfer between memory to memory using software request. A single transfer means that software or peripheral is ready to transfer one data (every data needs one request), and the burst transfer means that software or peripherals will transfer multiple data (multiple data only need one request).

6.9.5.1 Channel Priority

The PDMA controller supports two level channel priorities including fixed and round-robin priority. The fixed priority channel has higher priority than round-robin priority channel. If multiple channels are set as fixed or round-robin priority, the higher channel will have higher priority. The priority order is listed in Table 6.9-1.

PDMA_PRISET	Channel Number	Priority Setting	Arbitration Priority In Descending Order
1	15	Channel15, Fixed Priority	Highest
1	14	Channel14, Fixed Priority	---
---	---	---	---
1	0	Channel0, Fixed Priority	---
0	15	Channel15, Round-Robin Priority	---
0	14	Channel14, Round-Robin Priority	---
---	---	---	---
0	0	Channel0, Round-Robin Priority	Lowest

Table 6.9-1 Channel Priority Table

6.9.5.2 PDMA Operation Mode

The PDMA controller supports two operation modes including Basic mode and Scatter-gather mode.

Basic Mode

Basic mode is used to perform one descriptor table transfer mode. This mode can be used to transfer data between memory and memory, peripherals and memory or peripherals and peripherals, but if user want to transfer data between peripherals and peripherals, one thing must be sured is that the request from peripherals knows that the data is ready for transfer or not. PDMA controller operation mode can be set from OPMODE (PDMA_DSCTn_CTL[1:0], n denotes PDMA channel), the default setting is in idle state (OPMODE (PDMA_DSCTn_CTL[1:0]) = 0x0) and recommend user configure the descriptor table in idle state. If operation mode is not in idle state, user re-configure channel setting may make some operation error.

User must fill the transfer count TXCNT (PDMA_DSCTn_CTL[31:16]) register and select transfer width TXWIDTH (PDMA_DSCTn_CTL[13:12]), destination address increment size DAINC (PDMA_DSCTn_CTL[11:10]), source address increment size SAINC (PDMA_DSCTn_CTL[9:8]), burst size BURSIZE (PDMA_DSCTn_CTL[6:4]) and transfer type TXTYPE (PDMA_DSCTn_CTL[2]), then the PDMA controller will perform transfer operation in transfer state after receiving request signal. Finishing this task will generate an interrupt to CPU if corresponding PDMA interrupt bit INTENn (PDMA_INTEN[15:0]) is enabled and the operation mode will be updated to idle state as shown in Figure 6.9-3. If software configures the operation mode to idle state, the PDMA controller will not perform any transfer and then clear this operation request. Finishing this task will also generate an interrupt to CPU if corresponding PDMA interrupt bit is enabled.

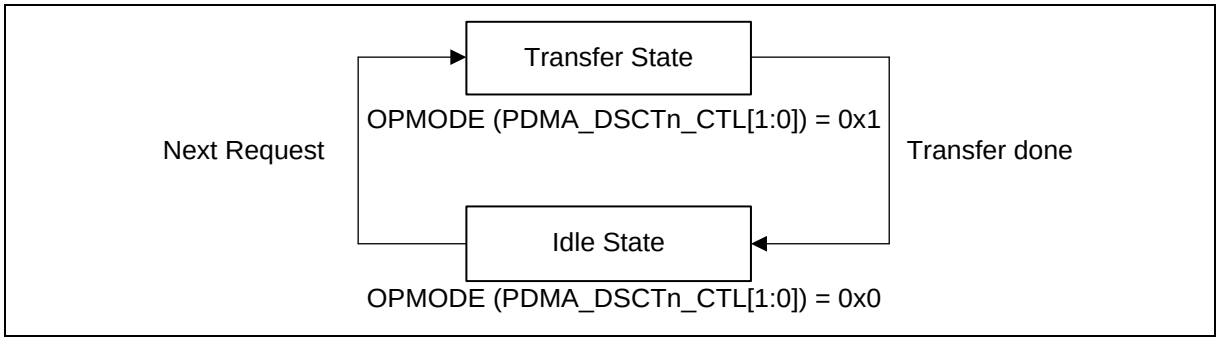


Figure 6.9-3 Basic Mode Finite State Machine

Scatter-gather Mode

Scatter-gather mode is a complex mode and can perform sophisticated transfer through the use of the description link list table as shown in Figure 6.9-4. Through operation mode user can perform peripheral wrapper-around, and multiple PDMA task can be used for data transfer between varied locations in system memory instead of a set of contiguous locations. Scatter-gather mode only needs a request to finish all table entries task till the last task with OPMODE (PDMA_DSCTn_CTL[1:0]) is idle state without ack. It also means Scatter-gather mode can only be used to transfer data between memory to memory without handshaking.

In Scatter-gather mode, the table is just used for jumping to the next table entry. The first task will not perform any operation transfer. Finishing each task will generate an interrupt to CPU if corresponding PDMA interrupt bit is enabled and TBINTDIS (PDMA_DSCTn_CTL[7]) bit is “0” (when finishing task and TBINTDIS bit is “0”, corresponding TDIFn (PDMA_TDSTS[15:0]) flag will be asserted and if this bit is “1” TDIFn will not be active).

If channel 7 has been triggered, and the operation mode is in Scatter-gather mode (OPMODE (PDMA_DSCTn_CTL[1:0]) = 0x2), the hardware will load the real PDMA information task from the address generated by adding PDMA_DSCTn_NEXT (link address) and PDMA_SCATBA (base address) registers. For example, base address is 0x2000_0000 (only MSB 16 bits valid in PDMA_SCATBA), the current link address is 0x0000_0100 (only LSB 16-bits without last two bits [1:0] valid in PDMA_DSCTn_NEXT), and then the next DSCT entry start address is 0x2000_0100.

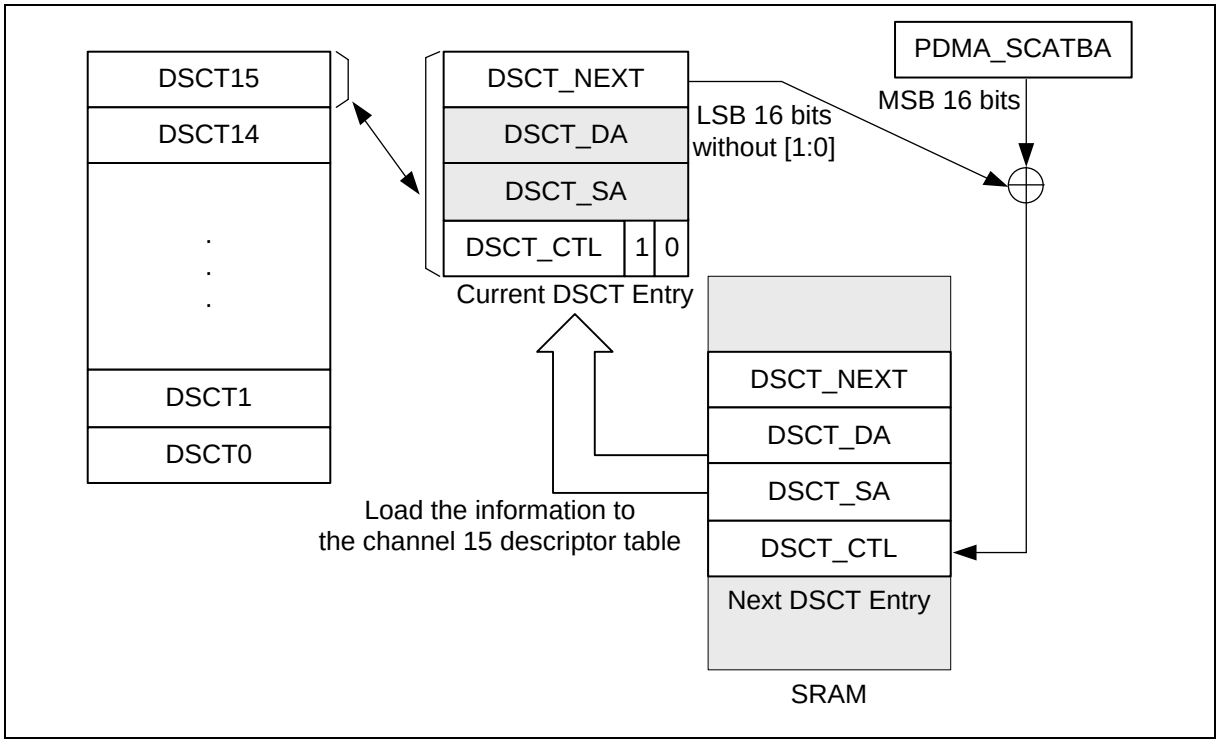


Figure 6.9-4 Descriptor Table Link List Structure

The above link list table operation is DSCT state in Scatter-gather Mode as shown in Figure 6.9-5. When loading the information is finished, it will go to transfer state and start transfer by this information automatically. However, if the next PDMA information is also set to Scatter-gather mode, the hardware will catch the next PDMA information block when the current task is finished. The Scatter-gather mode switches to basic mode when doing the next task. Then, the basic mode switches to Idle state when the last task is finished.

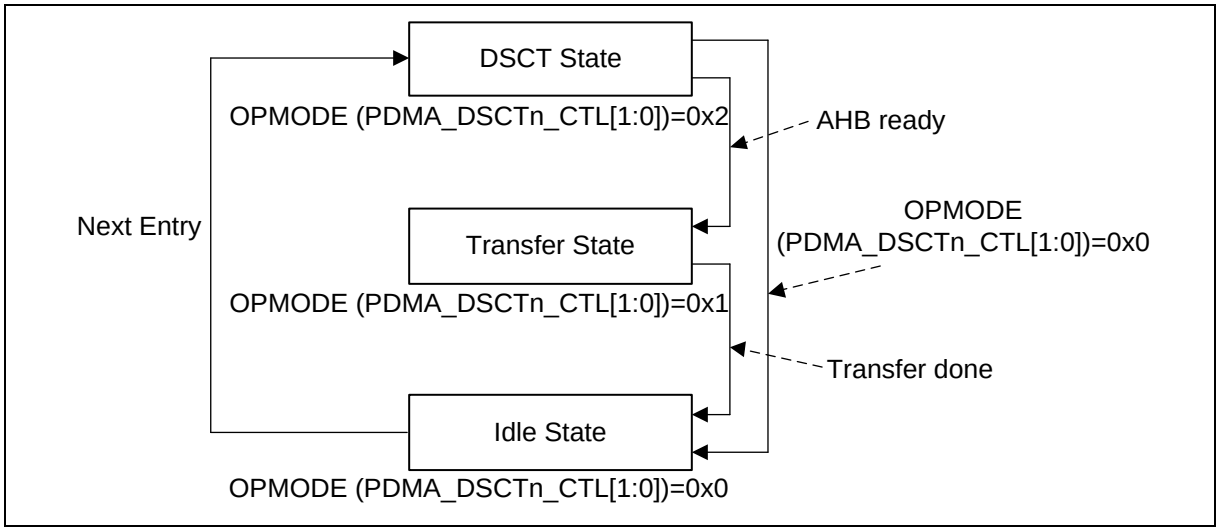


Figure 6.9-5 Scatter-gather Mode Finite State Machine

6.9.5.3 Transfer Type

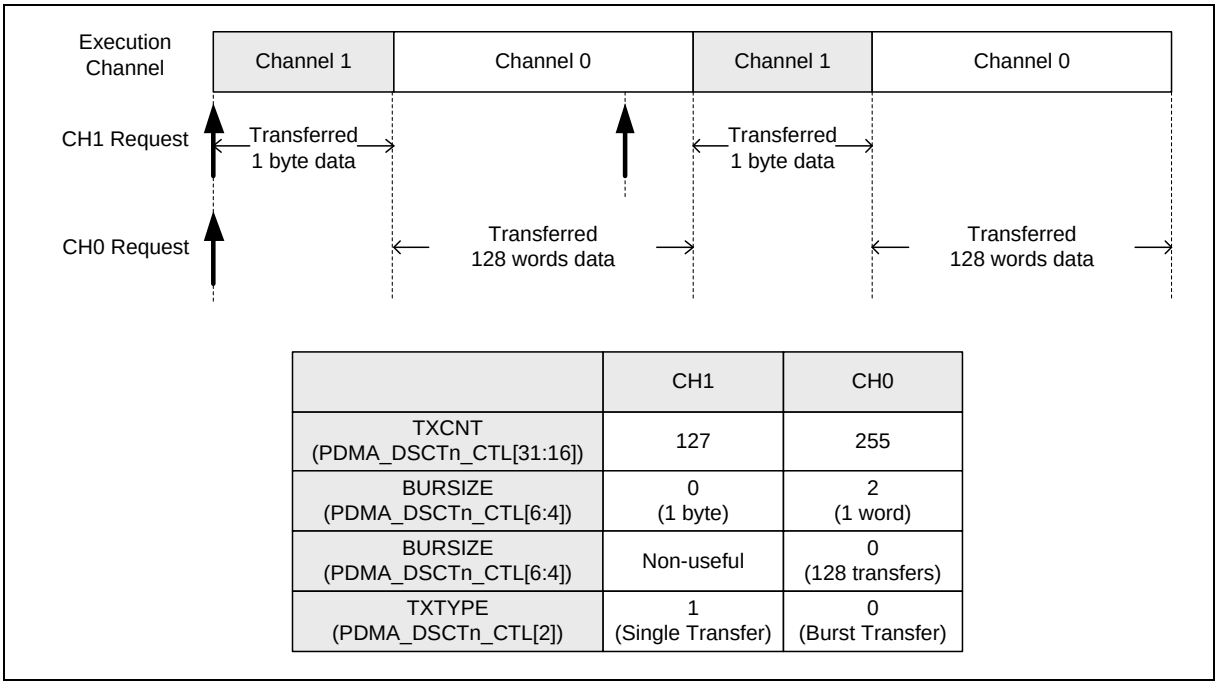
The PDMA controller supports two transfer types: single transfer type and burst transfer type, configure by setting TXTYPE (PDMA_DSCTn_CTL[2]).

When the PDMA controller is operated in single transfer type, each transfer data needs one request signal for one transfer, after transferred data, TXCNT (PDMA_DSCTn_CTL[31:16]) will decrease 1. Transfer will be finished after the TXCNT (PDMA_DSCTn_CTL[31:16]) decreases to 0. In this mode, the BURSIZE (PDMA_DSCTn_CTL[6:4]) is not useful to control the transfer size. The BURSIZE (PDMA_DSCTn_CTL[6:4]) will be fixed as one.

For the burst transfer type, the PDMA controller transfers TXCNT (PDMA_DSCTn_CTL[31:16]) of data and need only one request signal. After transferred BURSIZE (PDMA_DSCTn_CTL[6:4]) of data, TXCNT (PDMA_DSCTn_CTL[31:16]) will decrease BURSIZE number. Transfer will be done after the transfer count TXCNT (PDMA_DSCTn_CTL[31:16]) decreases to 0. Note that burst transfer type can only be used for PDMA controller to do burst transfer between memory and memory. User must use single request type for memory-to-peripheral and peripheral-to-memory transfers. Please note that, PDMA transfer data between Flash and memory should finish before MCU enter idle mode or power done mode to prevent access wrong data.

Figure 6.9-6 shows an example about single and burst transfer type in basic mode. In this example, channel 1 uses single transfer type and TXCNT (PDMA_DSCTn_CTL[31:16]) = 127. Channel 0 uses burst transfer type, BURSIZE (PDMA_DSCTn_CTL[6:4]) = 128 and TXCNT (PDMA_DSCTn_CTL[31:16]) = 255. The operation sequence is described below:

1. Channel 0 and channel 1 get the trigger signal at the same time.
2. Channel 1 has higher priority than channel 0 by default; the PDMA controller will load the channel 1 descriptor table first and executing. But channel 1 is single transfer type, and thus the PDMA controller will only transfer one transfer data.
3. Then, the PDMA controller turns to the channel 0 and loads channel 0's descriptor table. The channel 0 is burst transfer type and the burst size selected to 128. Therefore, the PDMA controller will transfer 128 transfer data.
4. When channel 0 transfers 128 data, channel 1 gets another request signal, then after channel 0 finishes 128 transfer data, the PDMA controller will turn to channel 1 and transfer next one data.
5. After channel 1 transfers data, the PDMA controller switches to low priority channel 0 to continuous next 128 data transfer. If no channel 1 request receives, PDMA will start next channel 0, 128 data transfer.
6. The PDMA controller will complete transfer when channel 0 finishes data transfer 256 times, and channel 1 finishes transferring 128 times.



	CH1	CH0
TXCNT (PDMA_DSCTn_CTL[31:16])	127	255
BURSIZE (PDMA_DSCTn_CTL[6:4])	0 (1 byte)	2 (1 word)
BURSIZE (PDMA_DSCTn_CTL[6:4])	Non-useful	0 (128 transfers)
TXTYPE (PDMA_DSCTn_CTL[2])	1 (Single Transfer)	0 (Burst Transfer)

Figure 6.9-6 Example of Single Transfer Type and Burst Transfer Type in Basic Mode

6.9.5.4 Channel Time-out

PDMA all channels support time-out function. When the transfer channel is enabled and selected to the peripheral, corresponding channel time-out TOUTENn (PDMA_TOUTEN [n]) is enabled, then channel's corresponding time-out counter will start count up from 0 while the channel has received trigger signal from the peripheral.

The time-out counter is based on output of HCLK prescaler, which is setting by corresponding channel's TOUTPSCn . If time-out counter counts up from 0 to corresponding channel's TOCn, the PDMA controller will generate interrupt signal when corresponding TOUTIENn (PDMA_TOUTIEN [n]) is enabled. When time-out occurred, corresponding channel's REQTOFn (PDMA_INTSTS [n+8], n=0~15) will be set to indicate channel time-out is happened.

Time-out counter will restart from 0 while counter count to TOCn, received trigger signal, time-out function is disabled or chip enters Power-down mode. The time-out counter will keep counting until time-out function is disabled.

Figure 6.9-7 shows an example about time-out counter operation. The operation sequence is described below:

1. The channel 0 time-out counter is not counting when time-out function is enabled by setting TOUTEN0(PDMA_TOUTEN[0]) bit to 1.
2. Time-out counter starts counting from 0 to the value of TOC0(PDMA_TOC0_1[15:0]) bits when receiving the first peripheral request.
3. Time-out counter is reset to 0 by received second peripheral request.
4. Channel 0 request time-out flag(REQTOF0(PDMA_INTSTS[8])) is set to high when time-out counter counts to 5. The counter will keep counting.
5. Time-out counter is reset to 0 when time-out function is disabled.

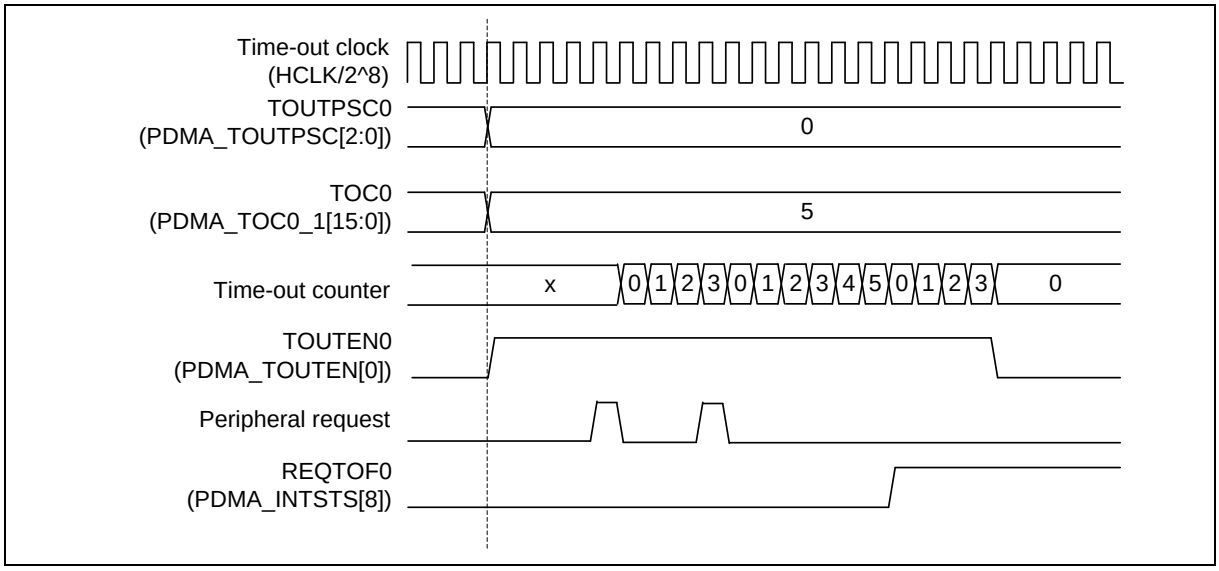


Figure 6.9-7 Example of PDMA Channel 0 Time-out Counter Operation

6.9.5.5 Stride Function

The PDMA supports six channels channel 0 to channel 5 with stride function. The stride function can transfer data from one address to another address and support block transfer with stride. When operating with stride function, the transfer address can be fixed or incremented successively.

Set STRIDEEN (PDMA_DSCTn_CTL[15]) to enable the stride function, and then write a valid source address to the PDMA_DSCTn_SA register and a source address offset count to SASOL (PDMA_ASOCRn[15:0]) register, a destination address to the PDMA_DSCTn_DA register and a destination address offset count to DASOL (PDMA_ASOCRn[31:16]), and a transfer count to the TXCNT (PDMA_DSCTn_CTL[31:16]) and a stride transfer count to STC (PDMA_STCRn[15:0]). Next, trigger the SWREQn (PDMA_SWREQ[5:0]). The PDMA will start and then stop the transfer after TXCNT (PDMA_DSCTn_CTL[31:16]) counts down to 0. Figure 6.9-8 shows the block transfer relationship between source memory and destination memory. The stride function also supports peripheral to memory or memory to peripheral transfer.

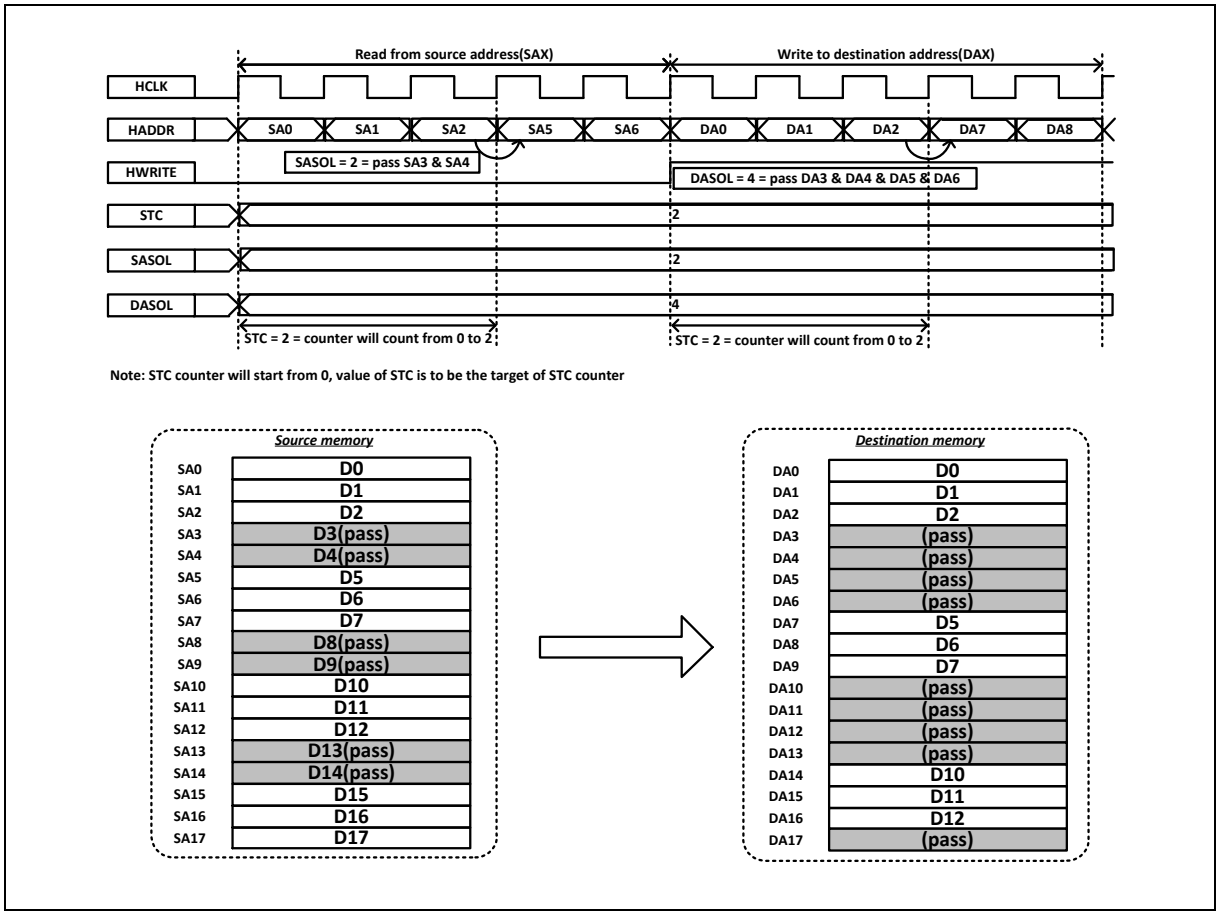


Figure 6.9-8 Stride Function Block Transfer

6.9.5.6 Enhanced Stride Function

To improve the stride function for DSP, image processing and other applications, channel 0 and channel 1 support interval count and repeat count. Interval count is used to decide the interval between each block transfer. SAICNT (PDMA_AICTLn[15:0]) is for source address interval count, and DAICNT (PDMA_AICTLn[31:16]) is for destination address interval count. Repeat count is used to decide repeat times of block transfer with only one trigger. If block transfer repeat RCNT (PDMA_RCNTn[15:0]) times, transfer is stopped. The following shows two cases of block transfer. Case one with SAICNT = 3, DAICNT = 9 and repeat count = 3. Case two with SAICNT = 3, DAICNT = 6 and repeat count = 3

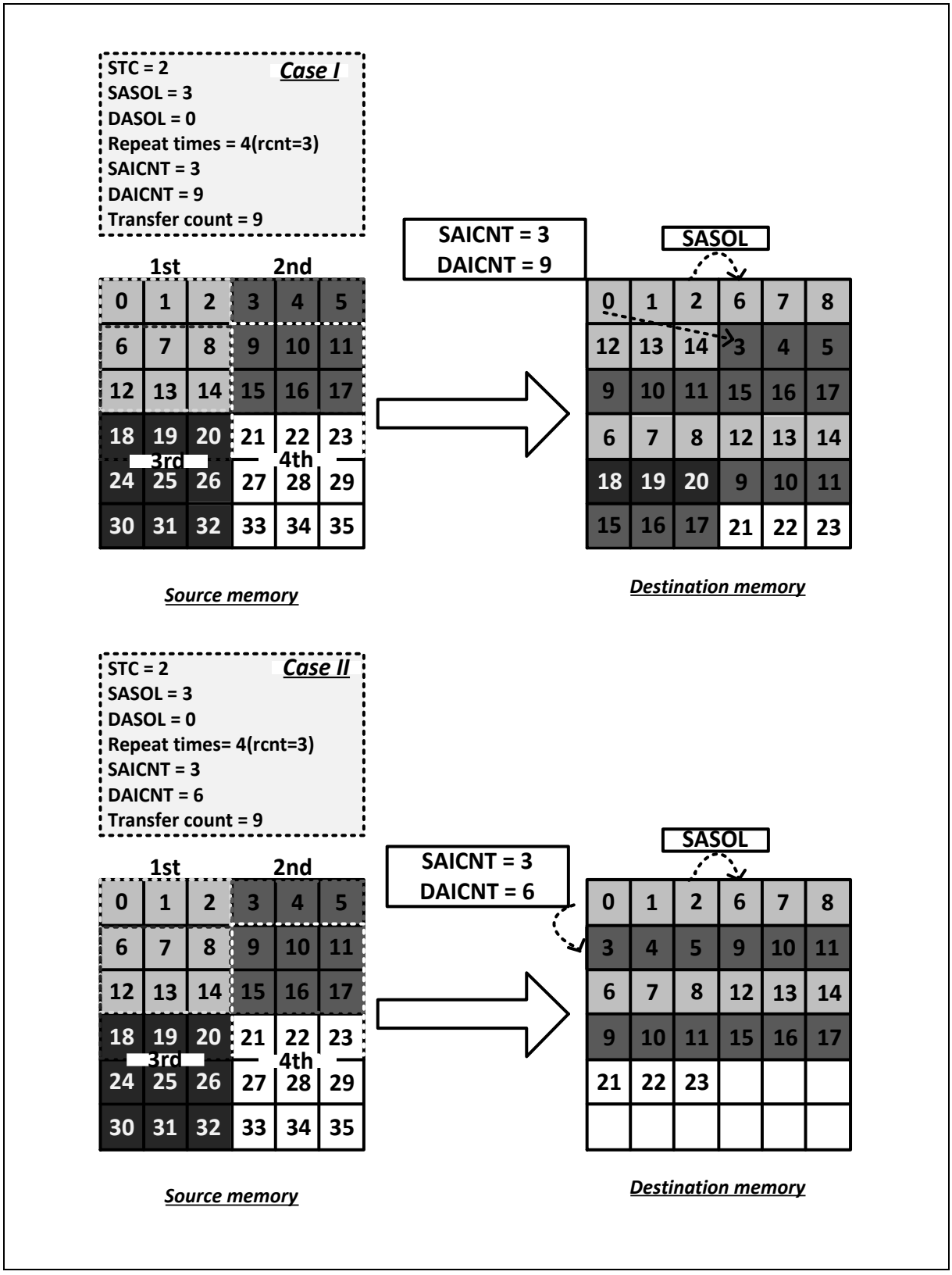


Figure 6.9-9 Block Transfer with Interval =3 and Repeat Count =2

6.9.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
PDMA Base Address: **PDMAx_BA = 0x4000_8000+(0x1_0000*x)** **x=0,1**				
PDMAx_DSCTn_CTL **n = 0,1..15**	PDMAx_BA+0x10*n	R/W	Descriptor Table Control Register of PDMA Channel n	0xXXXX_XXXX
PDMAx_DSCTn_SA **n = 0,1..15**	PDMAx_BA+0x0004+0x10*n	R/W	Source Address Register of PDMA Channel n	0xXXXX_XXXX
PDMAx_DSCTn_DA **n = 0,1..15**	PDMAx_BA+0x0008+0x10*n	R/W	Destination Address Register of PDMA Channel n	0xXXXX_XXXX
PDMAx_DSCTn_NEXT **n = 0,1..15**	PDMAx_BA+0x000c+0x10*n	R/W	Next Scatter-gather Descriptor Table Offset Address of PDMA Channel n	0xXXXX_XXXX
PDMAx_CURSCATn **n = 0,1..15**	PDMAx_BA+0x0080+0x004*n	R	Current Scatter-gather Descriptor Table Address of PDMA Channel n	0xXXXX_XXXX
PDMAx_CHCTL	PDMAx_BA+0x400	R/W	PDMA Channel Control Register	0x0000_0000
PDMAx_PAUSE	PDMAx_BA+0x404	W	PDMA Transfer Pause Control Register	0x0000_0000
PDMAx_SWREQ	PDMAx_BA+0x408	W	PDMA Software Request Register	0x0000_0000
PDMAx_TRGSTS	PDMAx_BA+0x40C	R	PDMA Channel Request Status Register	0x0000_0000
PDMAx_PRISET	PDMAx_BA+0x410	R/W	PDMA Fixed Priority Setting Register	0x0000_0000
PDMAx_PRICLR	PDMAx_BA+0x414	W	PDMA Fixed Priority Clear Register	0x0000_0000
PDMAx_INTEN	PDMAx_BA+0x418	R/W	PDMA Interrupt Enable Register	0x0000_0000
PDMAx_INTSTS	PDMAx_BA+0x41C	R/W	PDMA Interrupt Status Register	0x0000_0000
PDMAx_ABTSTS	PDMAx_BA+0x420	R/W	PDMA Channel Read/Write Target Abort Flag Register	0x0000_0000
PDMAx_TDSTS	PDMAx_BA+0x424	R/W	PDMA Channel Transfer Done Flag Register	0x0000_0000
PDMAx_ALIGN	PDMAx_BA+0x428	R/W	PDMA Transfer Alignment Status Register	0x0000_0000
PDMAx_TACTSTS	PDMAx_BA+0x42C	R	PDMA Transfer Active Flag Register	0x0000_0000
PDMAx_TOUTPSC0_7	PDMAx_BA+0x430	R/W	PDMA Time-out Prescaler Register(CH0 to CH7)	0x0000_0000
PDMAx_TOUTEN	PDMAx_BA+0x434	R/W	PDMA Time-out Enable Register	0x0000_0000
PDMAx_TOUTIEN	PDMAx_BA+0x438	R/W	PDMA Time-out Interrupt Enable Register	0x0000_0000
PDMAx_SCATBA	PDMAx_BA+0x43C	R/W	PDMA Scatter-gather Descriptor Table Base Address Register	0x2000_0000
PDMAx_TOC0_1	PDMAx_BA+0x440	R/W	PDMA Time-out Counter Ch0 and Ch1 Register	0x0000_0000
PDMAx_TOC2_3	PDMAx_BA+0x444	R/W	PDMA Time-out Counter Ch2 and Ch3 Register	0x0000_0000
PDMAx_TOC4_5	PDMAx_BA+0x448	R/W	PDMA Time-out Counter Ch4 and Ch5 Register	0x0000_0000
PDMAx_TOC6_7	PDMAx_BA+0x44C	R/W	PDMA Time-out Counter Ch6 and Ch7 Register	0x0000_0000

PDMAx_TOC8_9	PDMAx_BA+0x450	R/W	PDMA Time-out Counter Ch8 and Ch9 Register	0x0000_0000
PDMAx_TOC10_11	PDMAx_BA+0x454	R/W	PDMA Time-out Counter Ch10 and Ch11 Register	0x0000_0000
PDMAx_TOC12_13	PDMAx_BA+0x458	R/W	PDMA Time-out Counter Ch12 and Ch13 Register	0x0000_0000
PDMAx_TOC14_15	PDMAx_BA+0x45C	R/W	PDMA Time-out Counter Ch14 and Ch15 Register	0x0000_0000
PDMAx_CHRST	PDMAx_BA+0x460	R/W	PDMA Channel Reset Register	0x0000_0000
PDMAx_TOUTPSC8_15	PDMAx_BA+0x468	R/W	PDMA Time-out Prescaler Register(CH8 to CH15)	0x0000_0000
PDMAx_REQSEL0_3	PDMAx_BA+0x480	R/W	PDMA Request Source Select Register 0	0x0000_0000
PDMAx_REQSEL4_7	PDMAx_BA+0x484	R/W	PDMA Request Source Select Register 1	0x0000_0000
PDMAx_REQSEL8_11	PDMAx_BA+0x488	R/W	PDMA Request Source Select Register 2	0x0000_0000
PDMAx_REQSEL12_15	PDMAx_BA+0x48C	R/W	PDMA Request Source Select Register 3	0x0000_0000
PDMAx_STCR0	PDMAx_BA+0x500	R/W	Stride Transfer Count Register of PDMA Channel 0	0x0000_0000
PDMAx_ASOCR0	PDMAx_BA+0x504	R/W	Address Stride Offset Register of PDMA Channel 0	0x0000_0000
PDMAx_STCR1	PDMAx_BA+0x508	R/W	Stride Transfer Count Register of PDMA Channel 1	0x0000_0000
PDMAx_ASOCR1	PDMAx_BA+0x50C	R/W	Address Stride Offset Register of PDMA Channel 1	0x0000_0000
PDMAx_STCR2	PDMAx_BA+0x510	R/W	Stride Transfer Count Register of PDMA Channel 2	0x0000_0000
PDMAx_ASOCR2	PDMAx_BA+0x514	R/W	Address Stride Offset Register of PDMA Channel 2	0x0000_0000
PDMAx_STCR3	PDMAx_BA+0x518	R/W	Stride Transfer Count Register of PDMA Channel 3	0x0000_0000
PDMAx_ASOCR3	PDMAx_BA+0x51C	R/W	Address Stride Offset Register of PDMA Channel 3	0x0000_0000
PDMAx_STCR4	PDMAx_BA+0x520	R/W	Stride Transfer Count Register of PDMA Channel 4	0x0000_0000
PDMAx_ASOCR4	PDMAx_BA+0x524	R/W	Address Stride Offset Register of PDMA Channel 4	0x0000_0000
PDMAx_STCR5	PDMAx_BA+0x528	R/W	Stride Transfer Count Register of PDMA Channel 5	0x0000_0000
PDMAx_ASOCR5	PDMAx_BA+0x52C	R/W	Address Stride Offset Register of PDMA Channel 5	0x0000_0000
PDMAx_AICTL0	PDMAx_BA + 0x600	R/W	Address Interval Control Register of PDMA Channel 0	0x0000_0000
PDMAx_RCNT0	PDMAx_BA + 0x604	R/W	Repeat Count Register of PDMA Channel 0	0x0000_0000
PDMAx_AICTL1	PDMAx_BA + 0x608	R/W	Address Interval Control Register of PDMA Channel 1	0x0000_0000
PDMAx_RCNT1	PDMAx_BA + 0x60C	R/W	Repeat Count Register of PDMA Channel 1	0x0000_0000

6.9.7 Register Description

Descriptor Table Control Register (PDMAx_DSCTn_CTL)

Register	Offset	R/W	Description	Reset Value
PDMAx_DSCTn_CTL	PDMAx_BA+0x10*n	R/W	Descriptor Table Control Register of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
TXCNT							
23	22	21	20	19	18	17	16
TXCNT							
15	14	13	12	11	10	9	8
STRIDEEN	Reserved	TXWIDTH		DAINC		SAINC	
7	6	5	4	3	2	1	0
TBINTDIS	BURSIZE			Reserved	TXTYPE	OPMODE	

Bits	Description	
[31:16]	**TXCNT**	**Transfer Count** The TXCNT represents the required number of PDMA transfer, the real transfer count is (TXCNT + 1); The maximum transfer count is 65536, every transfer may be byte, half-word or word that is dependent on TXWIDTH field. **Note:** When PDMA finishes each transfer data, this field will be decreased immediately.
[15]	**STRIDEEN**	**Stride Mode Enable Bit** 0 = Stride transfer mode Disabled. 1 = Stride transfer mode Enabled.
[14]	**Reserved**	Reserved.
[13:12]	**TXWIDTH**	**Transfer Width Selection** This field is used for transfer width. 00 = One byte (8 bit) is transferred for every operation. 01= One half-word (16 bit) is transferred for every operation. 10 = One word (32-bit) is transferred for every operation. 11 = Reserved. **Note:** The PDMA transfer source address (PDMA_DSCTn_SA) and PDMA transfer destination address (PDMA_DSCTn_DA) should be alignment under the TXWIDTH selection
[11:10]	**DAINC**	**Destination Address Increment** This field is used to set the destination address increment size. 11 = No increment (fixed address). Others = Increment and size is depended on TXWIDTH selection. **Note:** The fixed address function does not support in memory to memory transfer type.
[9:8]	**SAINC**	**Source Address Increment** This field is used to set the source address increment size. 11 = No increment (fixed address). Others = Increment and size is depended on TXWIDTH selection.

Bits	Description	
		Note: The fixed address function does not support in memory to memory transfer type.
[7]	**TBINTDIS**	**Table Interrupt Disable Bit** This field can be used to decide whether to enable table interrupt or not. If the TBINTDIS bit is enabled it will not generates TDIFn(PDMA_TDSTS[7:0]) when PDMA controller finishes transfer task. 0 = Table interrupt Enabled. 1 = Table interrupt Disabled. **Note:** This function only for Scatter-gather mode.
[6:4]	**BURSIZE**	**Burst Size** This field is used for peripheral to determine the burst size or used for determine the re-arbitration size. 000 = 128 Transfers. 001 = 64 Transfers. 010 = 32 Transfers. 011 = 16 Transfers. 100 = 8 Transfers. 101 = 4 Transfers. 110 = 2 Transfers. 111 = 1 Transfers. **Note:** This field is only useful in burst transfer type.
[3]	**Reserved**	Reserved.
[2]	**TXTYPE**	**Transfer Type** 0 = Burst transfer type. 1 = Single transfer type.
[1:0]	**OPMODE**	**PDMA Operation Mode Selection** 00 = Idle state: Channel is stopped or this table is complete, when PDMA finish channel table task, OPMODE will be cleared to idle state automatically. 01 = Basic mode: The descriptor table only has one task. When this task is finished, the PDMA_INTSTS[1] will be asserted. 10 = Scatter-gather mode: When operating in this mode, user must give the next descriptor table address in PDMA_DSCTn_NEXT register; PDMA controller will ignore this task, then load the next task to execute. 11 = Reserved. **Note:** Before filling new transfer task in the Descriptor Table, user must check the PDMA_INTSTS[1] to make sure the current task is complete.

Start Source Address Register (PDMAx_DSCTn_SA)

Register	Offset	R/W	Description	Reset Value
PDMAx_DSCTn_SA	PDMAx_BA+0x0004+0x10*n	R/W	Source Address Register of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
SA							
23	22	21	20	19	18	17	16
SA							
15	14	13	12	11	10	9	8
SA							
7	6	5	4	3	2	1	0
SA							

Bits	Description	
[31:0]	SA	**PDMA Transfer Source Address** This field indicates a 32-bit source address of PDMA controller.

Destination Address Register (PDMAx_DSCTn_DA)

Register	Offset	R/W	Description	Reset Value
PDMAx_DSCTn_DA	PDMAx_BA+0x0008+0x10*n	R/W	Destination Address Register of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
DA							
23	22	21	20	19	18	17	16
DA							
15	14	13	12	11	10	9	8
DA							
7	6	5	4	3	2	1	0
DA							

Bits	Description	
[31:0]	DA	**PDMA Transfer Destination Address** This field indicates a 32-bit destination address of PDMA controller.

Next Scatter-gather Descriptor Table Offset Address (PDMAx_DSCTn_NEXT)

Register	Offset	R/W	Description	Reset Value
PDMAx_DSCTn_NEXT	PDMAx_BA+0x000c+0x10*n	R/W	Next Scatter-gather Descriptor Table Offset Address of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
EXENEXT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
EXENEXT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NEXT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NEXT							

Bits	Description	
[31:16]	**EXENEXT**	**PDMA Execution Next Descriptor Table Offset** This field indicates the offset of next descriptor table address of current execution descriptor table in system memory. **Note:** Write operation is useless in this field.
[15:0]	**NEXT**	**PDMA Next Descriptor Table Offset** This field indicates the offset of the next descriptor table address in system memory. **Write Operation:** If the system memory based address is 0x2000_0000 (PDMA_SCATBA), and the next descriptor table is start from 0x2000_0100, then this field must fill in 0x0100. **Read Operation:** When operating in Scatter-gather mode, the last two bits NEXT[1:0] will become reserved, and indicate the first next address of system memory. **Note 1:** The descriptor table address must be word boundary. **Note 2:** Before filling transfer task in the descriptor table, user must check if the descriptor table is complete.

Current Scatter-gather Descriptor Table Address (PDMAx_CURSCATn)

Register	Offset	R/W	Description	Reset Value
PDMAx_CURSCATn	PDMAx_BA+0x0080+0x004*n	R	Current Scatter-gather Descriptor Table Address of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
CURADDR							
23	22	21	20	19	18	17	16
CURADDR							
15	14	13	12	11	10	9	8
CURADDR							
7	6	5	4	3	2	1	0
CURADDR							

Bits	Description	
[31:0]	CURADDR	**PDMA Current Description Address (Read Only)** This field indicates a 32-bit current external description address of PDMA controller. **Note:** This field is read only and used for Scatter-gather mode only to indicate the current external description address.

Channel Control Register (PDMAx_CHCTL)

Register	Offset	R/W	Description	Reset Value
PDMAx_CHCTL	PDMAx_BA+0x400	R/W	PDMA Channel Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CHEN15	**CHEN14**	**CHEN13**	**CHEN12**	**CHEN11**	**CHEN10**	**CHEN9**	**CHEN8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CHEN7	**CHEN6**	**CHEN5**	**CHEN4**	**CHEN3**	**CHEN2**	**CHEN1**	**CHEN0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**CHENn**	**PDMA Channel Enable Bits** Set this bit to 1 to enable PDMAn operation. Channel cannot be active if it is not set as enabled. 0 = PDMA channel [n] Disabled. 1 = PDMA channel [n] Enabled. **Note:** Setting the corresponding bit of PDMA_PAUSE or PDMA_CHRST register will also clear this bit.

PDMA Transfer Pause Control Register (PDMAx_PAUSE)

Register	Offset	R/W	Description	Reset Value
PDMAx_PAUSE	PDMAx_BA+0x404	W	PDMA Transfer Pause Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PAUSE15	**PAUSE14**	**PAUSE13**	**PAUSE12**	**PAUSE11**	**PAUSE10**	**PAUSE9**	**PAUSE8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PAUSE7	**PAUSE6**	**PAUSE5**	**PAUSE4**	**PAUSE3**	**PAUSE2**	**PAUSE1**	**PAUSE0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**PAUSEn**	**PDMA Channel n Transfer Pause Control (Write Only)** User can set PAUSEn bit field to pause the PDMA transfer. When user sets PAUSEn bit, the PDMA controller will pause the on-going transfer, then clear the channel enable bit CHENn(PDMA_CHCTL [n], n=0,1.. 15) and clear request active flag(PDMA_TRGSTS[n:0], n=0,1.. 15). If the paused channel is re-enabled again, the remaining transfers will be processed. 0 = No effect. 1 = Pause PDMA channel n transfer.

PDMA Software Request Register (PDMAx_SWREQ)

Register	Offset	R/W	Description	Reset Value
PDMAx_SWREQ	PDMAx_BA+0x408	W	PDMA Software Request Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SWREQ15	**SWREQ14**	**SWREQ13**	**SWREQ12**	**SWREQ11**	**SWREQ10**	**SWREQ9**	**SWREQ8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWREQ7	**SWREQ6**	**SWREQ5**	**SWREQ4**	**SWREQ3**	**SWREQ2**	**SWREQ1**	**SWREQ0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**SWREQn**	**PDMA Software Request (Write Only)** Set this bit to 1 to generate a software request to PDMA [n]. 0 = No effect. 1 = Generate a software request. **Note 1:** User can read PDMA_TRGSTS register to know which channel is on active. Active flag may be triggered by software request or peripheral request. **Note 2:** If user does not enable corresponding PDMA channel, the software request will be ignored.

PDMA Channel Request Status Register (PDMAx_TRGSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_TRGSTS	PDMAx_BA+0x40C	R	PDMA Channel Request Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
REQSTS15	**REQSTS14**	**REQSTS13**	**REQSTS12**	**REQSTS11**	**REQSTS10**	**REQSTS9**	**REQSTS8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REQSTS7	**REQSTS6**	**REQSTS5**	**REQSTS4**	**REQSTS3**	**REQSTS2**	**REQSTS1**	**REQSTS0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**REQSTSn**	**PDMA Channel Request Status (Read Only)** This flag indicates whether channel[n] have a request or not, no matter request from software or peripheral. When PDMA controller finishes channel transfer, this bit will be cleared automatically. 0 = PDMA Channel n has no request. 1 = PDMA Channel n has a request. **Note:** If user pauses or resets each PDMA transfer by setting PDMA_PAUSE or PDMA_CHRST register respectively, this bit will be cleared automatically after finishing the current transfer.

PDMA Fixed Priority Setting Register (PDMAx_PRISET)

Register	Offset	R/W	Description	Reset Value
PDMAx_PRISET	PDMAx_BA+0x410	R/W	PDMA Fixed Priority Setting Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FPRISET15	**FPRISET14**	**FPRISET13**	**FPRISET12**	**FPRISET11**	**FPRISET10**	**FPRISET9**	**FPRISET8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FPRISET7	**FPRISET6**	**FPRISET5**	**FPRISET4**	**FPRISET3**	**FPRISET2**	**FPRISET1**	**FPRISET0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**FPRISETn**	**PDMA Fixed Priority Setting** Set this bit to 1 to enable fixed priority level. **Write Operation:** 0 = No effect. 1 = Set PDMA channel [n] to fixed priority channel. **Read Operation:** 0 = Corresponding PDMA channel is round-robin priority. 1 = Corresponding PDMA channel is fixed priority. **Note:** This field is only set to fixed priority. To clear fixed priority, use PDMA_PRICLR register.

PDMA Fix Priority Clear Register (PDMAx_PRICLR)

Register	Offset	R/W	Description	Reset Value
PDMAx_PRICLR	PDMAx_BA+0x414	W	PDMA Fixed Priority Clear Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FPRICLR15	**FPRICLR14**	**FPRICLR13**	**FPRICLR12**	**FPRICLR11**	**FPRICLR10**	**FPRICLR9**	**FPRICLR8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FPRICLR7	**FPRICLR6**	**FPRICLR5**	**FPRICLR4**	**FPRICLR3**	**FPRICLR2**	**FPRICLR1**	**FPRICLR0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**FPRICLRn**	**PDMA Fixed Priority Clear Bits (Write Only)** Set this bit to 1 to clear fixed priority level. 0 = No effect. 1 = Clear PDMA channel [n] fixed priority setting. **Note:** User can read PDMA_PRISET register to know the channel priority.

PDMA Interrupt Enable Register (PDMAx_INTEN)

Register	Offset	R/W	Description	Reset Value
PDMAx_INTEN	PDMAx_BA+0x418	R/W	PDMA Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
INTEN15	**INTEN14**	**INTEN13**	**INTEN12**	**INTEN11**	**INTEN10**	**INTEN9**	**INTEN8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTEN7	**INTEN6**	**INTEN5**	**INTEN4**	**INTEN3**	**INTEN2**	**INTEN1**	**INTEN0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**INTENn**	**PDMA Interrupt Enable Bits** This field is used to enable PDMA channel[n] interrupt. 0 = PDMA channel n interrupt Disabled. 1 = PDMA channel n interrupt Enabled. **Note:** The interrupt flag is time-out, abort, transfer done and align.

NPDMA Interrupt Status Register (PDMAx_INTSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_INTSTS	PDMAx_BA+0x41C	R/W	PDMA Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
REQTOF15	**REQTOF14**	**REQTOF13**	**REQTOF12**	**REQTOF11**	**REQTOF10**	**REQTOF9**	**REQTOF8**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
REQTOF7	**REQTOF6**	**REQTOF5**	**REQTOF4**	**REQTOF3**	**REQTOF2**	**REQTOF1**	**REQTOF0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**ALIGNF**	**TDIF**	**ABTIF**

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**REQTOF15**	**Request Time-out Flag for Channel 15** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC15(PDMA_TOC15_14[31:16]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[22]	**REQTOF14**	**Request Time-out Flag for Channel 14** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC14(PDMA_TOC15_14[15:0]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[21]	**REQTOF13**	**Request Time-out Flag for Channel 13** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC13(PDMA_TOC13_12[31:16]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[20]	**REQTOF12**	**Request Time-out Flag for Channel 12** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC12(PDMA_TOC13_12[15:0]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit.

Bits	Description	
		Note 2: This bit will be cleared by writing 1 to it.
[19]	**REQTOF11**	**Request Time-out Flag for Channel 11** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC11(PDMA_TOC10_11[31:16]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[18]	**REQTOF10**	**Request Time-out Flag for Channel 10** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC10(PDMA_TOC10_11[15:0]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[17]	**REQTOF9**	**Request Time-out Flag for Channel 9** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC9(PDMA_TOC8_9[31:16]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[16]	**REQTOF8**	**Request Time-out Flag for Channel 8** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC8(PDMA_TOC8_9[15:0]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[15]	**REQTOF7**	**Request Time-out Flag for Channel 7** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC7(PDMA_TOC6_7[31:16]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[14]	**REQTOF6**	**Request Time-out Flag for Channel 6** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC6(PDMA_TOC6_7[15:0]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[13]	**REQTOF5**	**Request Time-out Flag for Channel 5** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC5(PDMA_TOC4_5[31:16]). 0 = No request time-out.

Bits	Description	
		1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[12]	**REQTOF4**	**Request Time-out Flag for Channel 4** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC4(PDMA_TOC4_5[15:0]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[11]	**REQTOF3**	**Request Time-out Flag for Channel 3** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC3(PDMA_TOC2_3[31:16]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[10]	**REQTOF2**	**Request Time-out Flag for Channel 2** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC2(PDMA_TOC2_3[15:0]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[9]	**REQTOF1**	**Request Time-out Flag for Channel 1** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC1(PDMA_TOC0_1[31:16]). 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[8]	**REQTOF0**	**Request Time-out Flag for Channel 0** This flag indicates that PDMA controller has waited peripheral request for a period defined by TOC0(PDMA_TOC0_1[15:0]. 0 = No request time-out. 1 = Peripheral request time-out. **Note 1:** Please disable time-out function before clearing this bit. **Note 2:** This bit will be cleared by writing 1 to it.
[7:3]	**Reserved**	Reserved.
[2]	**ALIGNF**	**Transfer Alignment Interrupt Flag (Read Only)** 0 = PDMA channel source address and destination address both follow transfer width setting. 1 = PDMA channel source address or destination address is not follow transfer width setting.
[1]	**TDIF**	**Transfer Done Interrupt Flag (Read Only)** This bit indicates that PDMA controller has finished transmission; User can read PDMA_TDSTS register to indicate which channel finished transfer. 0 = Not finished yet.

Bits	Description	
		1 = PDMA channel has finished transmission.
[0]	**ABTIF**	**PDMA Read/Write Target Abort Interrupt Flag (Read Only)** This bit indicates that PDMA has target abort error; Software can read PDMA_ABTSTS register to find which channel has target abort error. 0 = No AHB bus ERROR response received. 1 = AHB bus ERROR response received.

PDMA Channel Read/Write Target Abort Flag Register (PDMAx_ABTSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_ABTSTS	PDMAx_BA+0x420	R/W	PDMA Channel Read/Write Target Abort Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ABTIF15	**ABTIF14**	**ABTIF13**	**ABTIF12**	**ABTIF11**	**ABTIF10**	**ABTIF9**	**ABTIF8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ABTIF7	**ABTIF6**	**ABTIF5**	**ABTIF4**	**ABTIF3**	**ABTIF2**	**ABTIF1**	**ABTIF0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**ABTIFn**	**PDMA Read/Write Target Abort Interrupt Status Flag** This bit indicates which PDMA controller has target abort error. 0 = No AHB bus ERROR response received when channel n transfer. 1 = AHB bus ERROR response received when channel n transfer. **Note 1:** If channel n target abort, REQSRCn should set 0 to disable peripheral request. **Note 2** This bit will be cleared by writing 1 to it.

PDMA Channel Transfer Done Flag Register (PDMAx_TDSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_TDSTS	PDMAx_BA+0x424	R/W	PDMA Channel Transfer Done Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TDIF15	**TDIF14**	**TDIF13**	**TDIF12**	**TDIF11**	**TDIF10**	**TDIF9**	**TDIF8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TDIF7	**TDIF6**	**TDIF5**	**TDIF4**	**TDIF3**	**TDIF2**	**TDIF1**	**TDIF0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**TDIFn**	**Transfer Done Flag** This bit indicates whether PDMA controller channel transfer has been finished or not. 0 = PDMA channel transfer has not finished. 1 = PDMA channel has finished transmission. **Note:** User can write 1 to clear these bits.

PDMA Transfer Alignment Status Register (PDMAx_ALIGN)

Register	Offset	R/W	Description	Reset Value
PDMAx_ALIGN	PDMAx_BA+0x428	R/W	PDMA Transfer Alignment Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ALIGN15	**ALIGN14**	**ALIGN13**	**ALIGN12**	**ALIGN11**	**ALIGN10**	**ALIGN9**	**ALIGN8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ALIGN7	**ALIGN6**	**ALIGN5**	**ALIGN4**	**ALIGN3**	**ALIGN2**	**ALIGN1**	**ALIGN0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**ALIGNn**	**Transfer Alignment Flag** This bit indicates whether source and destination address both follow transfer width setting. 0 = PDMA channel source address and destination address both follow transfer width setting. 1 = PDMA channel source address or destination address is not follow transfer width setting. **Note:** User can write 1 to clear these bits.

PDMA Transfer Active Flag Register (PDMAx_TACTSTS)

Register	Offset	R/W	Description	Reset Value
PDMAx_TACTSTS	PDMAx_BA+0x42C	R	PDMA Transfer Active Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXACTF15	**TXACTF14**	**TXACTF13**	**TXACTF12**	**TXACTF11**	**TXACTF10**	**TXACTF9**	**TXACTF8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXACTF7	**TXACTF6**	**TXACTF5**	**TXACTF4**	**TXACTF3**	**TXACTF2**	**TXACTF1**	**TXACTF0**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**TXACTFn**	**Transfer on Active Flag (Read Only)** This bit indicates which PDMA channel is in active. 0 = PDMA channel is finished. 1 = PDMA channel is active.

PDMA Time-out Prescaler Register (PDMAx_TOUTPSC0_7)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOUTPSC0_7	PDMAx_BA+0x430	R/W	PDMA Time-out Prescaler Register(CH0 to CH7)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	TOUTPSC7			Reserved	TOUTPSC6		
23	22	21	20	19	18	17	16
Reserved	TOUTPSC5			Reserved	TOUTPSC4		
15	14	13	12	11	10	9	8
Reserved	TOUTPSC3			Reserved	TOUTPSC2		
7	6	5	4	3	2	1	0
Reserved	TOUTPSC1			Reserved	TOUTPSC0		

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TOUTPSC7**	**PDMA Channel 7 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 7 time-out clock source is HCLK/2^8. 001 = PDMA channel 7 time-out clock source is HCLK/2^9. 010 = PDMA channel 7 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 7 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 7 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 7 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 7 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 7 time-out clock source is HCLK/2^{15}.
[27]	**Reserved**	Reserved.
[26:24]	**TOUTPSC6**	**PDMA Channel 6 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 6 time-out clock source is HCLK/2^8. 001 = PDMA channel 6 time-out clock source is HCLK/2^9. 010 = PDMA channel 6 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 6 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 6 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 6 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 6 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 6 time-out clock source is HCLK/2^{15}.
[23]	**Reserved**	Reserved.
[22:20]	**TOUTPSC5**	**PDMA Channel 5 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 5 time-out clock source is HCLK/2^8. 001 = PDMA channel 5 time-out clock source is HCLK/2^9. 010 = PDMA channel 5 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 5 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 5 time-out clock source is HCLK/2^{12}.

Bits	Description	
		101 = PDMA channel 5 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 5 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 5 time-out clock source is HCLK/2^{15}.
[19]	**Reserved**	Reserved.
[18:16]	**TOUTPSC4**	**PDMA Channel 4 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 4 time-out clock source is HCLK/2^{8}. 001 = PDMA channel 4 time-out clock source is HCLK/2^{9}. 010 = PDMA channel 4 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 4 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 4 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 4 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 4 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 4 time-out clock source is HCLK/2^{15}.
[15]	**Reserved**	Reserved.
[14:12]	**TOUTPSC3**	**PDMA Channel 3 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 3 time-out clock source is HCLK/2^{8}. 001 = PDMA channel 3 time-out clock source is HCLK/2^{9}. 010 = PDMA channel 3 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 3 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 3 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 3 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 3 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 3 time-out clock source is HCLK/2^{15}.
[11]	**Reserved**	Reserved.
[10:8]	**TOUTPSC2**	**PDMA Channel 2 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 2 time-out clock source is HCLK/2^{8}. 001 = PDMA channel 2 time-out clock source is HCLK/2^{9}. 010 = PDMA channel 2 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 2 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 2 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 2 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 2 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 2 time-out clock source is HCLK/2^{15}.
[7]	**Reserved**	Reserved.
[6:4]	**TOUTPSC1**	**PDMA Channel 1 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 1 time-out clock source is HCLK/2^{8}. 001 = PDMA channel 1 time-out clock source is HCLK/2^{9}. 010 = PDMA channel 1 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 1 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 1 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 1 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 1 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 1 time-out clock source is HCLK/2^{15}.
[3]	**Reserved**	Reserved.

Bits	Description	
[2:0]	**TOUTPSC0**	**PDMA Channel 0 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 0 time-out clock source is HCLK/2^8. 001 = PDMA channel 0 time-out clock source is HCLK/2^9. 010 = PDMA channel 0 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 0 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 0 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 0 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 0 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 0 time-out clock source is HCLK/2^{15}.

PDMA Time-out Enable Register (PDMAx_TOUTEN)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOUTEN	PDMAx_BA+0x434	R/W	PDMA Time-out Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TOUTEN15	**TOUTEN14**	**TOUTEN13**	**TOUTEN12**	**TOUTEN11**	**TOUTEN10**	**TOUTEN9**	**TOUTEN8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOUTEN7	**TOUTEN6**	**TOUTEN5**	**TOUTEN4**	**TOUTEN3**	**TOUTEN2**	**TOUTEN1**	**TOUTEN0**

Bits	Description	
[31:n+1]	**Reserved**	Reserved.
[n] n=0 ~ 15	**TOUTENn**	**PDMA Time-out Enable Bits** 0 = PDMA Channel n time-out function Disabled. 1 = PDMA Channel n time-out function Enabled.

PDMA Time-out Interrupt Enable Register (PDMAx_TOUTIEN)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOUTIEN	PDMAx_BA+0x438	R/W	PDMA Time-out Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TOUTIEN15	**TOUTIEN14**	**TOUTIEN13**	**TOUTIEN12**	**TOUTIEN11**	**TOUTIEN10**	**TOUTIEN9**	**TOUTIEN8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOUTIEN7	**TOUTIEN6**	**TOUTIEN5**	**TOUTIEN4**	**TOUTIEN3**	**TOUTIEN2**	**TOUTIEN1**	**TOUTIEN0**

Bits	Description	
[31:n+1]	**Reserved**	Reserved.
[n] n=0 ~ 15	**TOUTIENn**	**PDMA Time-out Interrupt Enable Bits** 0 = PDMA Channel n time-out interrupt Disabled. 1 = PDMA Channel n time-out interrupt Enabled.

PDMA Scatter-gather Descriptor Table Base Address Register (PDMAx_SCATBA)

Register	Offset	R/W	Description	Reset Value
PDMAx_SCATBA	PDMAx_BA+0x43C	R/W	PDMA Scatter-gather Descriptor Table Base Address Register	0x2000_0000

31	30	29	28	27	26	25	24
SCATBA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SCATBA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:16]	**SCATBA**	**PDMA Scatter-gather Descriptor Table Address** In Scatter-gather mode, this is the base address for calculating the next link - list address. The next link address equation is Next Link Address = PDMA_SCATBA + PDMA_DSCTn_NEXT. **Note:** Only useful in Scatter-gather mode.
[15:0]	**Reserved**	Reserved.

PDMA Time-out Period Counter Register 0 (PDMAx_TOC0_1)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC0_1	PDMAx_BA+0x440	R/W	PDMA Time-out Counter Ch0 and Ch1 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC1							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TOC1							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TOC0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOC0							

Bits	Description	
[31:16]	**TOC1**	**Time-out Counter for Channel 1** This controls the period of time-out function for channel 1. The calculation unit is based on TOUTPSC1 (PDMA_TOUTPSC0_7[6:4]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC0**	**Time-out Counter for Channel 0** This controls the period of time-out function for channel 0. The calculation unit is based on TOUTPSC0 (PDMA_TOUTPSC0_7[2:0]) clock. Time-out period = (Period of time-out clock) * (16-bit TOCn),n=0,1..15.

PDMA Time-out Period Counter Register 1 (PDMAx_TOC2_3)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC2_3	PDMAx_BA+0x444	R/W	PDMA Time-out Counter Ch2 and Ch3 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC3							
23	22	21	20	19	18	17	16
TOC3							
15	14	13	12	11	10	9	8
TOC2							
7	6	5	4	3	2	1	0
TOC2							

Bits	Description	
[31:16]	**TOC3**	**Time-out Counter for Channel 3** This controls the period of time-out function for channel 3. The calculation unit is based on TOUTPSC3 (PDMA_TOUTPSC0_7[14:12]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC2**	**Time-out Counter for Channel 2** This controls the period of time-out function for channel 2. The calculation unit is based on TOUTPSC2 (PDMA_TOUTPSC0_7[10:8]) clock.

PDMA Time-out Period Counter Register 2 (PDMAx_TOC4_5)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC4_5	PDMAx_BA+0x448	R/W	PDMA Time-out Counter Ch4 and Ch5 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC5							
23	22	21	20	19	18	17	16
TOC5							
15	14	13	12	11	10	9	8
TOC4							
7	6	5	4	3	2	1	0
TOC4							

Bits	Description	
[31:16]	**TOC5**	**Time-out Counter for Channel 5** This controls the period of time-out function for channel 5. The calculation unit is based on TOUTPSC5 (PDMA_TOUTPSC0_7[22:20]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC4**	**Time-out Counter for Channel 4** This controls the period of time-out function for channel 4. The calculation unit is based on TOUTPSC4 (PDMA_TOUTPSC0_7[18:16]) clock.

PDMA Time-out Period Counter Register 3 (PDMAx_TOC6_7)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC6_7	PDMAx_BA+0x44C	R/W	PDMA Time-out Counter Ch6 and Ch7 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC7							
23	22	21	20	19	18	17	16
TOC7							
15	14	13	12	11	10	9	8
TOC6							
7	6	5	4	3	2	1	0
TOC6							

Bits	Description	
[31:16]	**TOC7**	**Time-out Counter for Channel 7** This controls the period of time-out function for channel 7. The calculation unit is based on TOUTPSC7 (PDMA_TOUTPSC0_7[30:28]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC6**	**Time-out Counter for Channel 6** This controls the period of time-out function for channel 6. The calculation unit is based on TOUTPSC6 (PDMA_TOUTPSC0_7[26:24]) clock. Time-out period = (Period of time-out clock) * (16-bit TOCn),n=0,1..15.

PDMA Time-out Period Counter Register 4 (PDMAx_TOC8_9)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC8_9	PDMAx_BA+0x450	R/W	PDMA Time-out Counter Ch8 and Ch9 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC9							
23	22	21	20	19	18	17	16
TOC9							
15	14	13	12	11	10	9	8
TOC8							
7	6	5	4	3	2	1	0
TOC8							

Bits	Description	
[31:16]	**TOC9**	**Time-out Counter for Channel 9** This controls the period of time-out function for channel 9. The calculation unit is based on TOUTPSC9 (PDMA_TOUTPSC8_15[6:4]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC8**	**Time-out Counter for Channel 8** This controls the period of time-out function for channel 8. The calculation unit is based on TOUTPSC8 (PDMA_TOUTPSC8_15[2:0]) clock.

PDMA Time-out Period Counter Register 5 (PDMAx_TOC10_11)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC10_11	PDMAx_BA+0x454	R/W	PDMA Time-out Counter Ch10 and Ch11 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC11							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TOC11							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TOC10							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOC10							

Bits	Description	
[31:16]	**TOC11**	**Time-out Counter for Channel 11** This controls the period of time-out function for channel 11. The calculation unit is based on TOUTPSC11 (PDMA_TOUTPSC8_15[14:12]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC10**	**Time-out Counter for Channel 10** This controls the period of time-out function for channel 10. The calculation unit is based on TOUTPSC10 (PDMA_TOUTPSC8_15[10:8]) clock.

PDMA Time-out Period Counter Register 6 (PDMAx_TOC12_13)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC12_13	PDMAx_BA+0x458	R/W	PDMA Time-out Counter Ch12 and Ch13 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC13							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TOC13							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TOC12							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOC12							

Bits	Description	
[31:16]	**TOC13**	**Time-out Counter for Channel 13** This controls the period of time-out function for channel 13. The calculation unit is based on TOUTPSC13 (PDMA_TOUTPSC8_15[22:20]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC12**	**Time-out Counter for Channel 12** This controls the period of time-out function for channel 12. The calculation unit is based on TOUTPSC12 (PDMA_TOUTPSC8_15[18:16]) clock.

PDMA Time-out Period Counter Register 7 (PDMAx_TOC14_15)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOC14_15	PDMAx_BA+0x45C	R/W	PDMA Time-out Counter Ch14 and Ch15 Register	0x0000_0000

31	30	29	28	27	26	25	24
TOC15							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TOC15							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TOC14							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOC14							

Bits	Description	
[31:16]	**TOC15**	**Time-out Counter for Channel 15** This controls the period of time-out function for channel 15. The calculation unit is based on TOUTPSC15 (PDMA_TOUTPSC8_15[30:28]) clock. The example of time-out period can refer TOC0 bit description.
[15:0]	**TOC14**	**Time-out Counter for Channel 14** This controls the period of time-out function for channel 14. The calculation unit is based on TOUTPSC14 (PDMA_TOUTPSC8_15[26:24]) clock.

PDMA Channel Reset Register (PDMAx_CHRST)

Register	Offset	R/W	Description	Reset Value
PDMAx_CHRST	PDMAx_BA+0x460	R/W	PDMA Channel Reset Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CH15RST	**CH14RST**	**CH13RST**	**CH12RST**	**CH11RST**	**CH10RST**	**CH9RST**	**CH8RST**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CH7RST	**CH6RST**	**CH5RST**	**CH4RST**	**CH3RST**	**CH2RST**	**CH1RST**	**CH0RST**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**CHnRST**	**Channel n Reset** 0 = Corresponding channel n is not reset. 1 = Corresponding channel n is reset.

IPDMA Time-out Prescaler Register (PDMAx_TOUTPSC8_15)

Register	Offset	R/W	Description	Reset Value
PDMAx_TOUTPSC8_15	PDMAx_BA+0x468	R/W	PDMA Time-out Prescaler Register(CH8 to CH15)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	TOUTPSC15			Reserved	TOUTPSC14		
23	22	21	20	19	18	17	16
Reserved	TOUTPSC13			Reserved	TOUTPSC12		
15	14	13	12	11	10	9	8
Reserved	TOUTPSC11			Reserved	TOUTPSC10		
7	6	5	4	3	2	1	0
Reserved	TOUTPSC9			Reserved	TOUTPSC8		

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TOUTPSC15**	**PDMA Channel 15 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 15 time-out clock source is HCLK/2^8. 001 = PDMA channel 15 time-out clock source is HCLK/2^9. 010 = PDMA channel 15 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 15 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 15 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 15 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 15 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 15 time-out clock source is HCLK/2^{15}.
[27]	**Reserved**	Reserved.
[26:24]	**TOUTPSC14**	**PDMA Channel 14 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 14 time-out clock source is HCLK/2^8. 001 = PDMA channel 14 time-out clock source is HCLK/2^9. 010 = PDMA channel 14 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 14 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 14 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 14 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 14 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 14 time-out clock source is HCLK/2^{15}.
[23]	**Reserved**	Reserved.
[22:20]	**TOUTPSC13**	**PDMA Channel 13 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 13 time-out clock source is HCLK/2^8. 001 = PDMA channel 13 time-out clock source is HCLK/2^9. 010 = PDMA channel 13 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 13 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 13 time-out clock source is HCLK/2^{12}.

Bits	Description	
		101 = PDMA channel 13 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 13 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 13 time-out clock source is HCLK/2^{15}.
[19]	**Reserved**	Reserved.
[18:16]	**TOUTPSC12**	**PDMA Channel 12 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 12 time-out clock source is HCLK/2^{8}. 001 = PDMA channel 12 time-out clock source is HCLK/2^{9}. 010 = PDMA channel 12 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 12 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 12 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 12 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 12 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 12 time-out clock source is HCLK/2^{15}.
[15]	**Reserved**	Reserved.
[14:12]	**TOUTPSC11**	**PDMA Channel 11 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 11 time-out clock source is HCLK/2^{8}. 001 = PDMA channel 11 time-out clock source is HCLK/2^{9}. 010 = PDMA channel 11 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 11 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 11 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 11 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 11 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 11 time-out clock source is HCLK/2^{15}.
[11]	**Reserved**	Reserved.
[10:8]	**TOUTPSC10**	**PDMA Channel 10 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 10 time-out clock source is HCLK/2^{8}. 001 = PDMA channel 10 time-out clock source is HCLK/2^{9}. 010 = PDMA channel 10 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 10 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 10 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 10 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 10 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 10 time-out clock source is HCLK/2^{15}.
[7]	**Reserved**	Reserved.
[6:4]	**TOUTPSC9**	**PDMA Channel 9 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 9 time-out clock source is HCLK/2^{8}. 001 = PDMA channel 9 time-out clock source is HCLK/2^{9}. 010 = PDMA channel 9 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 9 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 9 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 9 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 9 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 9 time-out clock source is HCLK/2^{15}.
[3]	**Reserved**	Reserved.

Bits	Description	
[2:0]	**TOUTPSC8**	**PDMA Channel 8 Time-out Clock Source Prescaler Bits** 000 = PDMA channel 8 time-out clock source is HCLK/2^8. 001 = PDMA channel 8 time-out clock source is HCLK/2^9. 010 = PDMA channel 8 time-out clock source is HCLK/2^{10}. 011 = PDMA channel 8 time-out clock source is HCLK/2^{11}. 100 = PDMA channel 8 time-out clock source is HCLK/2^{12}. 101 = PDMA channel 8 time-out clock source is HCLK/2^{13}. 110 = PDMA channel 8 time-out clock source is HCLK/2^{14}. 111 = PDMA channel 8 time-out clock source is HCLK/2^{15}.

PDMA Request Source Select Register 0 (PDMAx_REQSEL0_3)

Register	Offset	R/W	Description	Reset Value
PDMAx_REQSEL0_3	PDMAx_BA+0x480	R/W	PDMA Request Source Select Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	REQSRC3						
23	22	21	20	19	18	17	16
Reserved	REQSRC2						
15	14	13	12	11	10	9	8
Reserved	REQSRC1						
7	6	5	4	3	2	1	0
Reserved	REQSRC0						

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**REQSRC3**	**Channel 3 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 3. User can configure the peripheral setting by REQSRC3. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23]	**Reserved**	Reserved.
[22:16]	**REQSRC2**	**Channel 2 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 2. User can configure the peripheral setting by REQSRC2. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15]	**Reserved**	Reserved.
[14:8]	**REQSRC1**	**Channel 1 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 1. User can configure the peripheral setting by REQSRC1. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7]	**Reserved**	Reserved.
[6:0]	**REQSRC0**	**Channel 0 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 0. User can configure the peripheral by setting REQSRC0. 0 = Disable PDMA peripheral request. 1 = Reserved. 2 = Channel connects to USB_TX. 3 = Channel connects to USB_RX. 4 = Channel connects to UART0_TX. 5 = Channel connects to UART0_RX.

Bits	Description
	6 = Channel connects to UART1_TX. 7 = Channel connects to UART1_RX. 8 = Channel connects to UART2_TX. 9 = Channel connects to UART2_RX. 10=Channel connects to UART3_TX. 11 = Channel connects to UART3_RX. 12 = Channel connects to UART4_TX. 13 = Channel connects to UART4_RX. 14 = Channel connects to UART5_TX. 15 = Channel connects to UART5_RX. 16 = Channel connects to USCI0_TX. 17 = Channel connects to USCI0_RX. 18 = Reserved. 19 = Reserved. 20 = Channel connects to QSPI0_TX. 21 = Channel connects to QSPI0_RX. 22 = Channel connects to SPI0_TX. 23 = Channel connects to SPI0_RX. 24 = Channel connects to SPI1_TX. 25 = Channel connects to SPI1_RX. 26 = Channel connects to SPI2_TX. 27 = Channel connects to SPI2_RX. 28 = Channel connects to SPI3_TX. 29 = Channel connects to SPI3_RX. 30 = Channel connects to QSPI1_TX. 31 = Channel connects to QSPI1_RX. 32 = Channel connects to EPWM0_P1_RX. 33 = Channel connects to EPWM0_P2_RX. 34 = Channel connects to EPWM0_P3_RX. 35 = Channel connects to EPWM1_P1_RX. 36 = Channel connects to EPWM1_P2_RX. 37 = Channel connects to EPWM1_P3_RX. 38 = Channel connects to I2C0_TX. 39 = Channel connects to I2C0_RX. 40 = Channel connects to I2C1_TX. 41 = Channel connects to I2C1_RX. 44 = Channel connects to I2S0_TX. 45 = Channel connects to I2S0_RX. 46 = Channel connects to TMR0. 47 = Channel connects to TMR1. 48 = Channel connects to TMR2. 49 = Channel connects to TMR3. 50 = Channel connects to ADC0_RX. 51 = Channel connects to DAC0_TX. 52 = Channel connects to DAC1_TX. 53 = Channel connects to EPWM0_CH0_TX. 54 = Channel connects to EPWM0_CH1_TX. 55 = Channel connects to EPWM0_CH2_TX.

Bits	Description	
		56 = Channel connects to EPWM0_CH3_TX. 57 = Channel connects to EPWM0_CH4_TX. 58 = Channel connects to EPWM0_CH5_TX. 59 = Channel connects to EPWM1_CH0_TX. 60 = Channel connects to EPWM1_CH1_TX. 61 = Channel connects to EPWM1_CH2_TX. 62 = Channel connects to EPWM1_CH3_TX. 63 = Channel connects to EPWM1_CH4_TX. 64 = Channel connects to EPWM1_CH5_TX. 65 = Reserved. 66 = Channel connects to UART6_TX. 67 = Channel connects to UART6_RX. 68 = Channel connects to UART7_TX. 69 = Channel connects to UART7_RX. 70 = Channel connects to ADC1_RX. 71 = Channel connects to ACMP0. 72 = Channel connects to ACMP1. 73 = Channel connects to PSIO_TX. 74 = Channel connects to PSIO_RX. 79 = Channel connects to I2S1_TX. 80 = Channel connects to I2S1_RX. 81 = Channel connects to EINT0. 82 = Channel connects to EINT1. 83 = Channel connects to EINT2. 84 = Channel connects to EINT3. 85 = Channel connects to EINT4. 86 = Channel connects to EINT5. 87 = Channel connects to EINT6. 88 = Channel connects to EINT7. 89 = Channel connects to UART8_TX. 90 = Channel connects to UART8_RX. 91 = Channel connects to UART9_TX. 92 = Channel connects to UART9_RX. 93 = Channel connects to ADC2_RX. 94 = Channel connects to ACMP2. 95 = Channel connects to ACMP3. 96= Channel connects to SPI4_TX. 97= Channel connects to SPI4_RX. 98= Channel connects to SPI5_TX. 99= Channel connects to SPI5_RX. 100= Channel connects to SPI6_TX. 101= Channel connects to SPI6_RX. 102= Channel connects to SPI7_TX. 103= Channel connects to SPI7_RX. 104= Channel connects to SPI8_TX. 105= Channel connects to SPI8_RX. 106= Channel connects to SPI9_TX. 107= Channel connects to SPI9_RX.

Bits	Description	
		108= Channel connects to SPI10_TX. 109= Channel connects to SPI10_RX. Others = Reserved. **Note 1:** A peripheral cannot be assigned to two channels at the same time. **Note 2:** This field is useless when transfer between memory and memory.

PDMA Request Source Select Register 1 (PDMAx_REQSEL4_7)

Register	Offset	R/W	Description	Reset Value
PDMAx_REQSEL4_7	PDMAx_BA+0x484	R/W	PDMA Request Source Select Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	**REQSRC7**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved	**REQSRC6**						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**REQSRC5**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**REQSRC4**						

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**REQSRC7**	**Channel 7 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 7. User can configure the peripheral setting by REQSRC7. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23]	**Reserved**	Reserved.
[22:16]	**REQSRC6**	**Channel 6 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 6. User can configure the peripheral setting by REQSRC6. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15]	**Reserved**	Reserved.
[14:8]	**REQSRC5**	**Channel 5 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 5. User can configure the peripheral setting by REQSRC5. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7]	**Reserved**	Reserved.
[6:0]	**REQSRC4**	**Channel 4 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 4. User can configure the peripheral setting by REQSRC4. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.

PDMA Request Source Select Register 2 (PDMAx_REQSEL8_11)

Register	Offset	R/W	Description	Reset Value
PDMAx_REQSEL8_11	PDMAx_BA+0x488	R/W	PDMA Request Source Select Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	REQSRC11						
23	22	21	20	19	18	17	16
Reserved	REQSRC10						
15	14	13	12	11	10	9	8
Reserved	REQSRC9						
7	6	5	4	3	2	1	0
Reserved	REQSRC8						

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**REQSRC11**	**Channel 11 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 11. User can configure the peripheral setting by REQSRC11. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23]	**Reserved**	Reserved.
[22:16]	**REQSRC10**	**Channel 10 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 10. User can configure the peripheral setting by REQSRC10. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15]	**Reserved**	Reserved.
[14:8]	**REQSRC9**	**Channel 9 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 9. User can configure the peripheral setting by REQSRC9. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7]	**Reserved**	Reserved.
[6:0]	**REQSRC8**	**Channel 8 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 8. User can configure the peripheral setting by REQSRC8. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.

PDMA Request Source Select Register 3 (PDMAx_REQSEL12_15)

Register	Offset	R/W	Description	Reset Value
PDMAx_REQSEL12_15	PDMAx_BA+0x48C	R/W	PDMA Request Source Select Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	REQSRC15						
23	22	21	20	19	18	17	16
Reserved	REQSRC14						
15	14	13	12	11	10	9	8
Reserved	REQSRC13						
7	6	5	4	3	2	1	0
Reserved	REQSRC12						

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**REQSRC15**	**Channel 15 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 15. User can configure the peripheral setting by REQSRC15. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23]	**Reserved**	Reserved.
[22:16]	**REQSRC14**	**Channel 14 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 14. User can configure the peripheral setting by REQSRC14. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15]	**Reserved**	Reserved.
[14:8]	**REQSRC13**	**Channel 13 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 13. User can configure the peripheral setting by REQSRC13. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7]	**Reserved**	Reserved.
[6:0]	**REQSRC12**	**Channel 12 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 12. User can configure the peripheral setting by REQSRC12. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.

PDMA Stride Transfer Count Register n (PDMAx_STCRn)

Register	Offset	R/W	Description	Reset Value
PDMAx_STCR0	PDMAx_BA+0x500	R/W	Stride Transfer Count Register of PDMA Channel 0	0x0000_0000
PDMAx_STCR1	PDMAx_BA+0x508	R/W	Stride Transfer Count Register of PDMA Channel 1	0x0000_0000
PDMAx_STCR2	PDMAx_BA+0x510	R/W	Stride Transfer Count Register of PDMA Channel 2	0x0000_0000
PDMAx_STCR3	PDMAx_BA+0x518	R/W	Stride Transfer Count Register of PDMA Channel 3	0x0000_0000
PDMAx_STCR4	PDMAx_BA+0x520	R/W	Stride Transfer Count Register of PDMA Channel 4	0x0000_0000
PDMAx_STCR5	PDMAx_BA+0x528	R/W	Stride Transfer Count Register of PDMA Channel 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
11	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
STC							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**STC**	**PDMA Stride Transfer Count** The 16-bit register defines the stride transfer count of each row.

PDMA Address Stride Offset Control Register n (PDMAx_ASOCRn)

Register	Offset	R/W	Description	Reset Value
PDMAx_ASOCR0	PDMAx_BA+0x504	R/W	Address Stride Offset Register of PDMA Channel 0	0x0000_0000
PDMAx_ASOCR1	PDMAx_BA+0x50C	R/W	Address Stride Offset Register of PDMA Channel 1	0x0000_0000
PDMAx_ASOCR2	PDMAx_BA+0x514	R/W	Address Stride Offset Register of PDMA Channel 2	0x0000_0000
PDMAx_ASOCR3	PDMAx_BA+0x51C	R/W	Address Stride Offset Register of PDMA Channel 3	0x0000_0000
PDMAx_ASOCR4	PDMAx_BA+0x524	R/W	Address Stride Offset Register of PDMA Channel 4	0x0000_0000
PDMAx_ASOCR5	PDMAx_BA+0x52C	R/W	Address Stride Offset Register of PDMA Channel 5	0x0000_0000

31	30	29	28	27	26	25	24
DASOL							
11	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DASOL							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SASOL							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SASOL							

Bits	Description	
[31:16]	**DASOL**	**PDMA Destination Address Stride Offset Length** The 16-bit register defines the destination address stride transfer offset count of each row.
[15:0]	**SASOL**	**PDMA Source Address Stride Offset Length** The 16-bit register defines the source address stride transfer offset count of each row.

PDMA Address Interval Control Register n (PDMAx_AICTLn)

Register	Offset	R/W	Description	Reset Value
PDMAx_AICTL0	PDMAx_BA + 0x600	R/W	Address Interval Control Register of PDMA Channel 0	0x0000_0000
PDMAx_AICTL1	PDMAx_BA + 0x608	R/W	Address Interval Control Register of PDMA Channel 1	0x0000_0000

31	30	29	28	27	26	25	24
DAICNT							
11	22	21	20	19	18	17	16
DAICNT							
15	14	13	12	11	10	9	8
SAICNT							
7	6	5	4	3	2	1	0
SAICNT							

Bits	Description	
[31:16]	**DAICNT**	**PDMA Destination Address Interval Count** The 16-bit register defines the destination address interval count of each row.
[15:0]	**SAICNT**	**PDMA Source Address Interval Count** The 16-bit register defines the source address interval count of each row.

PDMA Repeat Count Register n (PDMAx_RCNTn)

Register	Offset	R/W	Description	Reset Value
PDMAx_RCNT0	PDMAx_BA + 0x604	R/W	Repeat Count Register of PDMA Channel 0	0x0000_0000
PDMAx_RCNT1	PDMAx_BA + 0x60C	R/W	Repeat Count Register of PDMA Channel 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
11	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RCNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RCNT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RCNT**	**PDMA Repeat Count** The 16-bit register defines the repeat times of block transfer.

6.10 Timer Controller (TMR)

6.10.1 Overview

The timer controller includes four 32-bit timers, Timer0 ~ Timer3, allowing user to easily implement a timer control for applications. The timer can perform functions, such as frequency measurement, delay timing, clock generation, and event counting by external input pins, and interval measurement by external capture pins.

The timer controller also provides four PWM generators. Each PWM generator supports two PWM output channels in independent mode and complementary mode. The output state of PWM output pin can be control by pin mask, polarity and break control, and dead-time generator.

6.10.2 Features

6.10.2.1 Timer Function Features

- Four sets of 32-bit timers, each timer having one 24-bit up counter and one 8-bit prescale counter
- Independent clock source for each timer
- Provides one-shot, periodic, toggle-output and continuous counting operation modes
- 24-bit up counter value is readable through CNT (TIMERx_CNT[23:0])
- Supports event counting function
- Supports 3-bit capture input noise filter
- 24-bit capture value is readable through CAPDAT (TIMERx_CAP[23:0])
- Supports external capture pin event for interval measurement
- Supports external capture pin event to reset 24-bit up counter
- Supports chip wake-up from Idle/Power-down mode if a timer interrupt signal is generated
- Support Timer0 ~ Timer3 time-out interrupt signal or capture interrupt signal to trigger EPWM, BPWM, EADC, DAC and PDMA function
- Supports internal capture triggered while internal ACMP output signal transition
- Supports internal clock (HIRC, LIRC) and external clock (HXT, LXT) for capture event
- Supports Inter-Timer trigger mode
- Supports event counting source from ACMP or internal USB SOF signal

6.10.2.2 PWM Function Features

- Supports maximum clock frequency up to maximum PCLK
- Supports independent mode for PWM generator with two output channels
- Supports complementary mode for PWM generator with paired PWM output channel
  - 12-bit dead-time insertion with 12-bit prescale
- Supports 12-bit prescale from 1 to 4096
- Supports 16-bit PWM counter
  - Up, down and up-down count operation type
  - One-shot or auto-reload counter operation mode
- Supports mask function and tri-state enable for each PWM output pin

- Supports brake function
  - Brake source from pin, analog comparator and system safety events (clock failed, Brown-out detection, SRAM parity error and CPU lockup)
  - Brake pin noise filter control for brake source
  - Edge detect brake source to control brake state until brake status cleared
  - Level detect brake source to auto recover function after brake condition removed
- Supports interrupt on the following events:
  - PWM zero point, period point, up-count compared or down-count compared point events
  - Brake condition happened
- Supports PWM output accumulator event to trigger PDMA transfer
- Supports PWM output accumulator event to stop PWM counting
- Supports trigger EADC on the following events:
  - PWM zero point, period, zero or period point, up-count compared or down-count compared point events
- Supports External Pin event to trigger PWM counter action

6.10.3 Block Diagram

The timer controller block diagram and clock control are shown as follows.

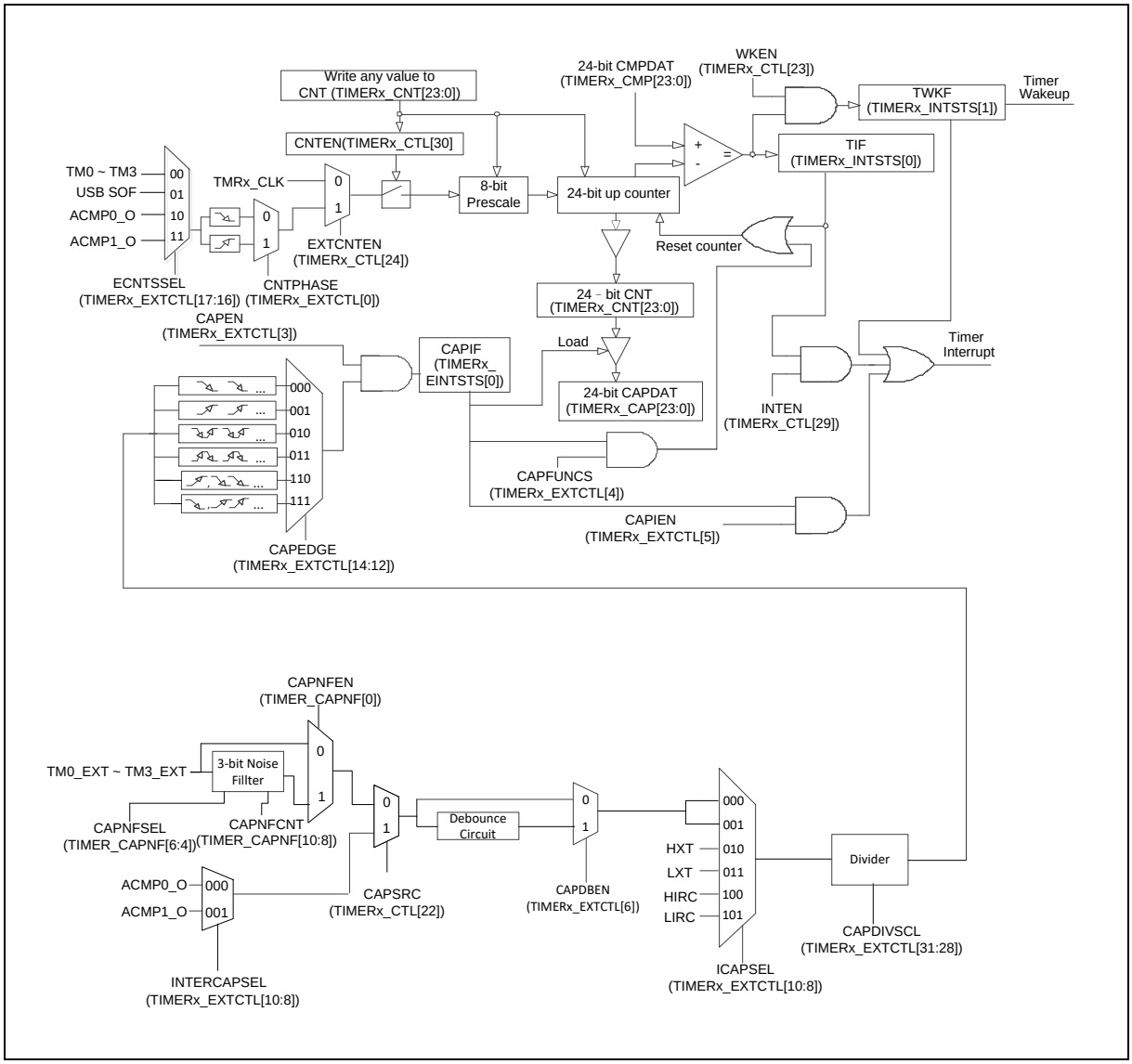


Figure 6.10-1 Timer Controller Block Diagram

Set FUNMODE (TIMERx_ALTCTL[0]) 0 to enable timer mode. The clock source of Timer0 ~ Timer3 in timer mode can be enabled in TMRxCKEN (CLK_APBCLK0[5:2]) and selected as different frequency in TMR0SEL (CLK_CLKSEL1[10:8]) for Timer0, TMR1SEL (CLK_CLKSEL1[14:12]) for Timer1, TMR2SEL (CLK_CLKSEL1[18:16]) for Timer2 and TMR3SEL (CLK_CLKSEL1[22:20]) for Timer3 as Figure 6.10-2.

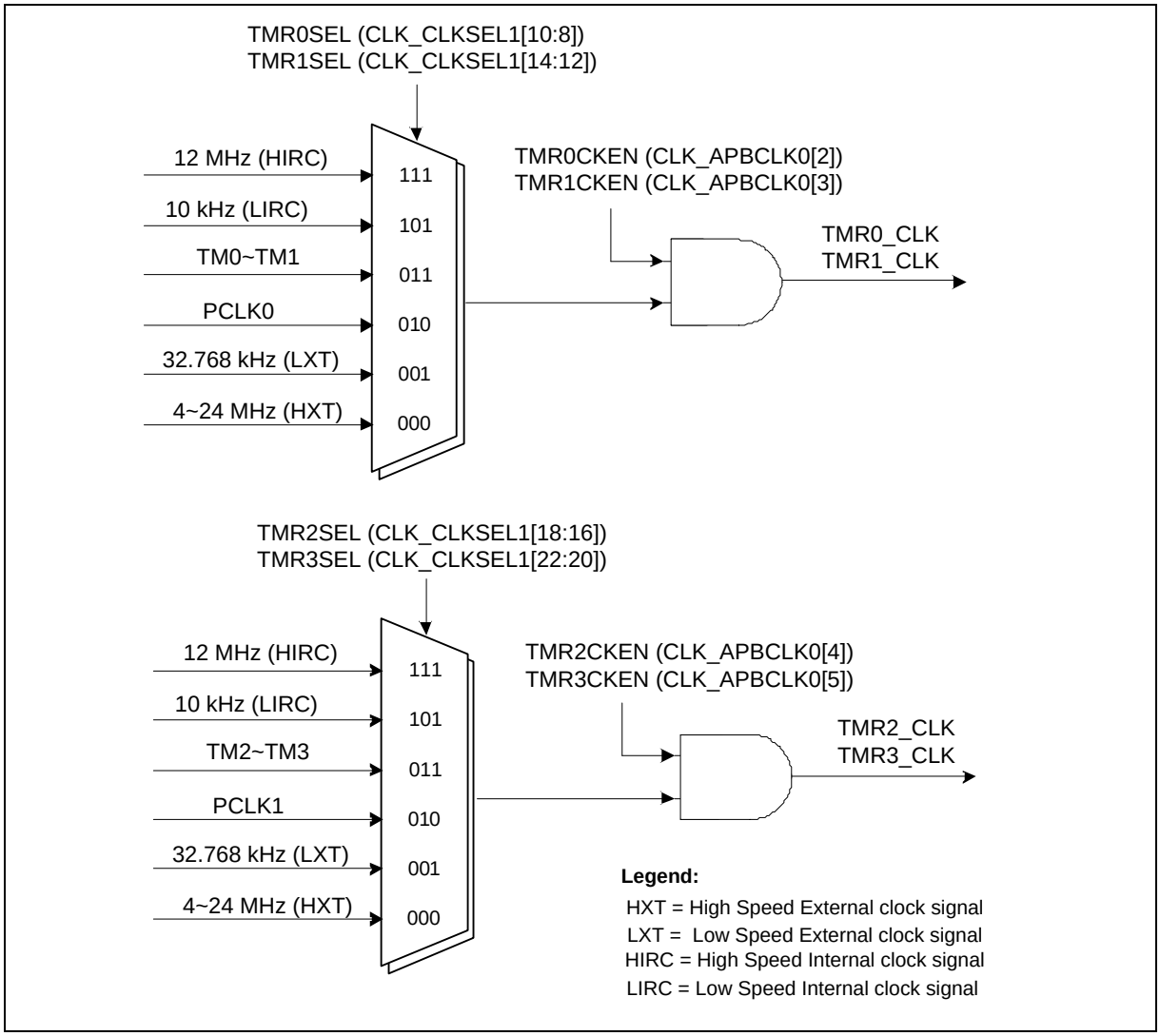


Figure 6.10-2 Clock Source of Timer Controller

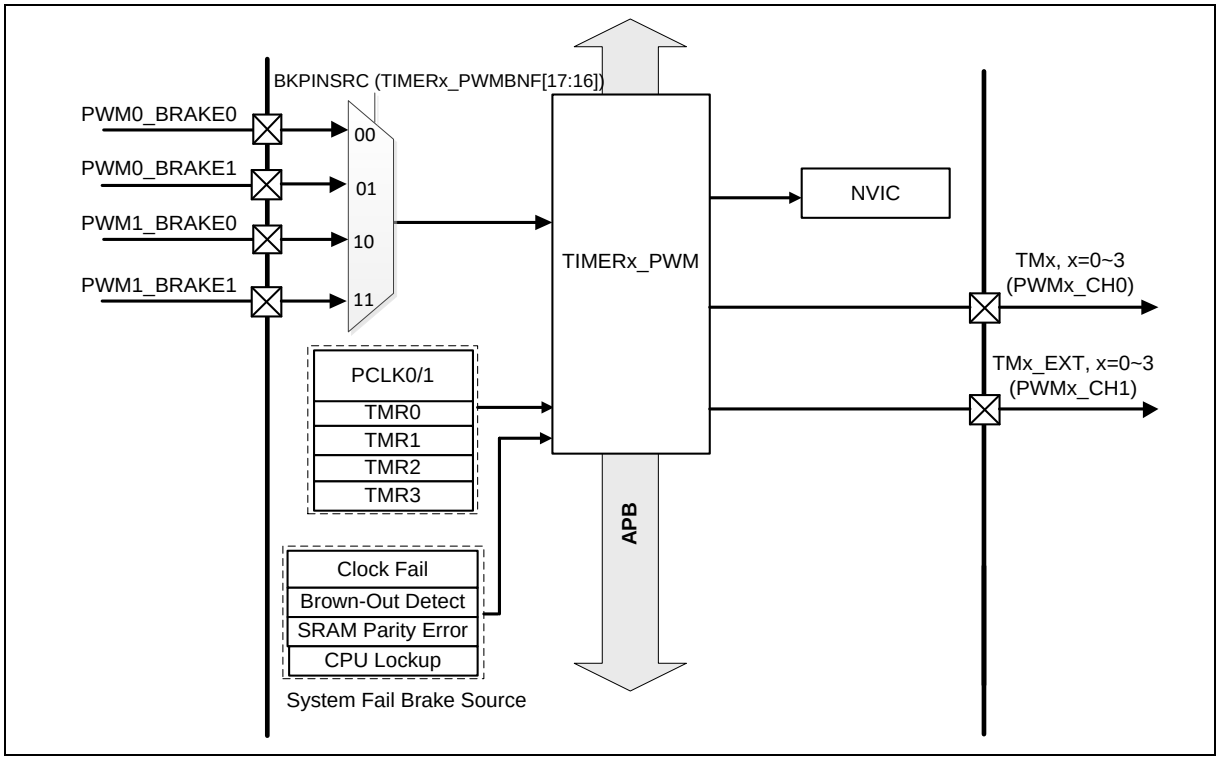


Figure 6.10-3 PWM Generator Overview Block Diagram

Set FUNMODE (TIMERx_ALTCTL[0]) 1 to enable PWM mode. The clock source of Timer0 ~ Timer3 in PWM mode can be enabled in TMRxCKEN (CLK_APBCLK0[5:2]). TMR0_CLK and TMR1_CLK clock sources are fixed to PCLK0. TMR2_CLK and TMR3_CLK clock sources are fixed to PCLK1. PWM system clock frequency will be PCLKx frequency as Figure 6.10-4 .

The clock source of PWM counter (TIMERx_PWMCLK) can be selected from PWM system clock (TMRx_CLK) or Timer interrupt events (TMRx_INT) as Figure 6.10-5.

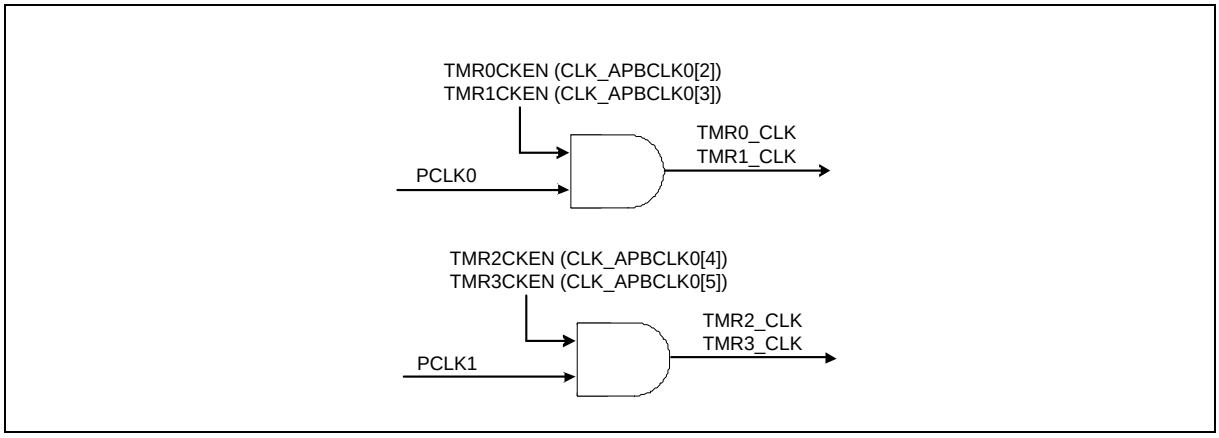


Figure 6.10-4 PWM System Clock Source Control

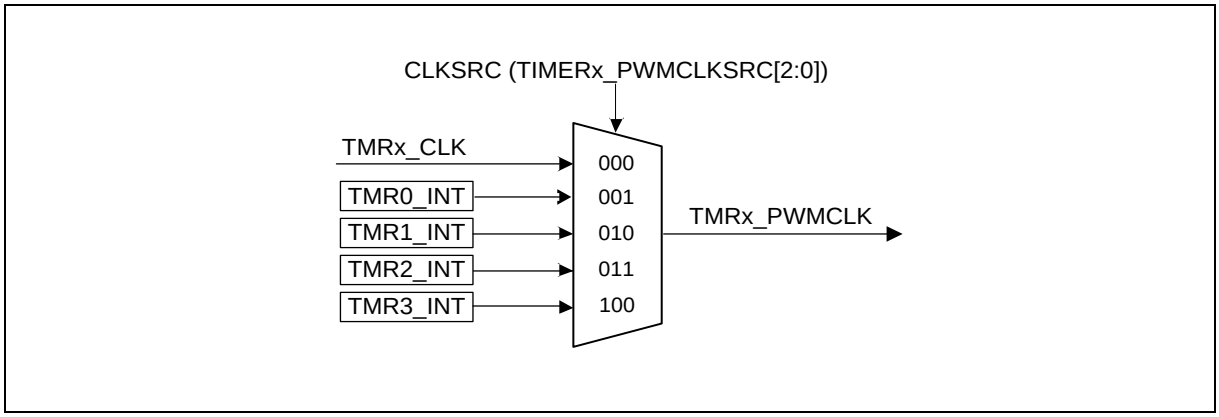


Figure 6.10-5 PWM Counter Clock Source Control

Figure 6.10-6 and Figure 6.10-7 illustrate the architecture of PWM independent mode and complementary mode. Both independent mode and complementary mode supports PWMx_CH0 and PWMx_CH1 output channels in each PWM generator.

When counter counts to 0, PERIOD (TIMERx_PWMPERIOD[15:0]) or equal to CMP (TIMERx_PWMCMPDAT[15:0]), relative events will be generated. These events are passed to corresponding generators to generate PWM pulse (Pulse Generator), interrupt signal (Interrupt Generator) and trigger signal (Trigger Generator) for EADC to start conversion. Output Control block is used to decide PWM pulse output; brake function in Output Control block also generates interrupt events. And Dead-Time Control is available only in PWM complementary mode.

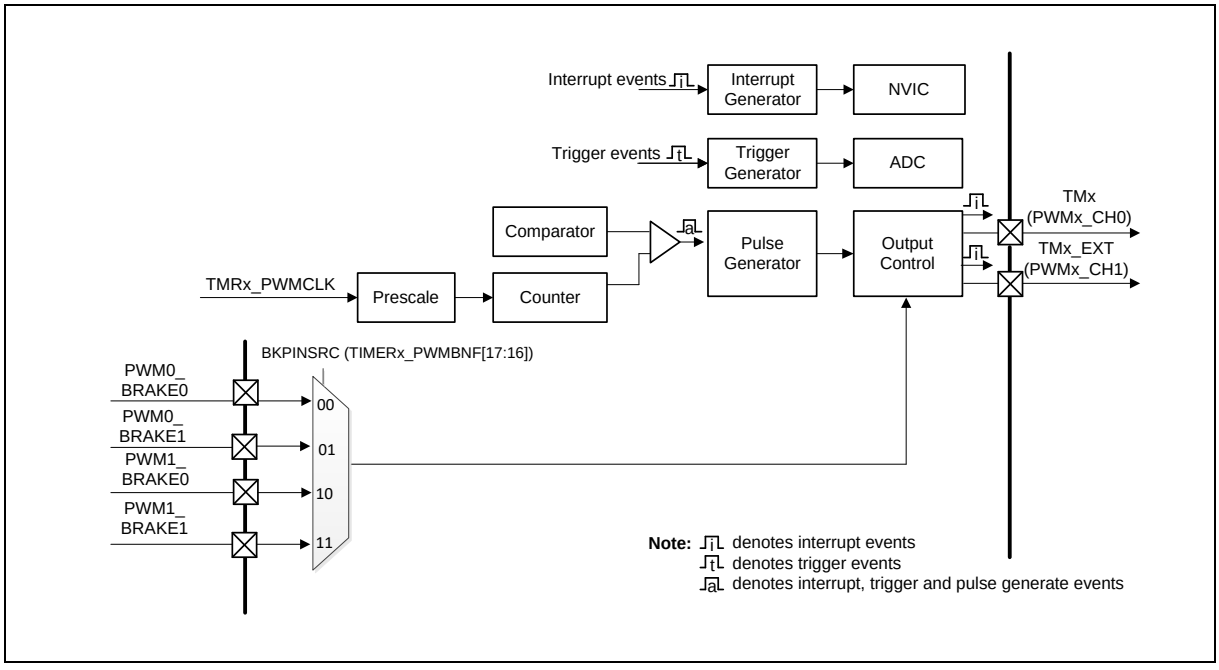


Figure 6.10-6 PWM Independent Mode Architecture Diagram

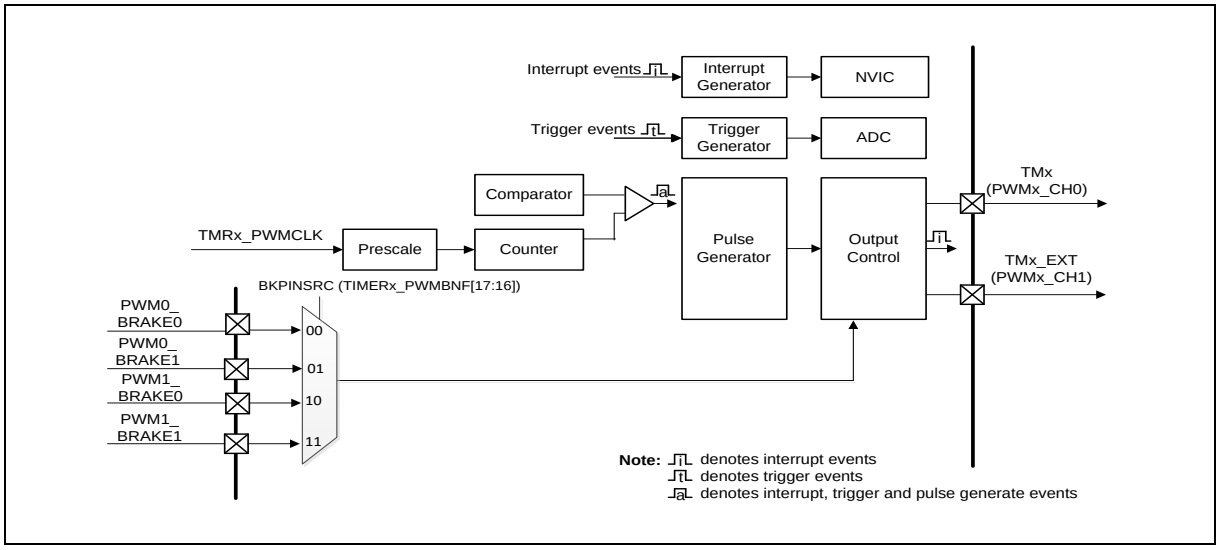


Figure 6.10-7 PWM Complementary Mode Architecture Diagram

6.10.4 Basic Configuration

Set FUNMODE (TIMERx_ALTCTL[0]) 0 to enable timer mode. The clock source of Timer0 ~ Timer3 in timer mode can be enabled in TMRxCKEN (CLK_APBCLK0[5:2]) and selected as different frequency in TMR0SEL (CLK_CLKSEL1[10:8]) for Timer0, TMR1SEL (CLK_CLKSEL1[14:12]) for Timer1, TMR2SEL (CLK_CLKSEL1[18:16]) for Timer2 and TMR3SEL (CLK_CLKSEL1[22:20]) for Timer3.

Set FUNMODE (TIMERx_ALTCTL[0]) 1 to enable PWM mode. The clock source of Timer0 ~ Timer3 in PWM mode can be enabled in TMRxCKEN (CLK_APBCLK0[5:2]). TMR0_CLK and TMR1_CLK clock sources are fixed to PCLK0. TMR2_CLK and TMR3_CLK clock sources are fixed to PCLK1.

6.10.4.1 TIMER01 Basic Configuration

- Clock source Configuration
  - Enable TIMER0 peripheral clock in TMR0CKEN (CLK_APBCLK0[2]).
  - Enable TIMER1 peripheral clock in TMR1CKEN (CLK_APBCLK0[3]).
- Reset Configuration
  - Reset TIMER0 controller in TMR0RST (SYS_IPRST1[2]).
  - Reset TIMER1 controller in TMR1RST (SYS_IPRST1[3]).

6.10.4.2 TIMER23 Basic Configuration

- Clock source Configuration
  - Enable TIMER2 peripheral clock in TMR2CKEN (CLK_APBCLK0[4]).
  - Enable TIMER3 peripheral clock in TMR3CKEN (CLK_APBCLK0[5]).
- Reset Configuration
  - Reset TIMER2 controller in TMR2RST (SYS_IPRST1[4]).
  - Reset TIMER3 controller in TMR3RST (SYS_IPRST1[5]).

6.10.5 Timer Functional Description

6.10.5.1 Timer Interrupt Flag

The timer controller supports the following interrupt flags: TIF (TIMERx_INTSTS[0]), which is set while the timer counter value CNT (TIMERx_CNT[23:0]) matches the timer compared value CMPDAT

(TIMERx_CMP[23:0]), and CAPIF (TIMERx_EINTSTS[0]), which is set when the transition on the external TMx_EXT pin, or the internal ACMP0_O, ACMP1_O signal and clock source (HXT, LXT, HIRC, LIRC) is associated with CAPEDGE (TIMERx_EXTCTL[14:12]) setting. The TWKF (TIMERx_INTSTS[1]) bit indicates the interrupt wake-up flag status of timer. Set WKEN (TIMERx_CTL[23]) to 1 to use the wake-up function.

6.10.5.2 Timer Counting Mode

The timer controller provides four timer counting modes: one-shot, periodic, toggle-output and continuous counting operation modes:

6.10.5.3 One–Shot Mode

If the timer controller is configured at one-shot mode (TIMERx_CTL[28:27] is 00) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) will be set to 1, CNT value and CNTEN bit is cleared automatically by timer controller then timer counting operation stops. In the meantime, if the INTEN (TIMERx_CTL[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also.

User can monitor the counter reset operation active by RSTACT (TIMERx_CNT[31]). And set ICEDEBUG (TIMERx_CTL[31]) to 1 that disable ICE debug mode acknowledgement effects TIMER counting.

6.10.5.4 Periodic Mode

If the timer controller is configured at periodic mode (TIMERx_CTL[28:27] is 01) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) will be set to 1, CNT value will be cleared automatically by timer controller and timer counter operates counting again. In the meantime, if the INTEN (TIMERx_CTL[29]) bit is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. In this mode, the timer controller operates counting and compares with CMPDAT value periodically until the CNTEN bit is cleared by user.

User can set PERIOSEL (TIMERx_CTL[20]) to select timer counting behavior if updating CMPDAT value at periodic mode.

6.10.5.5 Toggle-Output Mode

If the timer controller is configured at toggle-output mode (TIMERx_CTL[28:27] is 10) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. The counting operation of toggle-output mode is almost the same as periodic mode, except toggle-output mode has associated TM0 ~ TM3 or TM0_EXT ~ TM3_EXT pin to output signal while specify TIF (TIMERx_INTSTS[0]) is set. Use can set TGPINSEL (TIMERx_CTL[21]) to choose Tx or Tx_EXT as toggle-output pin. Thus, the toggle-output signal is 50% duty cycle.

6.10.5.6 Continuous Counting Mode

If the timer controller is configured at continuous counting mode (TIMERx_CTL[28:27] is 11) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) will be set to 1 and CNT value keeps up counting. In the meantime, if the INTEN (TIMERx_CTL[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. User can change different CMPDAT value immediately without disabling timer counting and restarting timer counting in this mode.

For example, CMPDAT value is set as 80, first. The TIF will be set to 1 when the CNT value is equal to 80, timer counter is kept counting and CNT value will not goes back to 0, it continues to count 81, 82, 83, ⋯ to 2^{24} -1, 0, 1, 2, 3, ⋯ to 2^{24} -1 again and again. Next, if user programs CMPDAT value as 200 and clears TIF, the TIF will set to 1 again when CNT value reaches 200. At last, user programs CMPDAT as 500 and clears TIF, the TIF will be set to 1 again when CNT value reaches 500.

In this mode, the timer counting is continuous. So, this operation mode is called as continuous counting mode.

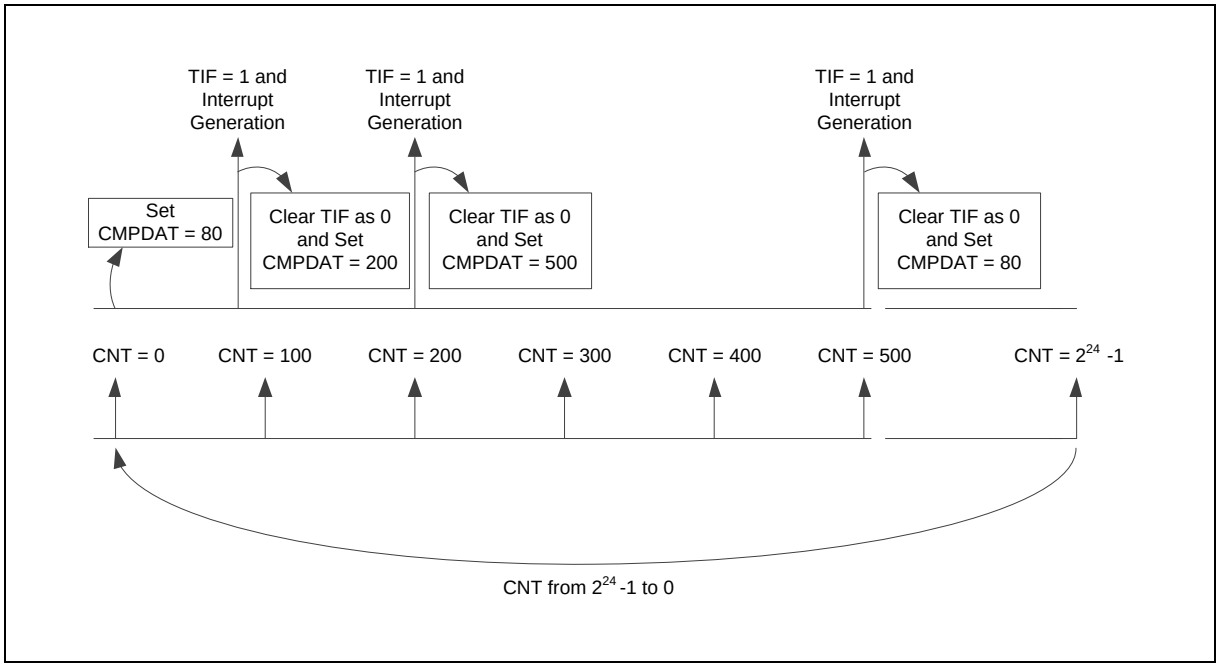


Figure 6.10-8 Continuous Counting Mode

6.10.5.7 Event Counting Mode

The timer controller also provides an application which can count the input event from TMx (x= 0~3) pin and the number of event will reflect to CNT (TIMERx_CNT[23:0]) value. It is also called as event counting function. In this function, EXTCNTEN (TIMERx_CTL[24]) should be set and the timer peripheral clock source should be set as PCLK.

If ECNTSSEL (TIMERx_EXTCTL[17:16]) is 00, the event counter source is from external TMx pin. User can enable or disable TMx pin de-bounce circuit by setting CNTDBEN (TIMERx_EXTCTL[7]). The input event frequency should be less than 1/3 PCLK if TMx pin de-bounce disabled or less than 1/8 PCLK if TMx pin de-bounce enabled to assure the returned CNT value is correct, and user can also select edge detection phase of TMx pin by setting CNTPHASE (TIMERx_EXTCTL[0]) bit.

In event counting mode, the timer counting operation mode can be selected as one-shot, periodic, continuous counting mode, or toggle-output mode on TMx_EXT pin by setting TGPINSEL (TIMERx_CTL[21]) to 1.

If ECNTSSEL (TIMERx_EXTCTL[17:16]) is 01, the event counter source will be generated by USB device detect the start-of-frame (SOF) packet. Please refer to USB device specifications.

If ECNTSSEL (TIMERx_EXTCTL[17:16]) is 10 or 11, the event counter source will be generated from internal ACMP0 output signal if ECNTSSEL is 10, or from internal ACMP1 output signal if ECNTSSEL is 11.

6.10.5.8 Capture Mode

The event capture function is used to load CNT (TIMERx_CNT[23:0]) value to CAPDAT (TIMERx_CAP[23:0]) value while edge transition detected on capture source. In this mode, CAPEN (TIMERx_EXTCTL[3]) should be as 1 and CAPFUNCS (TIMERx_EXTCTL[4]) should be as 0 to enable event capture function and the timer peripheral clock source should be set as PCLK.

If CAPSRC (TIMERx_CTL[22]) is 0, the capture event is triggered by TMx_EXT pin transition. The TMx_EXT signals will be filtered by a 3-bit noise filter. User can enable the noise filter function by CAPNFEN (TIMERx_CAPNF[0]) bit, and noise filter sampling clock can be selected by setting CAPNFSEL (TIMERx_CAPNF[6:4]) bits to fit different noise properties. Moreover, by setting the CAPNFCNT (TIMERx_CAPNF[10:8]) bits, user can define by how many sampling clock cycles a filter

will recognize the effective edge of the TMx_EXT signal.

User can also enable or disable TMx_EXT pin de-bounce circuit by setting CAPDBEN (TIMERx_EXTCTL[6]). The transition frequency of TMx_EXT pin should be less than 1/3 PCLK if TMx_EXT pin de-bounce disabled or less than 1/8 PCLK if TMx_EXT pin de-bounce enabled to assure the capture function can work normally, and user can also select edge transition detection of TMx_EXT pin by setting CAPEDGE (TIMERx_EXTCTL[14:12]).

If CAPSRC (TIMERx_CTL[22]) is 1, the capture event can be triggered by internal output signal transition on ACMP0 if ICAPSEL (TIMERx_EXTCTL[10:8]) is 000, or ACMP1 if ICAPSEL is 001; Other capture sources are HXT, LXT, HIRC, LIRC if ICAPSEL is 010, 011, 100, 101 respectively. User can switch capture source only when original source and target source is all off. For example, if user wants to switch LXT or LIRC, both of capture source should be set off first.

In event capture mode, user does not consider what timer counting operation mode is selected, and the capture event occurred only if edge transition on capture source is detected.

User can enable CAPIEN (TIMERx_EXTCTL[5]) to use capture interrupt function. When the capture source edge transition meets setting, CAPIF is high.

User must consider the Timer will keep register TIMERx_CAP unchanged and drop the new capture value if the CPU does not clear the CAPIF status.

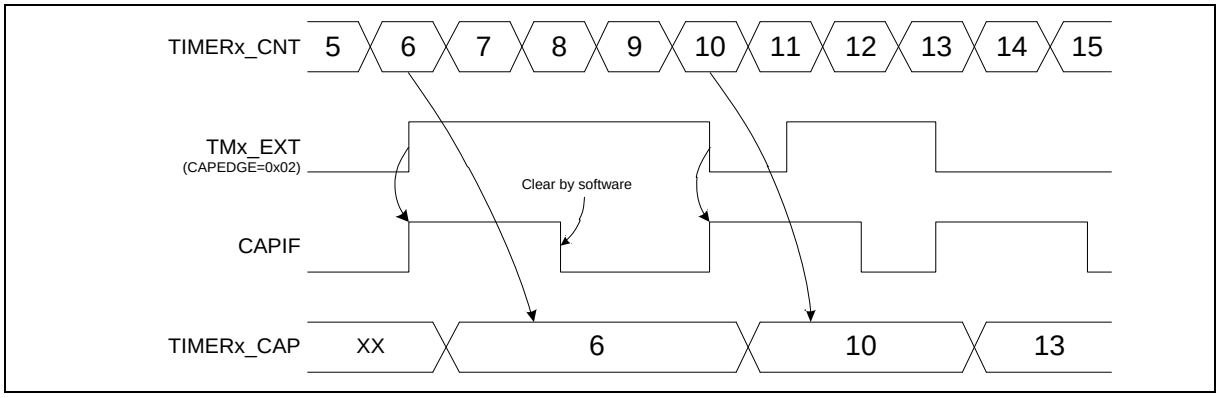


Figure 6.10-9 Capture Mode on TMx_EXT Pin

6.10.5.9 Capture and Reset Counter Mode

The timer controller also provides reset counter function to reset CNT (TIMERx_CNT[23:0]) value while capture event is generated. In this mode, CAPFUNCS (TIMERx_EXTCTL[4]) should be as 1, and user must set CAPSRC and ICAPSEL to select capture source to obtain the capture data and reset the counter value.

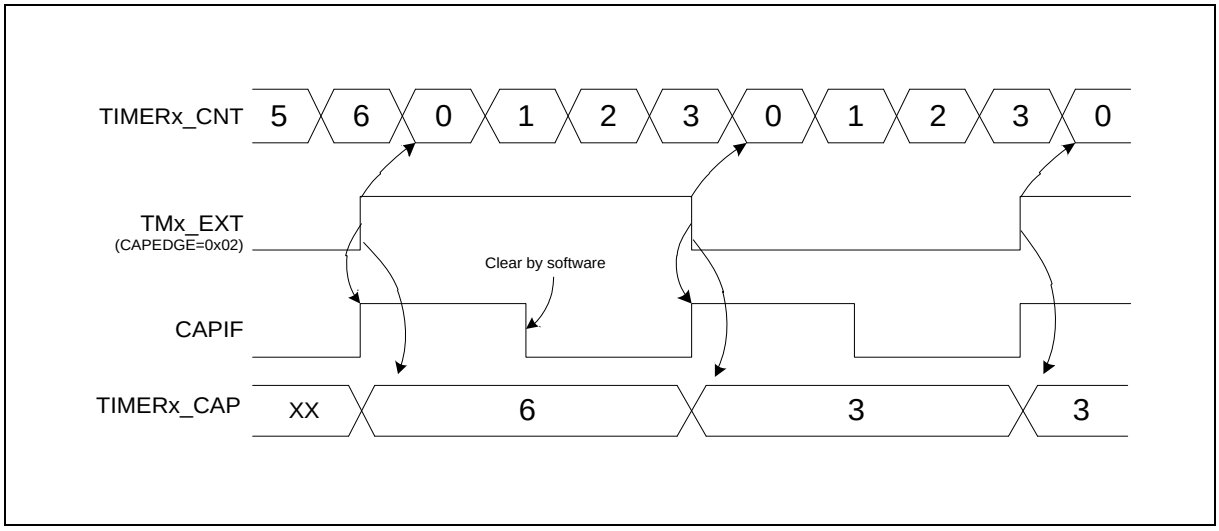


Figure 6.10-10 Reset Counter Mode

6.10.5.10 Timer Trigger Function

The timer controller provides timer time-out interrupt or capture interrupt to trigger EPWM/BPWM, EADC, DAC and PDMA.

If TRGSSEL (TIMERx_TRGCTL[0]) is 0, time-out interrupt signal is used to trigger EPWM/BPWM, EADC, DAC and PDMA. If TRGSSEL (TIMERx_TRGCTL [0]) is 1, capture interrupt signal is used to trigger EPWM/BPWM, EADC, DAC and PDMA.

When the TRGPWM (TIMERx_TRGCTL [1]) is set, if the timer interrupt signal is generated, the timer controller will generate a trigger pulse as EPWM and BPWM external clock source.

When the TRGEADC (TIMERx_TRGCTL [2]) is set, if the timer interrupt signal is generated, the timer controller will trigger EADC to start converter.

When the TRGDAC (TIMERx_TRGCTL [3]) is set, if the timer interrupt signal is generated, the timer controller will trigger DAC to start converter.

When the TRGPDMA (TIMERx_TRGCTL [4]) is set, if the timer interrupt signal is generated, the timer controller will trigger PDMA.

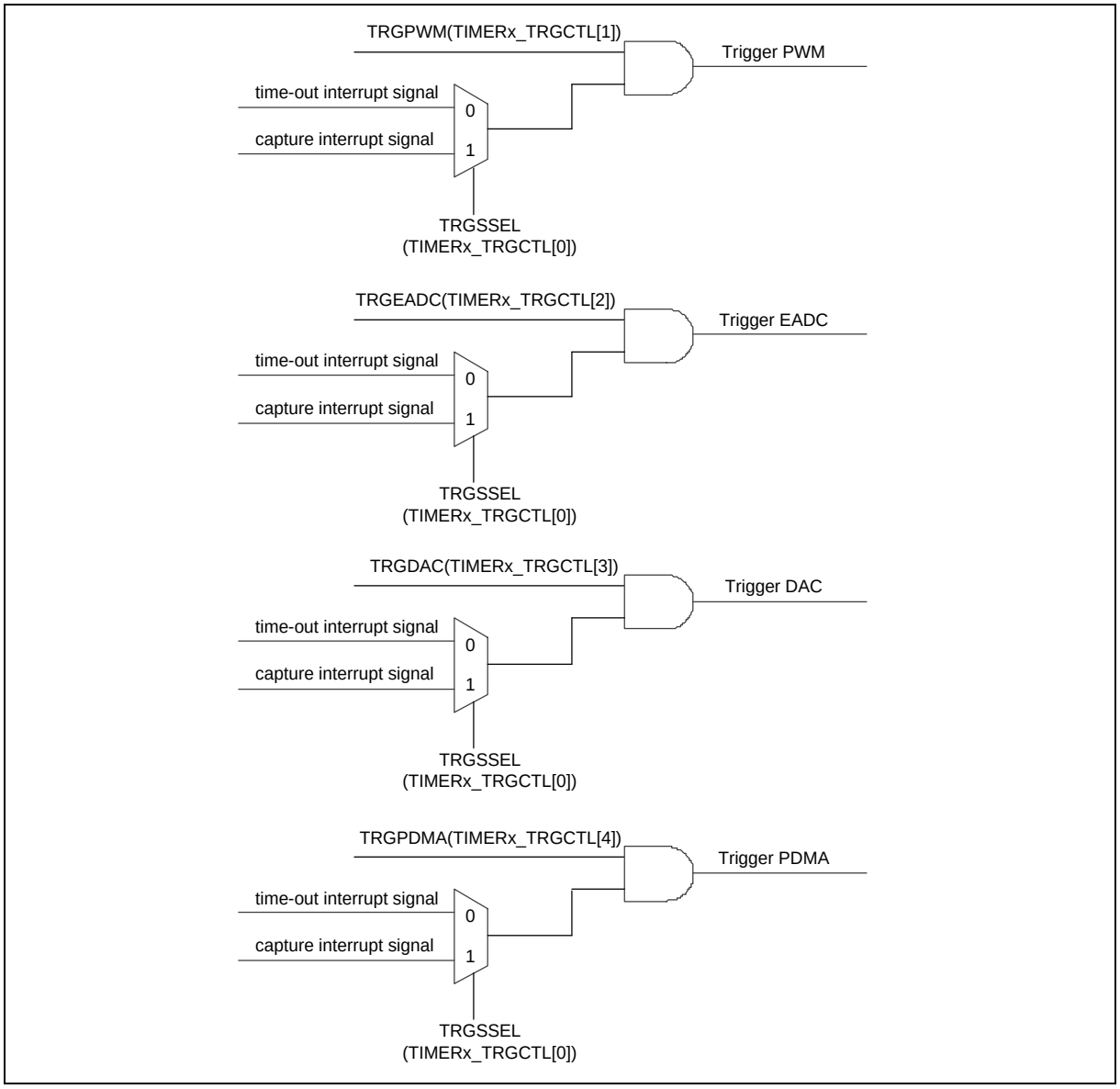


Figure 6.10-11 Internal Timer Trigger

6.10.5.11 Inter-Timer Trigger Capture Mode

In this mode, the Timer0/2 will be forced in event counting mode, counting with external event, and will generate an internal signal (INTR_TMR_TRG) to trigger Timer1/3 start or stop counting. Also, the Timer1/3 will be forced in capture mode and start/stop trigger-counting by Timer0/2 counter status.

Setting Timer0 Inter-Timer Trigger Capture enabled, trigger-counting capture function is forced on Timer1. Setting Timer2 Inter-Timer Trigger enabled, trigger-counting capture function is forced on Timer3.

Start Trigger

While INTRGEN (TIMERx_CTL[19]) in Timer0/2 is set, the Timer0/2 will make a rising-edge transition of INTR_TMR_TRG while Timer0/2 24-bit counter value (CNT) is counting from 0x0 to 0x1 and Timer1/3 counter will start counting immediately and automatically.

Stop Trigger

When Timer0/2 CNT reaches the Timer0/2 CMPDAT value, the Timer0/2 will make a falling-edge transition of INTR_TMR_TRG. Then Timer0/2 counter mode function will be disabled and INTRGEN (TIMERx_CTL[19]) will be cleared by hardware then Timer1/3 will stop counting also. At the same time, the Timer1/3 CNT value will be saved into Timer1/3 CAPDAT (TIMERx_CAP[23:0]).

User can use inter-timer trigger mode to measure the period of external event (TMx) more precisely. Figure 6.10-12 shows the sample flow of Inter-Timer Trigger Capture Mode for Timer0 as event counting mode and Timer1 as trigger-counting capture mode.

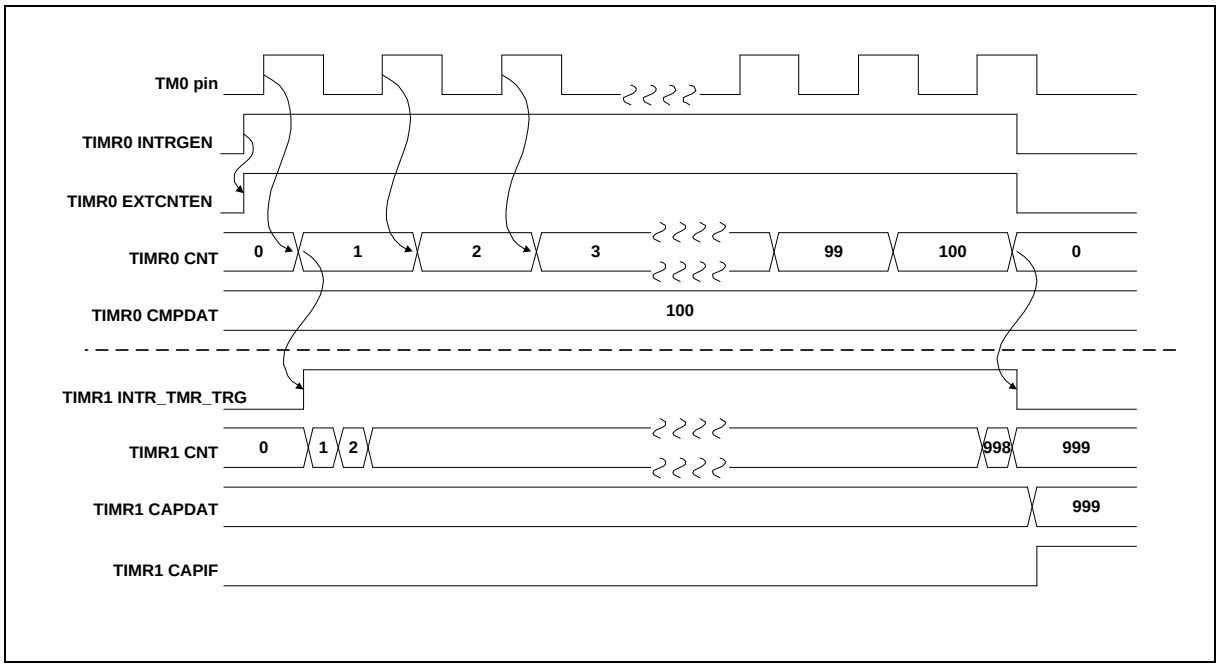


Figure 6.10-12 Inter-Timer Trigger Capture Timing

6.10.6 PWM Functional Description

6.10.6.1 PWM Prescale

The PWM prescale is used to divide clock source, and the clock of PWM counter is divided by (CLKPSC+1). The prescale is set by CLKPSC (TIMERx_PWMCLKPSC[11:0]). Figure 6.10-13 shows an example of PWM prescale waveform in up count type.

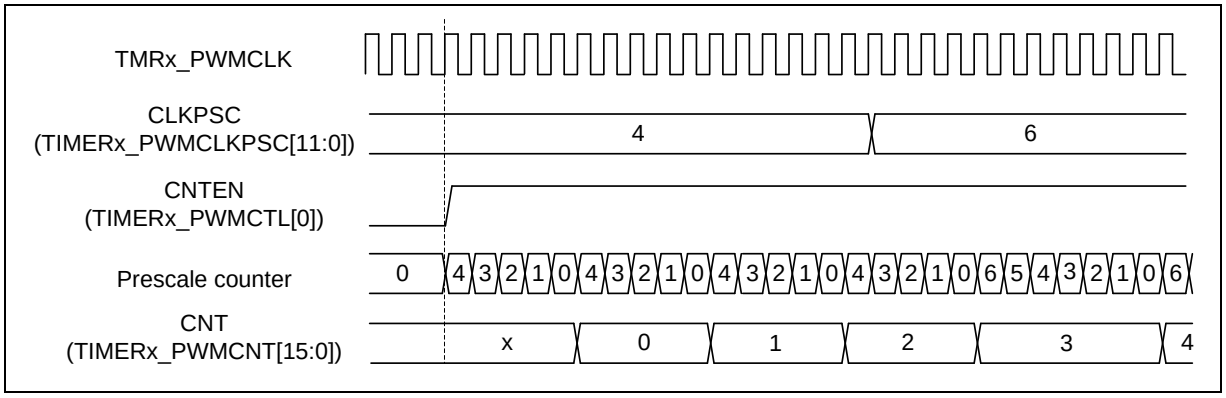


Figure 6.10-13 PWM Prescale Waveform in Up Count Type

6.10.6.2 PWM Counter

The PWM supports three counter types operation: up count, down count and up-down count types.

6.10.6.3 Up Count Type

When PWM counter is set to up count type, CNTTYPE (TIMERx_PWMCTL[2:1]) is 0x0, it starts up-counting from 0 to PERIOD (TIMERx_PWMPERIOD[15:0]). The current counter value can be read from the CNT (TIMERx_PWMCNT[15:0]). PWM generates a zero-point event when both counter and prescale counts to 0. PWM generates a period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.10-14 shows an example of PWM up count type, where PWM period time is (PERIOD+1) * (CLKPSC+1) * TMRx_PWMCLK.

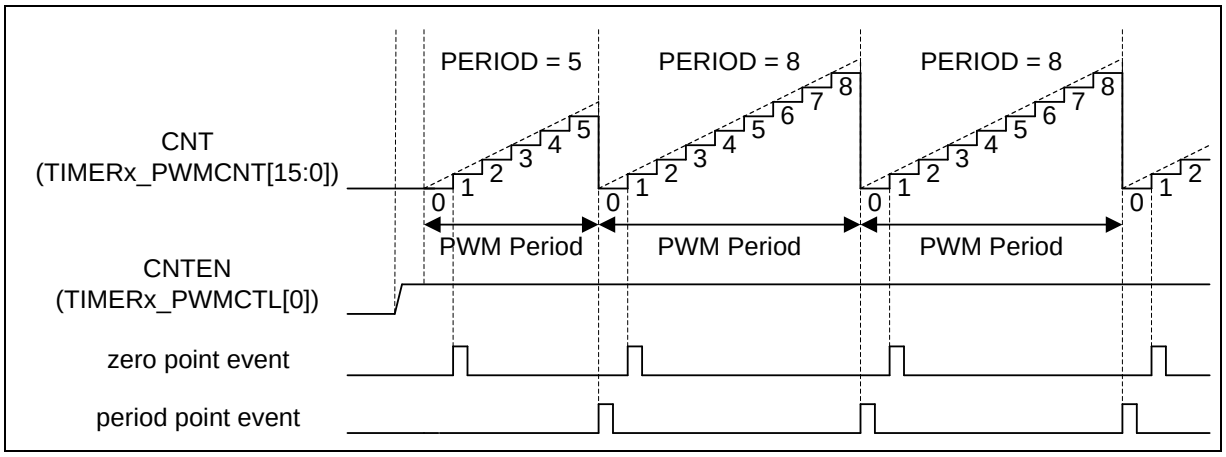


Figure 6.10-14 PWM Up Count Type

6.10.6.4 Down Count Type

When PWM counter is set to down count type, CNTTYPE (TIMERx_PWMCTL[2:1]) is 0x1, it starts down-counting from PERIOD to 0, current counter value can be read from CNT (TIMERx_PWMCNT[15:0]). PWM generates a zero-point event when both counter and prescale counts to 0. PWM generates a period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.10-15 is an example of PWM down count type, where PWM period time is (PERIOD+1) * (CLKPSC+1) * TMRx_PWMCLK.

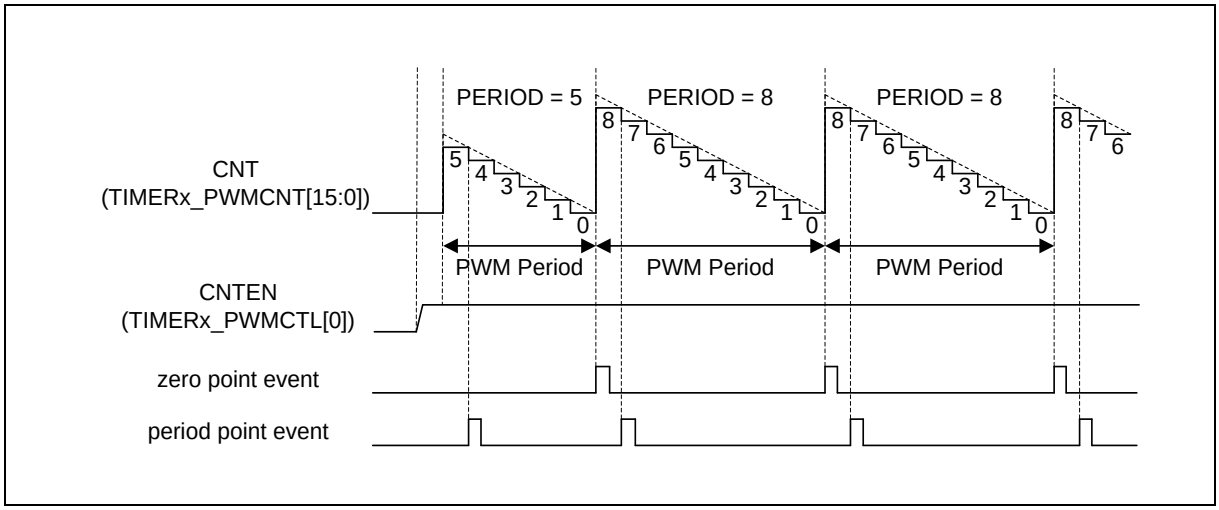


Figure 6.10-15 PWM Down Count Type

6.10.6.5 Up-Down Count Type

When PWM counter is set to up-down count type, CNTTYPE (TIMERx_PWMCTL[2:1]) is 0x2, it starts

counting up from 0 to PERIOD and then starts counting down to 0. The current counter value can be read from CNT (TIMERx_PWMCNT[15:0]). PWM generates a zero-point event when both counter and prescale counts to 0. PWM generates a center point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.10-16 shows an example of PWM up-down count type, where PWM period time is (2 * PERIOD) * (CLKPSC+1) * TMRx_PWMCLK. The DIRF (TIMERx_PWMCNT [16]) is counter direction indicator flag, where 1 is up counting, and 0 is down counting.

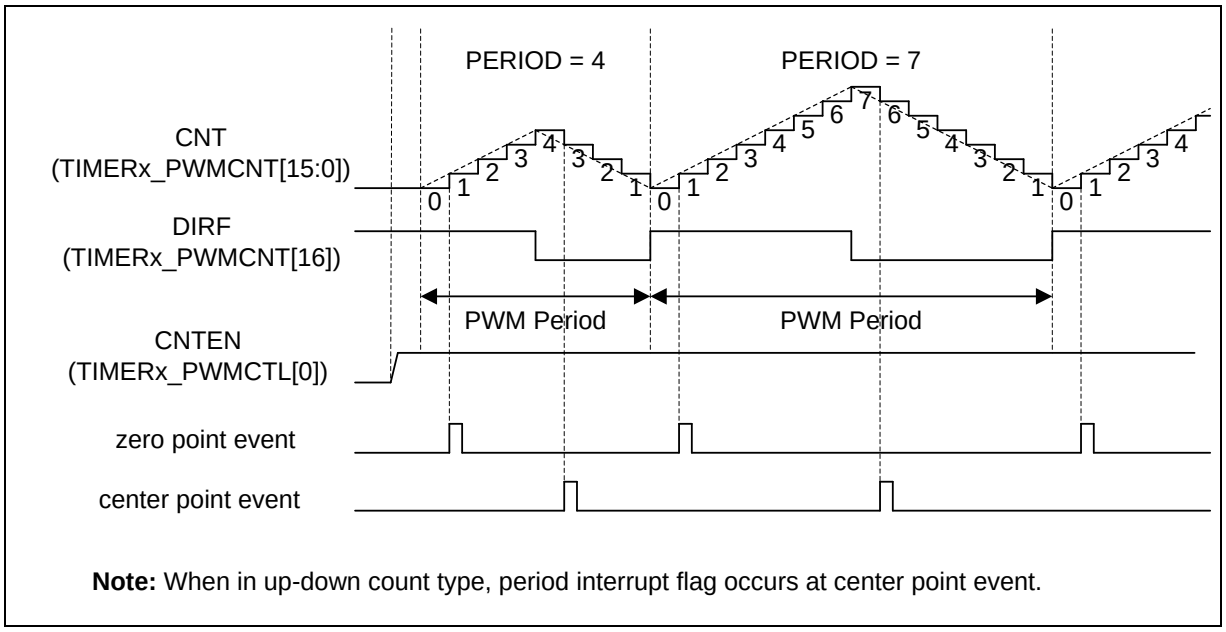


Figure 6.10-16 PWM Up-Down Count Type

6.10.6.6 PWM Counter Operation mode

The PWM counter supports two operation modes: one-shot mode and auto-reload mode. PWM counter will operate in one-shot mode if CNTMODE (TIMERx_PWMCTL[3]) bit is set to 1, and operate in auto-reload mode if CNTMODE bit is set to 0.

In both modes, CMP (TIMERx_PWMCMPDAT [15:0]) and PERIOD (TIMERx_PWMPERIOD[15:0]) should be written first and then set CNTEN (TIMERx_PWMCTL[0]) bit to 1 to start counter running.

In one-shot mode, PWM counter value will reload to default value according count type after one PWM period is completed. User can write CMP to continuous one-shot operation to generate next one-shot pulse once no matter current one-shot counter is running or completed.

In auto-reload mode, PWM counter is continuous running with current active PERIOD and CMP. If user sets PERIOD to zero in auto-reload mode, PWM counter value will reload to default value according count type after one PWM period is completed and stop counting.

6.10.6.7 PWM Comparator

The CMP (TIMERx_PWMCMPDAT[15:0]) is comparator register of PWM. The CMP value is continuously compared to the corresponding counter value. When the counter is equal to CMP, PWM generates a compared point event. This event will generate PWM output pulse, interrupt signal or trigger EADC start convert. In up-down count type, two events will be generated in a PWM period as shown in Figure 6.10-17. The CMPU is up count compared point event and CMPD is down count compared point event.

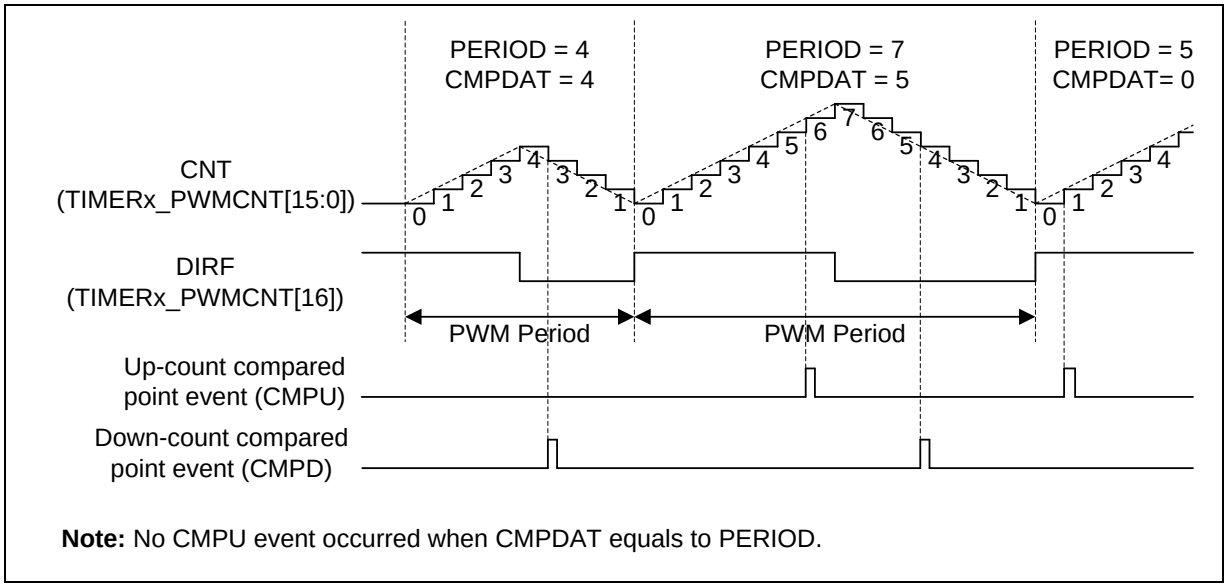


Figure 6.10-17 PWM Comparator Events in Up-Down Count Type

6.10.6.8 Period Loading Mode

When the IMMLDEN (TIMERx_PWMCTL[9]) bit set to 0, the PWM operates at Period Loading mode. The PWM provides PBUF (TIMERx_PWMPBUF[15:0]), the active PERIOD buffer register and CMPBUF (TIMERx_PWMCMPBUF[15:0]), the active CMP buffer register. In period loading mode, both PERIOD (TIMERx_PWMPERIOD[15:0]) and CMP (TIMERx_PWMCMPDAT[15:0]) will be loaded to their active PBUF and CMPBU register while each PWM period is completed. Figure 6.10-18 shows period loading timing of up count type, where PERIOD DATA0 denotes the initial data of PERIOD, PERIOD DATA1 denotes the first updated PERIOD data by user and so on, CMP also follows this rule. The following steps are the sequence of Figure 6.10-18.

1. Write CMP DATA1 to CMP at point 1.
2. Period load CMP DATA1 to CMPBUF at the end of PWM period at point 2.
3. Write PERIOD DATA1 to PERIOD at point 3.
4. Period load PERIOD DATA1 to PBUF at the end of PWM period at point 4.
5. Write PERIOD DATA2 to PERIOD at point 5.
6. Period load PERIOD DATA2 to PBUF at the end of PWM period at point 6.

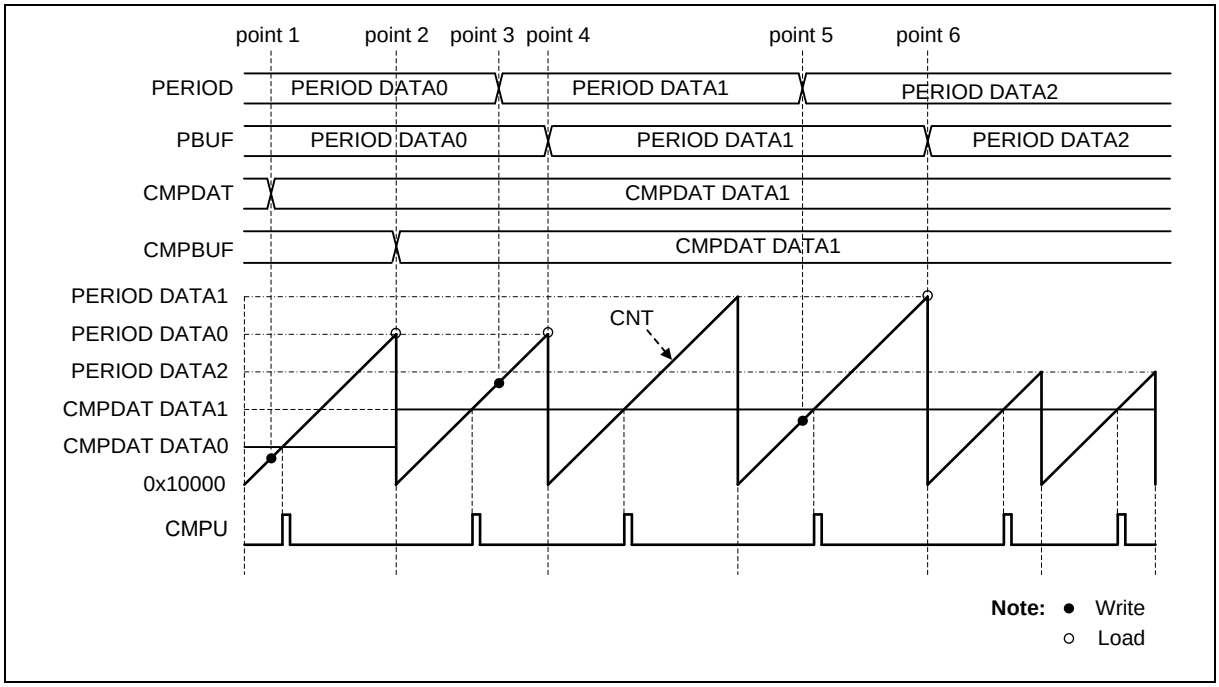


Figure 6.10-18 Period Loading Mode with Up Count Type

6.10.6.9 Immediately Loading Mode

When the IMMLDEN (TIMERx_PWMCTL[9]) bit set to 1, PWM operates at immediately loading mode. In immediately loading mode, when user update PERIOD (TIMERx_PWMPERIOD[15:0]) or CMP (TIMERx_PWMCMPDAT[15:0]), PERIOD or CMP will be loaded to active PBUF (TIMERx_PWMPBUF[15:0]) or CMPBUF (TIMERx_PWMCMPBUF[15:0]) after current counter count is completed. If the updated PERIOD value is less than current counter value, counter will count wraparound. The following steps are the sequence of Figure 6.10-19.

1. Write CMP DATA1 at point 1 and hardware will load CMP DATA1 to CMPBUF after current counter count is completed.
2. Write PERIOD DATA1 at point 2 and PERIOD DATA1 is greater than current counter value, PWM counter will continuously count until equal to PERIOD DATA1 to complete one PWM period.
3. Write PERIOD DATA2 at point 3 and PERIOD DATA2 is less than the current counter value, PWM counter will continuously count to maximum counter value 0x1FFFF and wraparound from 0x10000 to PERIOD DATA2 to complete one PWM period.

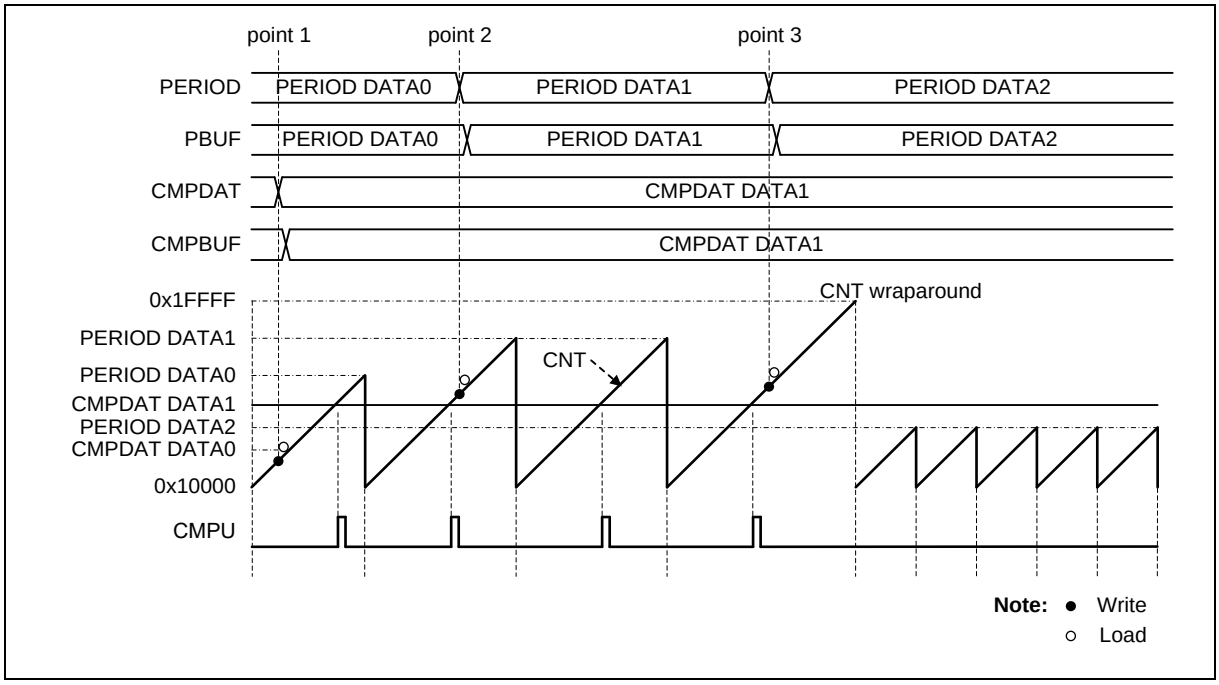


Figure 6.10-19 Immediately Loading Mode with Up Count Type

6.10.6.10 PWM Pulse Generator

The PWM pulse generator uses counter and comparator events to generate PWM output pulse. The events are zero point and period point in up count type and down count type, center point in up-down count type and counter equal to comparator point in three count types.

Each event point can generate PWM output waveform in different count type as shown in Figure 6.10-20, Figure 6.10-21 and Figure 6.10-22.

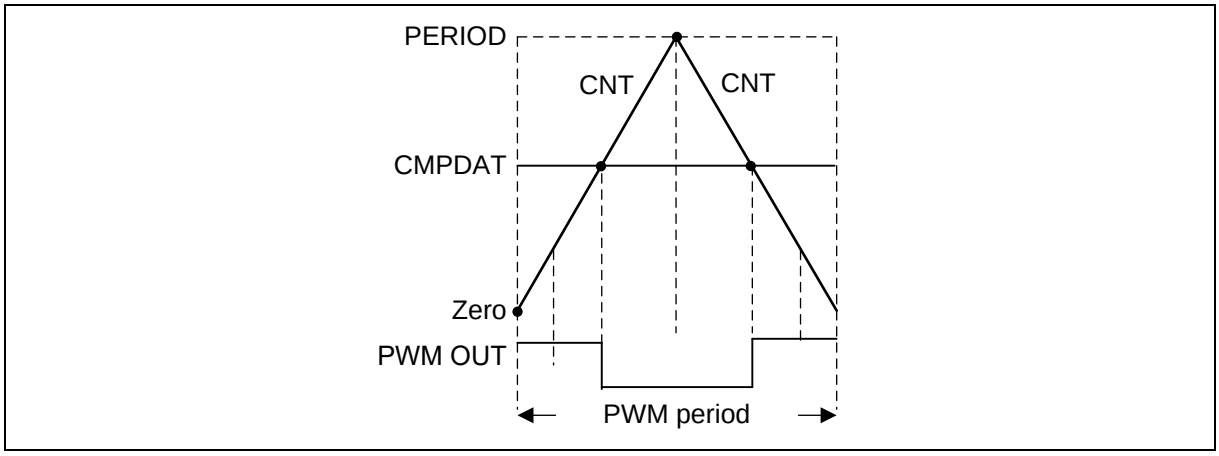


Figure 6.10-20 PWM Pulse Generation in Up-Down Count Type

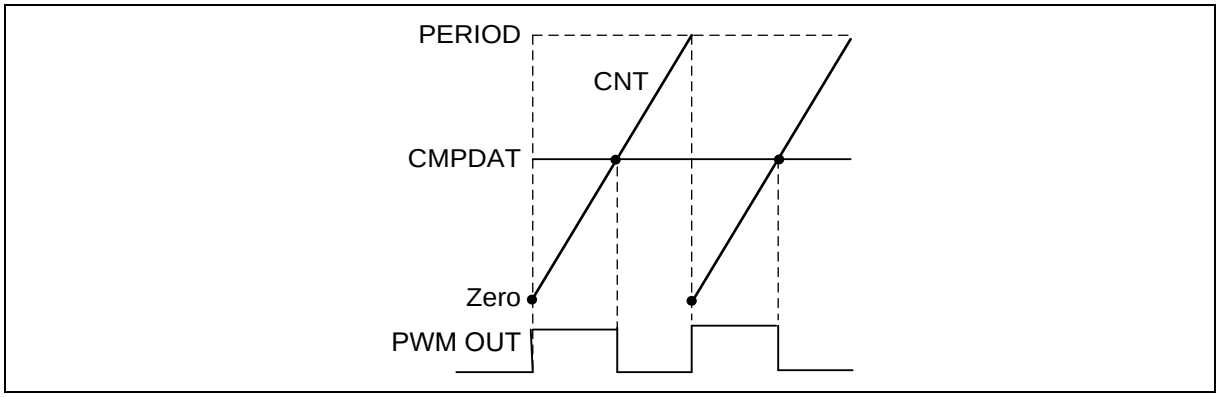


Figure 6.10-21 PWM Pulse Generation in Up Count Type

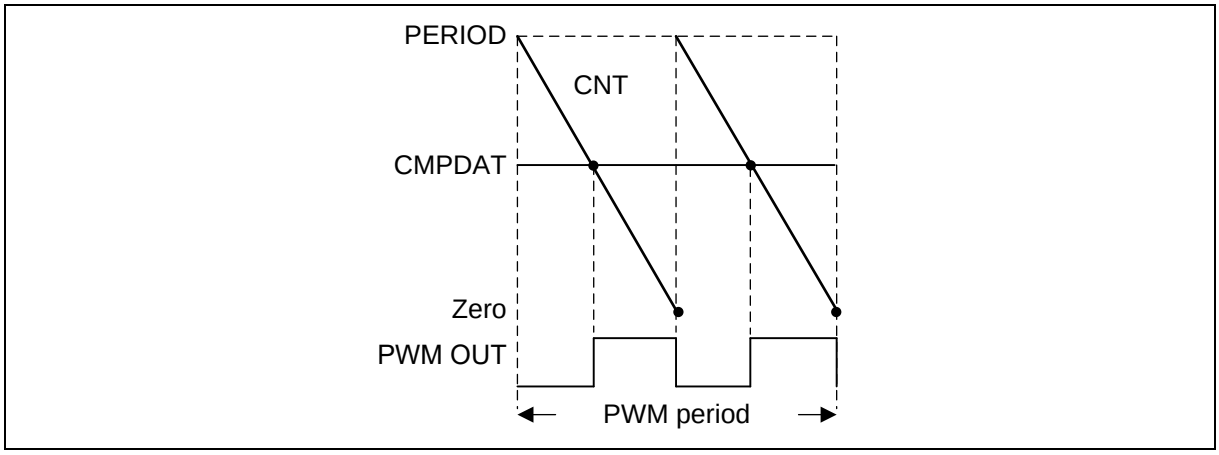


Figure 6.10-22 PWM Pulse Generation in Down Count Type

The PWM generation events may be sometimes generated at the same time, as the reason, events priority between different counter types should be taken care are listed in Table 6.10-1, Table 6.10-2 and Table 6.10-3, including event priority in up count type, event priority in down count type and event priority in up-down count type.

Priority	Zero and CMPU Point Event (CMP = 0)	PWM Output
1 (High)	Compare up event	Low
2 (Low)	Zero event	High

Table 6.10-1 PWM Pulse Generation Event Priority in Up Count Type

Priority	Zero and CMPD Point Event (CMP = 0)	PWM Output
1 (High)	Zero event	Low
2 (Low)	Compare down event	High
Priority	Period and CMPD point event (CMP = PERIOD)	PWM output

1 (High)	Compare down event	High
2 (Low)	Period event	Low

Table 6.10-2 PWM Pulse Generation Event Priority in Down Count Type

Priority	CMPU And CMPU Point Event (CMP = PERIOD)	PWM Output
1 (High)	Compare down event	High
2 (Low)	Compare up event	Low

Table 6.10-3 PWM Pulse Generation Event Priority in Up-Down Count Type

According to event priority limitation, the PWM generator can support 0% and 100% duty cycle PWM output waveform only in up count and up-down count type. Figure 6.10-23 is an example about PWM duty cycle from 0% to 100% in up count type and up-down count type where PERIOD is 4 with different CMP value.

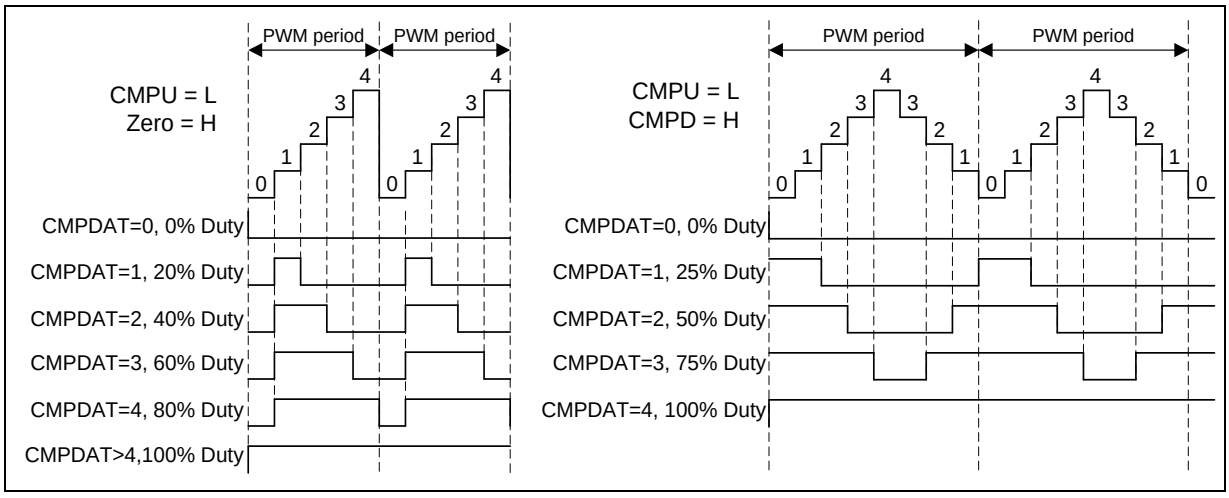


Figure 6.10-23 PWM 0% to 100% Duty Cycle in Up Count Type and Up-Down Count Type

6.10.6.11 PWM Output Mode

The PWM supports two output modes: independent mode which may be applied to DC motor system, complementary mode with dead-time insertion which may be used in the application of AC induction motor and permanent magnet synchronous motor.

6.10.6.12 Independent Mode

When OUTMODE (TIMERx_PWMCTL[16]) bit is set to 0, PWM output operates in independent mode. In this mode, both PWMx_CH0 and PWMx_CH1 can output the same waveform as shown in Figure 6.10-24.

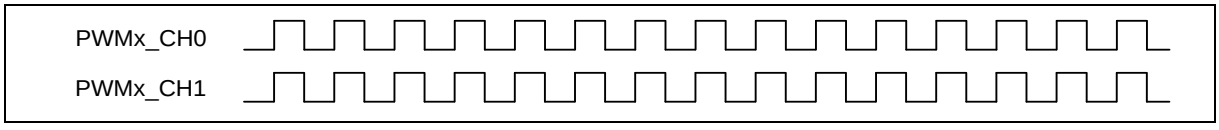


Figure 6.10-24 PWM Independent Mode Output Waveform

6.10.6.13 Complementary Mode

When OUTMODE (TIMERx_PWMCTL[16]) bit is set to 1, PWM output operates in complementary mode. In this mode, both PWMx_CH0 and PWMx_CH1 can output waveform and PWMx_CH1 must always be the complement of PWMx_CH0 as shown in Figure 6.10-25.

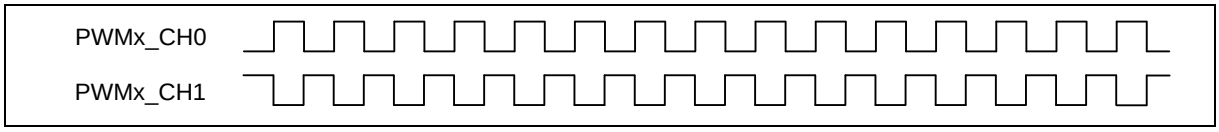


Figure 6.10-25 PWM Complementary Mode Output Waveform

6.10.6.14 PWM Output Control

After PWM pulse generator, there are four steps to control output waveform in independent output mode and five control steps in complementary output mode. User can set POEN0 (TIMERx_PWMPOEN[0]) and POEN1 (TIMERx_PWMPOEN[1]) 1 to enable PWMx_CH0 and PWMx_CH1 output waveform.

In Independent mode, there are mask control, brake control, polarity control and output enable control to control output waveform as shown in Figure 6.10-26.

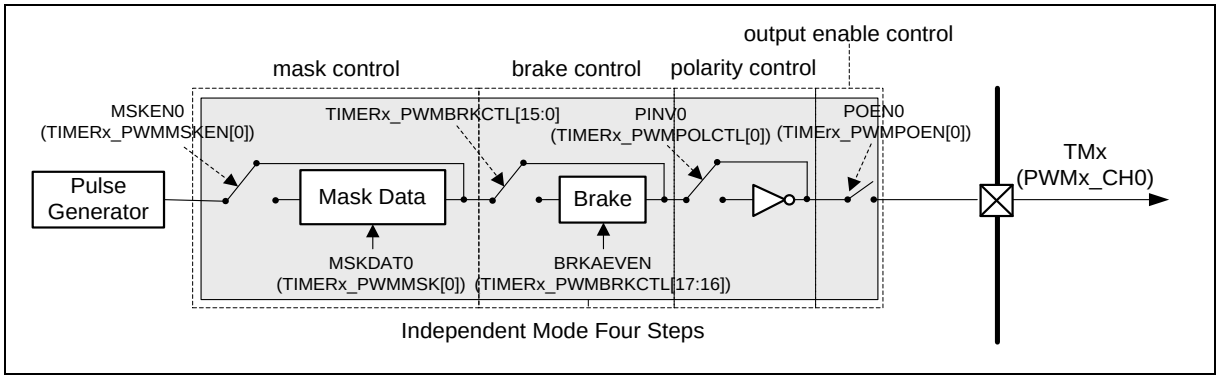


Figure 6.10-26 PWMx_CH0 Output Control in Independent Mode

In complementary mode, there are dead-time insertion control and four control steps the same as independent mode to control PWMx_CH0 and PWMx_CH1 outputs as shown in Figure 6.10-27.

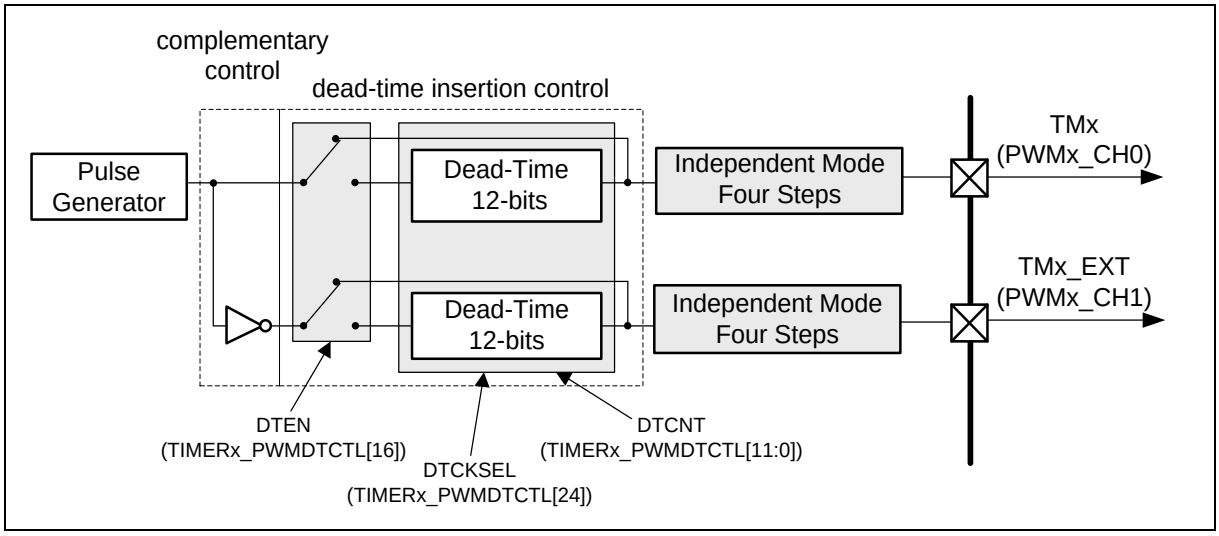


Figure 6.10-27 PWMx_CH0 and PWMx_CH1 Output Control in Complementary Mode

6.10.6.15 Dead-Time Insertion Control

In the complementary application, the complement channels may drive the external devices like power switches. The dead-time generator inserts a low level interval between complementary outputs PWMx_CH0 and PWMx_CH1 as shown in Figure 6.10-28. User sets DTEN (TIMERx_PWMDTCTL[16]) bit to enable dead-time control function, DTCNT (TIMERx_PWMDTCTL[11:0]) and DTCKSEL (TIMERx_PWMDTCTL[24]) to control dead-time interval. The dead-time interval can be calculated from the following formula:

Dead-time interval = (DTCNT + 1) * TMRx_PWMCLK period, if DTCKSEL is 0

Dead-time interval = (DTCNT + 1) * TMRx_PWMCLK * (CLKPSC + 1) period, if DTCKSEL is 1

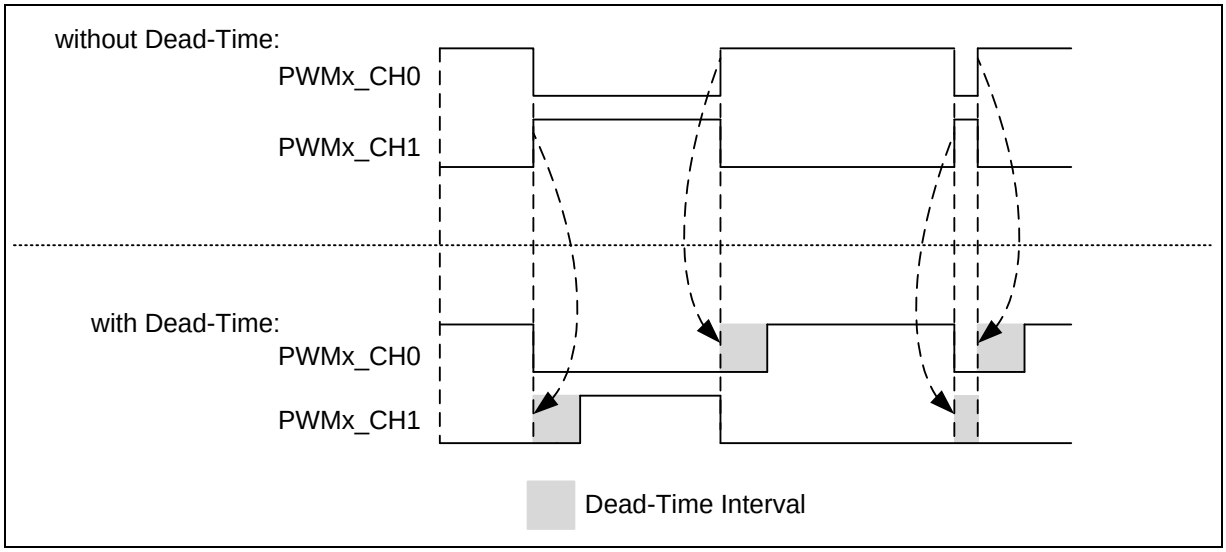


Figure 6.10-28 Dead-Time Insertion

6.10.6.16 PWM Mask Output Control

PWMx_CH0/CH1 output value can be masked to specified logic states by setting MSKEN0/1 (TIMERx_PWMMSKEN[1:0]) and MSKDAT0/1 (TIMERx_PWMMSK[1:0]). The PWM output mask function is useful when controlling various types of Electrically Commutated Motor (ECM) like a BLDC motor. Figure 6.10-29 shows an example of PWM output mask control in PWMx_CH0 and PWMx_CH1.

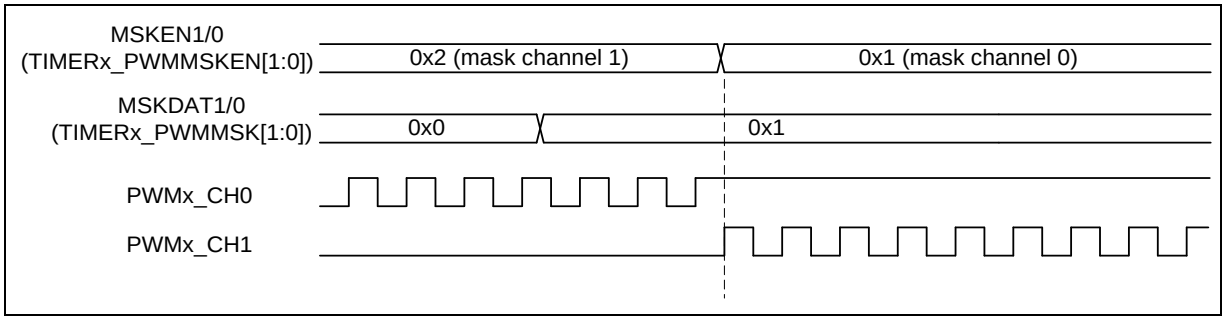


Figure 6.10-29 PWM Output Mask Control Waveform

6.10.6.17 PWM Brake Control

Each PWM generator supports one external input brake pin as PWM brake event source. User can select active brake pin source in BKPINSRC (TIMERx_PWMBNF[17:16]), PWMx_BRAKEy (x=0,1 and y=0,1). There is a 3-bit noise filter counter to filter the external brake pin signal. User can enable BRKNFEN (TIMERx_PWMBNF[0]) to enable the brake pin noise filter function and the noise filter sampling clock can be selected by setting BRKNFSEL (TIMERx_PWMBNF[3:1]) to fit different noise

properties. Moreover, by setting BRKFCNT (TIMERx_PWMBNF[6:4]), user can define by how many sampling clock cycles a filter will recognize the effective edge of the brake pin signal. In addition, brake pin polar can be inversed by setting BRKPINV (TIMERx_PWMBNF[7]) to realize the polarity setup for the brake control signals. Set BRKPINV to 0, brake event will occur when PWMx_BRAKEy pin status from low to high; set BRKPINV to 1, brake event will occurred when PWMx_BRAKEy pin status from high to low.

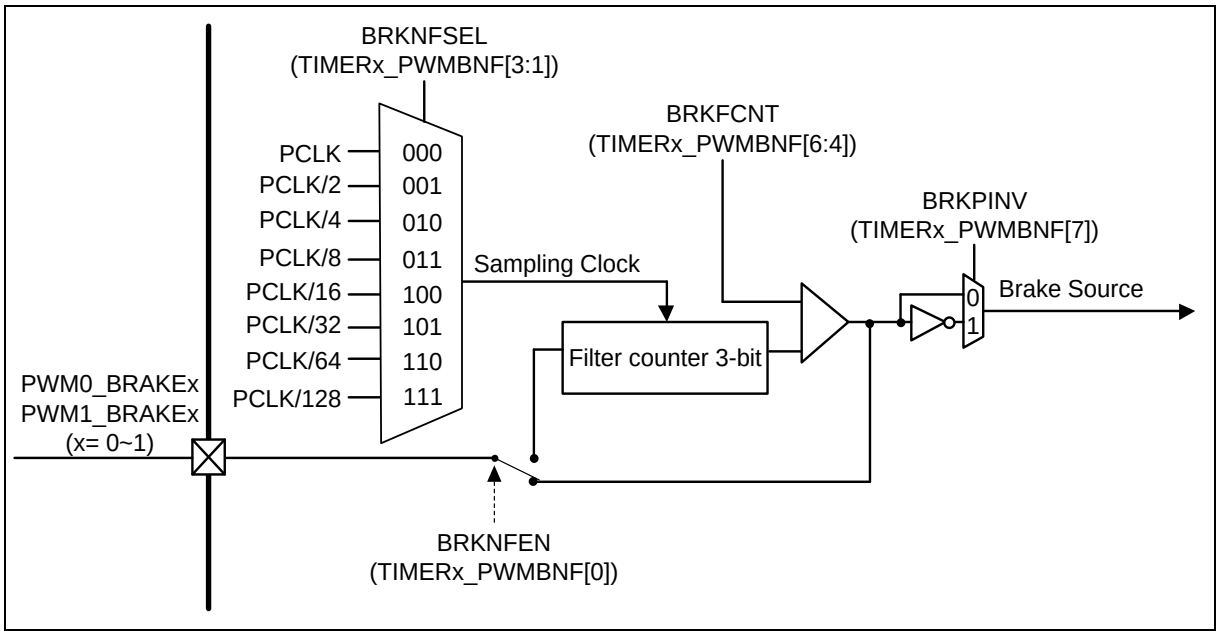


Figure 6.10-30 Brake Pin Noise Filter Block Diagram

User can set BRKAEVEN (TIMERx_PWMBRKCTL[17:16]) for PWMx_CH0 output state and BRKAODD (TIMERx_PWMBRKCTL[19:18]) for PWMx_CH1 output state when PWM brake event happened. There are two brake detector sources, edge detect brake source and level detect brake source when brake event happened. Figure 6.10-31 shows the brake event block diagram for PWMx_CH0 and PWMx_CH1.

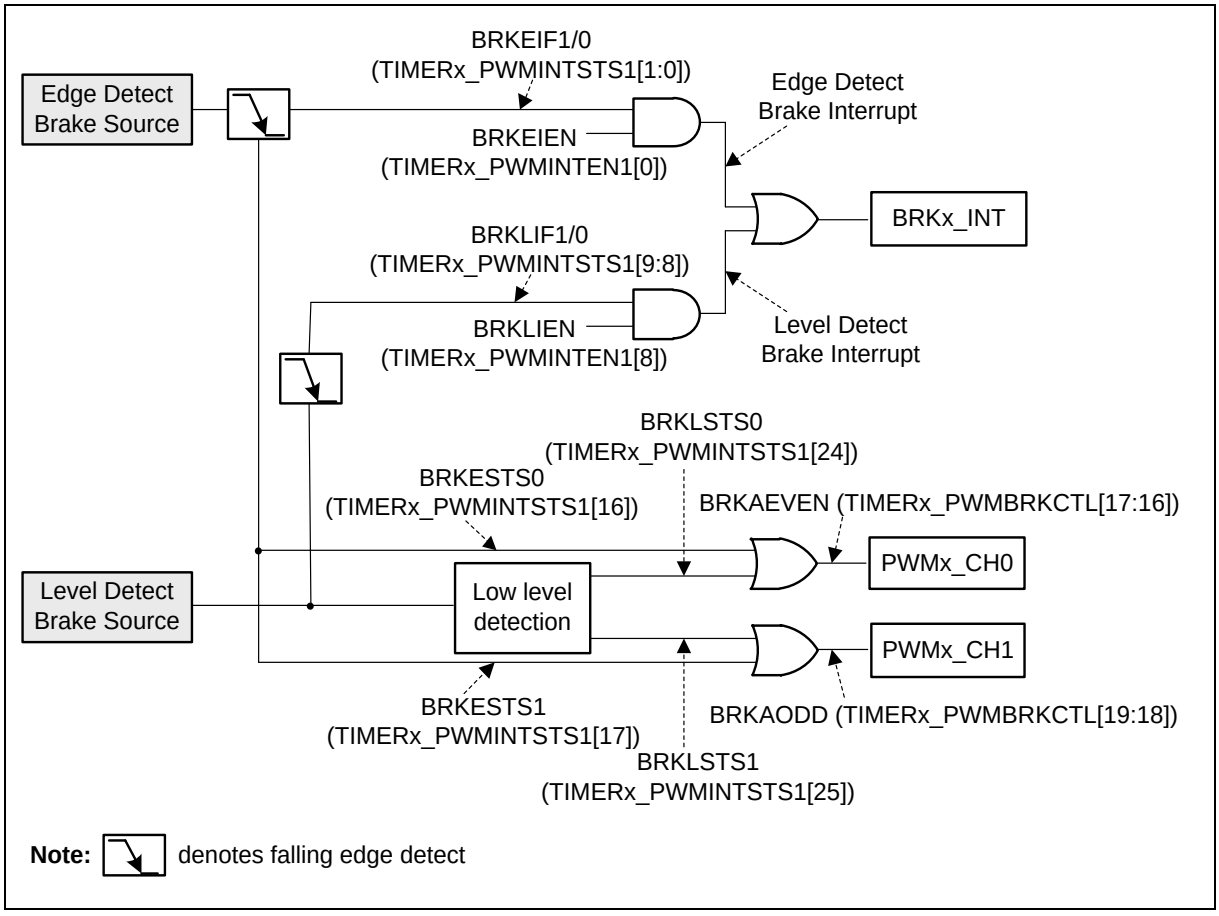
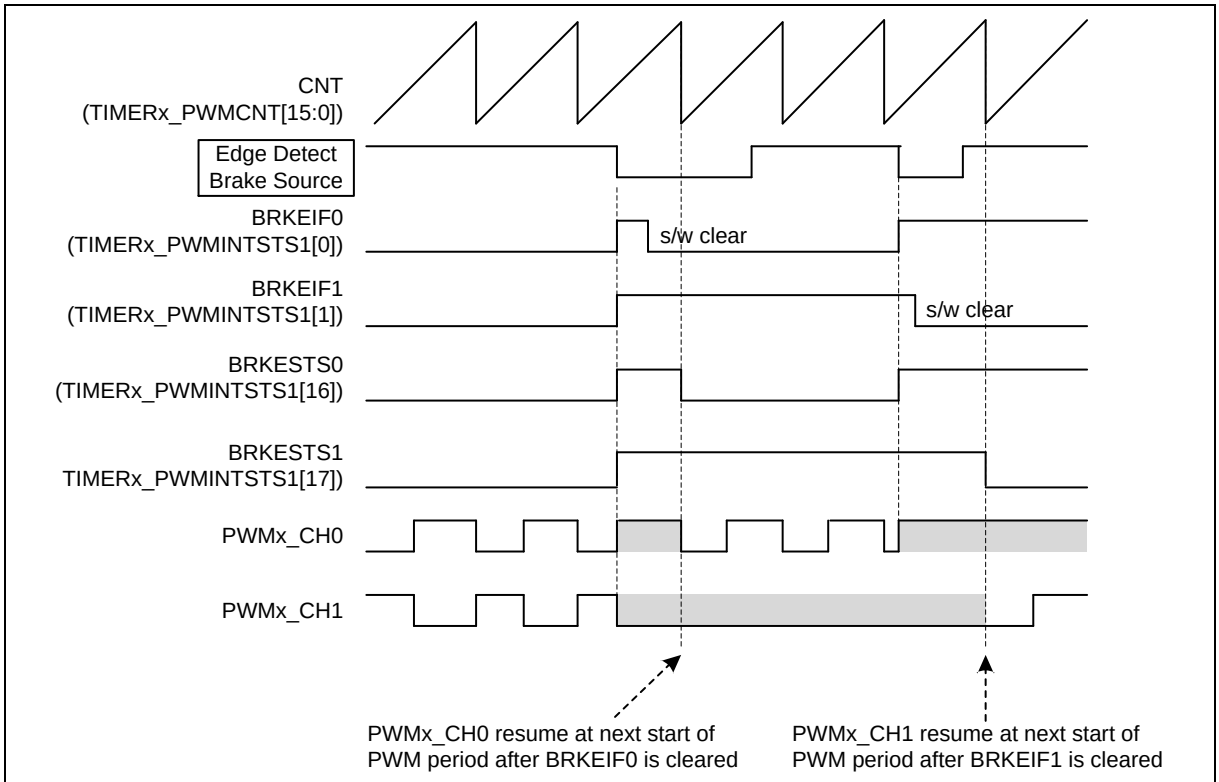


Figure 6.10-31 Brake Event Block Diagram for PWMx_CH0 and PWMx_CH1

When the edge detector detects the brake signal, the brake function generates interrupt status for PWMx_CH1/0 is BRKEIF1/0 (TIMERx_PWMINTSTS1[1:0]) and brake event status for PWMx_CH1/0 is BRKESTS1/0 (TIMERx_PWMINTSTS1[17:16]). The interrupt status BRKEIF1/0 can be cleared by writing 1 to it, and the brake event status BRKESTS1/0 will keep until the next PWM period starts when corresponding BRKEIF1/0 flag has been cleared and PWM generator can resume normal output.

Figure 6.10-32 shows an example of edge detector brake waveform for PWMx_CH0 and PWMx_CH1. In this case, the edge detect brake source has occurred twice for the brake events. When the first brake event occurs, both of the BRKEIF0 and BRKEIF1 flags are set and BRKESTS0 and BRKESTS1 status are also set to indicate brake state of PWMx_CH0 and PWMx_CH1. For the first occurring event, user writes 1 to clear the BRKEIF0. After that, the BRKESTS0 is cleared by hardware at the next start of the PWM period and the PWMx_CH0 outputs the normal waveform even though the edge brake event is still occurring. At the same time, BRKESTS1 keeps 1 and PWMx_CH1 keeps outputting low in brake state. The second event also triggers the same flags, but at this time, user writes 1 to clear the BRKEIF1. Afterward, PWMx_CH1 outputs normally at the next start of the PWM period.

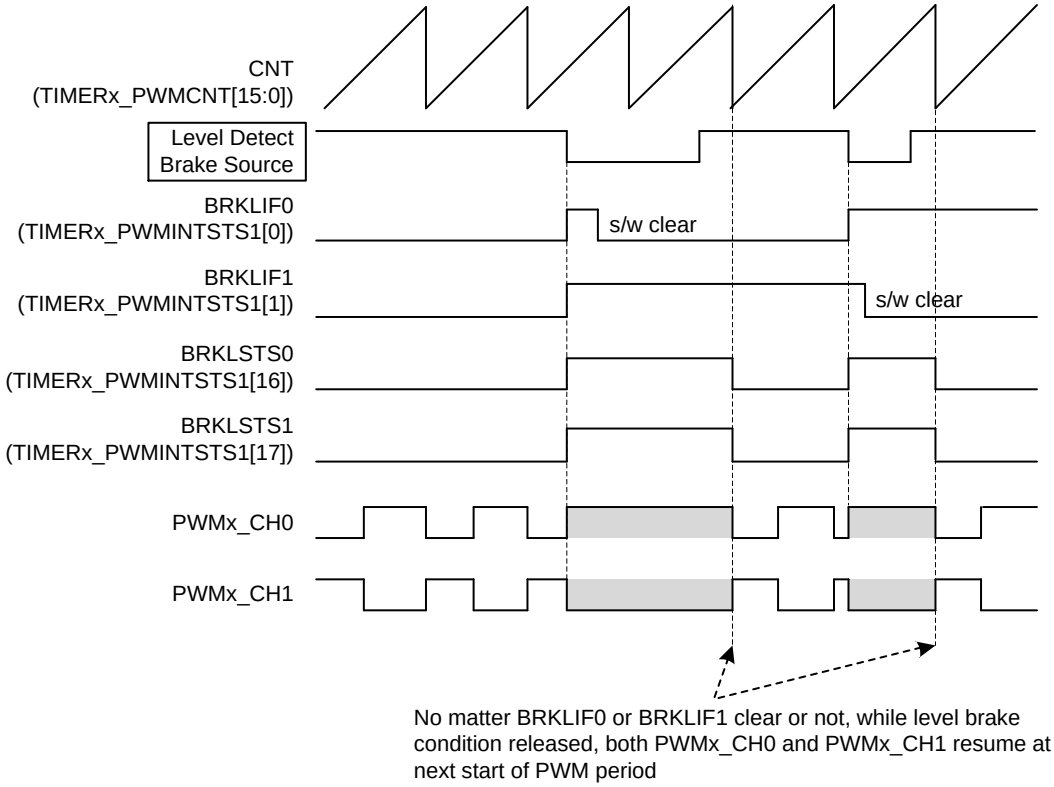


Note1: BRKAEVEN: 0x11, output high at brake condition
BRKAODD : 0x10, output low at brake condition
Note2: Output Brake State

Figure 6.10-32 Edge Detector Brake Waveform for PWMx_CH0 and PWMx_CH1

When the level detector detects the brake signal, the brake function generates interrupt status for PWMx_CH1/0 is BRKLIF1/0 (TIMERx_PWMINTSTS1[9:8]) and brake event status for PWMx_CH1/0 is BRKLSTS1/0 (TIMERx_PWMINTSTS1[25:24]). The interrupt status BRKLIF1/0 can be cleared by writing 1 to it, and the brake event status BRKLSTS1/0 will be cleared only when current period is completed and brake condition removed, then PWM generator can resume normal output when next PWM period starts.

Figure 6.10-33 shows an example of level detector brake waveform for PWMx_CH0 and PWMx_CH1. In this case, the BRKLIF0 and BRKLIF1 can only indicate the brake event has occurred, writes 1 to clear this flags will not affect BRKLSTS0 and BRKLSTS1 brake event status. Both BRKLSTS0 and BRKLSTS1 brake states will automatically cleared at the start of the next PWM period when level brake condition has released no matter BRKLIF0 and BRKLIF1 status.



Note1: BRKAEVEN: 0x11, output high at brake condition
BRKAODD : 0x10, output low at brake condition
Note2: Output Brake State

Figure 6.10-33 Level Detector Brake Waveform for PWMx_CH0 and PWMx_CH1

The two kinds of detectors detecting the same five brake sources as shown in Figure 6.10-34 are one from PWMx_BRAKEy (x=0,1 and y=0,1) external input signals, two from internal ACMP comparator signals, one from system fail events and one from software trigger brake event. ACMP brake sources will be detected only when internal ACMP0_O or ACMP1_O signal from low to high.

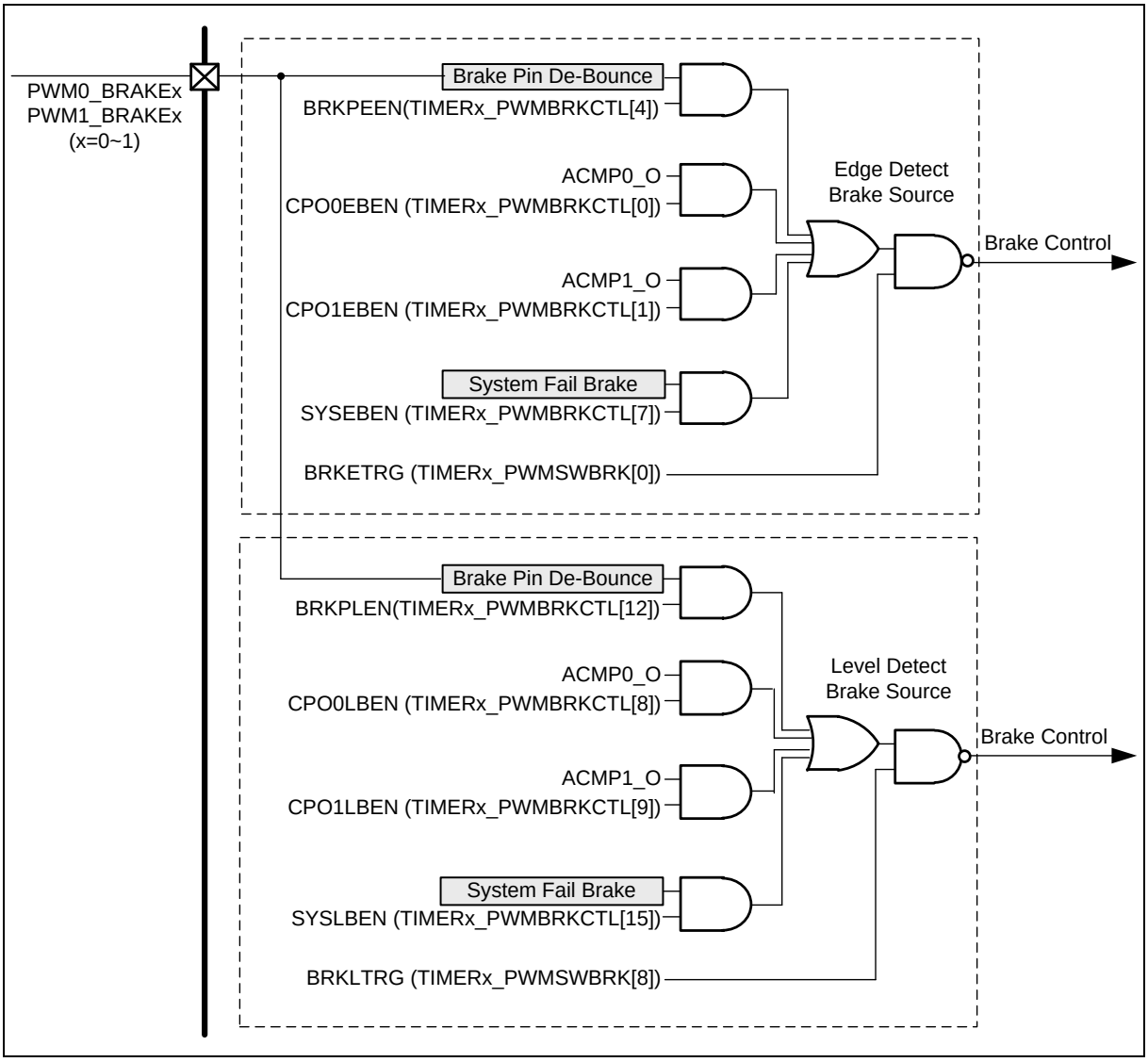


Figure 6.10-34 Brake Source Block Diagram

Among the above described brake sources, the brake source coming from system fail event can be specified to one of the different system fail conditions, these conditions include clock fail, BOD detect, SRAM parity error and CPU lockup as shown in Figure 6.10-35.

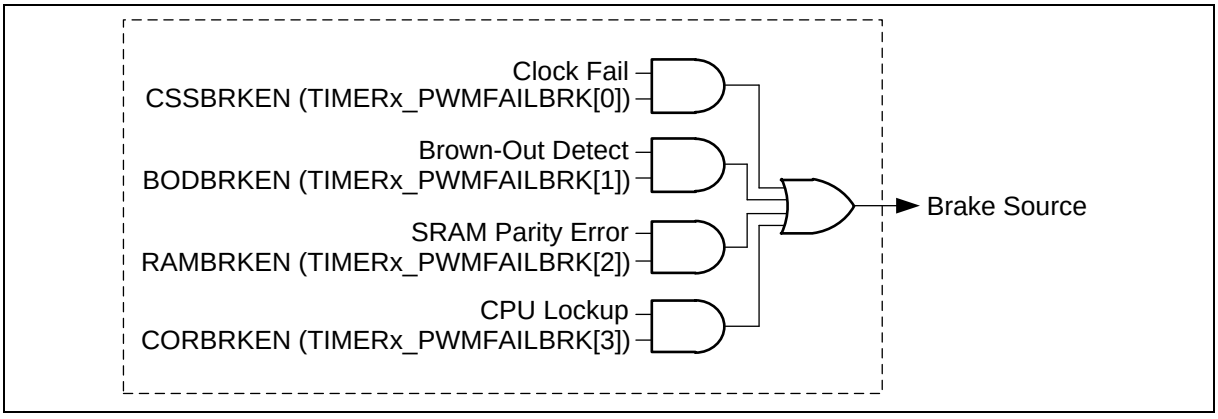


Figure 6.10-35 System Fail Brake Block Diagram

6.10.6.18 Polarity Control

Each PWMx_CH0 and PWMx_CH1 has an independent polarity control to configure the polarity of the active state of PWM output. User can control polarity state of PWMx_CH0 on PINV0 (TIMERx_PWMPOLCTL[0]) and PWMx_CH1 on PINV1 (TIMERx_PWMPOLCTL[1]). Figure 6.10-36 shows the PWMx_CH0 and PWMx_CH1 output with polarity control and dead-time insertion.

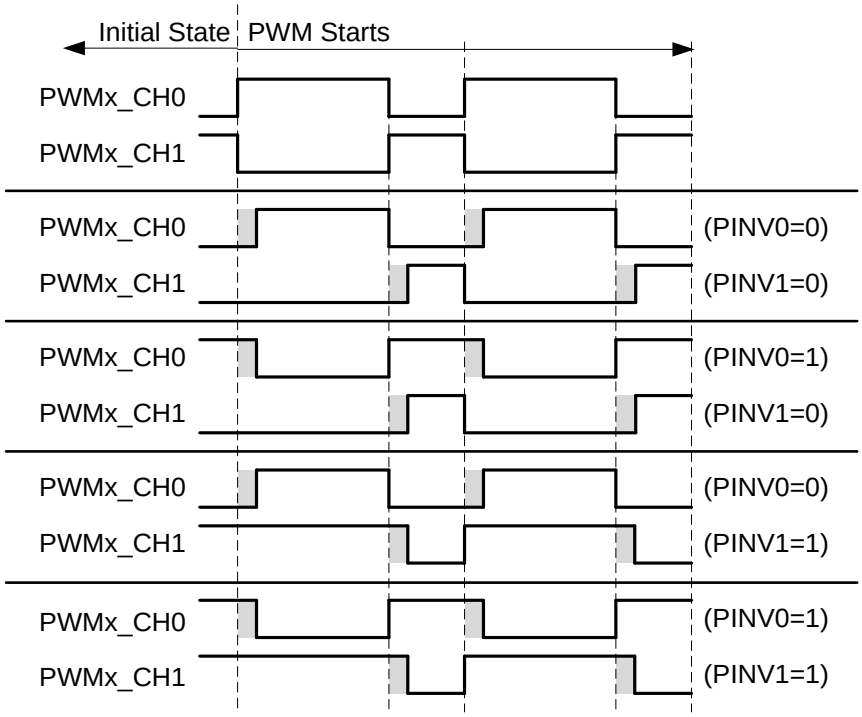


Figure 6.10-36 PWMx_CH0 and PWMx_CH1 Polarity Control with Dead-Time Insertion

6.10.6.19 PWM Interrupt Generator

There are independent interrupts for each PWM as shown in Figure 6.10-38.

The PWM interrupt (PWMx_INT) comes from PWM complementary pair events. The counter can generate the zero point interrupt flag ZIF (TIMERx_PWMINTSTS0[0]) and the period point interrupt flag PIF (TIMERx_PWMINTSTS0[1]). When counter equals to the comparator value stored in CMP (TIMERx_PWMCMPDAT[15:0]), the different interrupt flags will be triggered depending on the counting direction. If counter and CMP matched occurs at up-count direction, the comparator up interrupt flag CMPUIF (TIMERx_PWMINTSTS0[2]) is set and if matched at down-count direction, the comparator down interrupt flag CMPDIF (TIMERx_PWMINTSTS0[3]) is set. If the corresponding interrupt enable bits are set, the interrupt trigger events will also generate interrupt signals. When PWM brake event occurred, the relatives interrupt event will be triggered according to PWM brake settings.

PWMx_INT can use the TIMERx_PWMIFA register to accumulate the number of times that the interrupt flags have been triggered for each channel. By setting IFAEN (TIMERx_PWMIFA[31]) bit to 1 to enable accumulator, PWMx_INT will switch interrupt source from every event trigger interrupt to trigger interrupt once every accumulate times.

By setting the IFASEL (TIMERx_PWMIFA[29:28]) bits, user can select one of the 4 interrupt sources to accumulate interrupt flag times for each channel, and the number of times interrupt flags will compare

with IFACNT (TIMERx_PWMIFA[15:0]) bits. When interrupt accumulator equals IFACNT+1, the IFAIF (TIMERx_PWMAINTSTS[0]) bit will be set as PWMx_INT signal if user enables IFAIEN (TIMERx_PWMAINTEN[0]) bit. Accumulator interrupt of each channel can also be as request source of PDMA. Figure 6.10-38 is an example using TIMERx_PWMIFA register to output PWMx_INT once every IFCNT0+1 times interrupt events occurred.

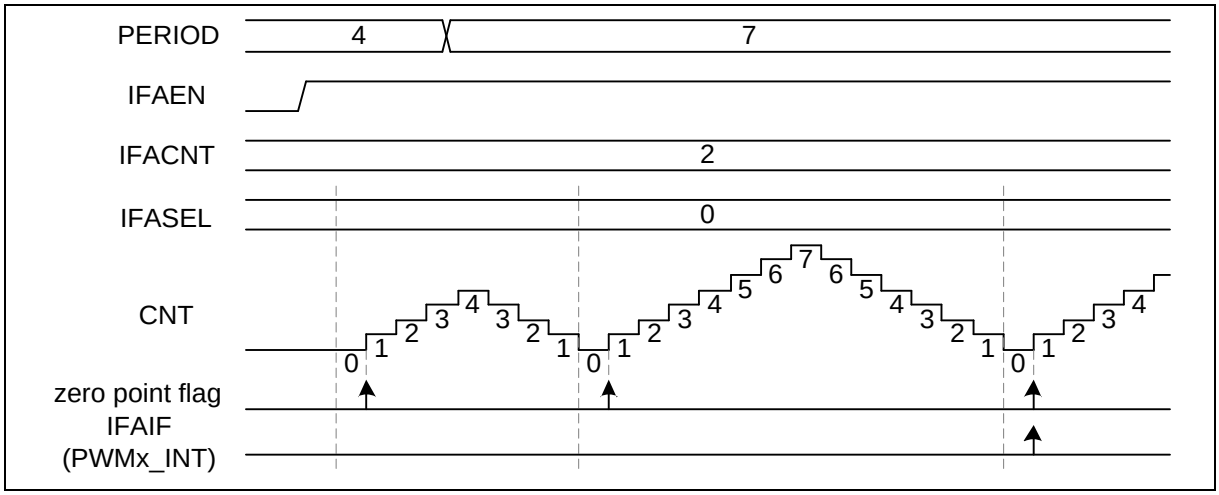


Figure 6.10-37 PWM Accumulate Interrupt Waveform

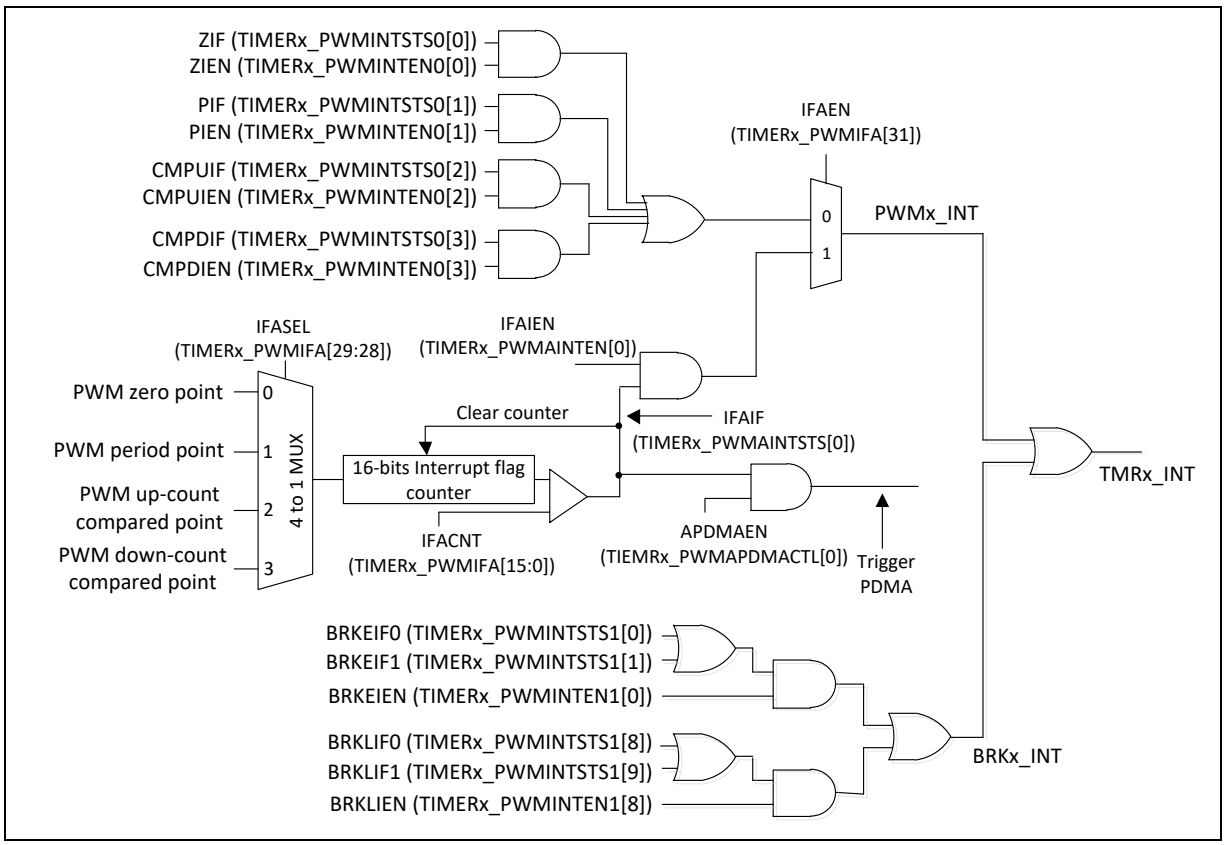


Figure 6.10-38 PWM Interrupt Architecture Diagram

6.10.6.20 Accumulator PDMA Function

The PWM module supports to trigger the PDMA transfer function when accumulator interrupt happened.

When the corresponding PDMA enable bit APDMAEN (TIMERx_PWMAPDMACTL[0]) is set, accumulator module will send a request to PDMA controller when accumulator interrupt has happened, meaning that IFAIF (TIMERx_PWMAINTSTS[0]) is set 1. The PDMA controller will issue an acknowledge to accumulator and auto clear IFAIF (TIMERx_PWMAINTSTS[0]) after it has read memory data and send the data to the particular register (PERIOD, etc.). So, user can use this function to change accumulator interrupt frequency.

6.10.6.21 Accumulator Stop Mode

The PWM module supports the Accumulator Stop Mode to stop counting when accumulator interrupt happened. Figure 6.10-39 shows Accumulator Stop Mode waveform. When the corresponding STPMOD (TIMERx_PWMIFA[24]) is set, accumulator module will disable CNTEN (TIMERx_PWMCTL[0]) after counter finishes whole period.

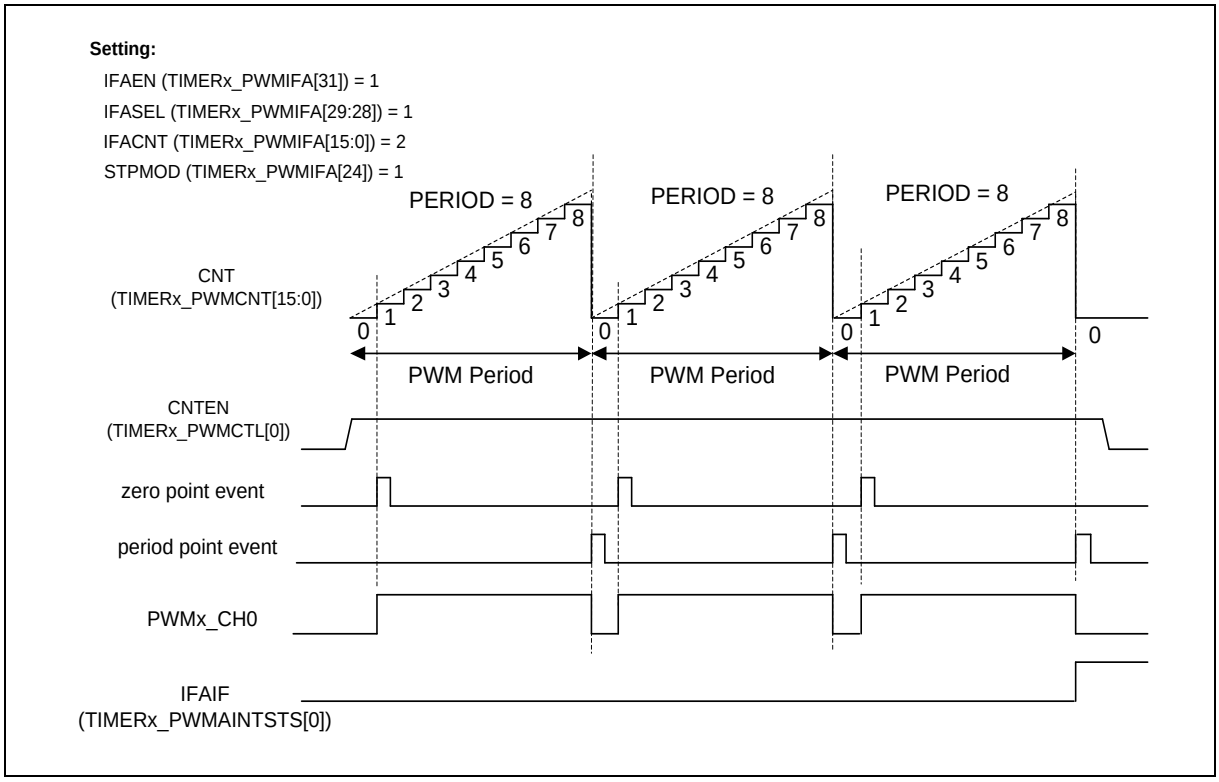


Figure 6.10-39 PWM Accumulator Stop Mode Waveform

6.10.6.22 PWM Trigger EADC Generator

PWM counter event can be one of the EADC conversion trigger source. User sets TRGSEL (TIMERx_PWMTRGCTL[2:0]) to select which PWM counter event can trigger EADC conversion after TRGEADC (TIMERx_PWMTRGCTL[7]) is enabled.

There are five PWM counter events that can be selected as the trigger source to start EADC conversion as shown in Figure 6.10-40.

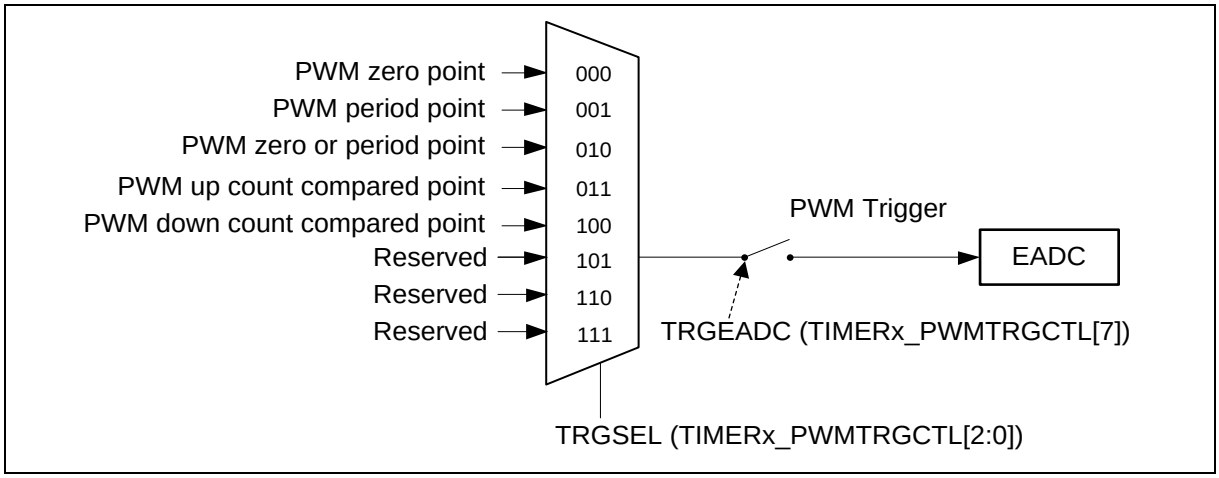


Figure 6.10-40 PWM Trigger EADC Block Diagram

6.10.6.23 External Pin Event Trigger

External pin event can trigger PWM counter action. Setting EXTETEN (TIMERx_PWMEXTETCTL[0]) bit to 1 can enable this function. Setting CNTACTS (TIMERx_PWMEXTETCTL[5:4]) can choose counter action: reset, start or reset and start. Setting EXTTRGS (TIMERx_PWMEXTETCTL[11:8]) can select external trigger source from INT0 ~ INT7.

Trigger event comes from detecting INT0 ~ INT7 rising, falling or both of rising and falling. More detailed setting is in GPIO chapter.

6.10.7 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
TIMER Base Address: **TMR01_BA = 0x4005_0000** **TMR23_BA = 0x4005_1000**				
TIMER0_CTL	TMR01_BA+0x00	R/W	Timer0 Control Register	0x0000_0005
TIMER0_CMP	TMR01_BA+0x04	R/W	Timer0 Comparator Register	0x0000_0000
TIMER0_INTSTS	TMR01_BA+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TIMER0_CNT	TMR01_BA+0x0C	R/W	Timer0 Data Register	0x0000_0000
TIMER0_CAP	TMR01_BA+0x10	R	Timer0 Capture Data Register	0x0000_0000
TIMER0_EXTCTL	TMR01_BA+0x14	R/W	Timer0 External Control Register	0x0000_0000
TIMER0_EINTSTS	TMR01_BA+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TIMER0_TRGCTL	TMR01_BA+0x1C	R/W	Timer0 Trigger Control Register	0x0000_0000
TIMER0_ALTCTL	TMR01_BA+0x20	R/W	Timer0 Alternative Control Register	0x0000_0000
TIMER0_CAPNF	TMR01_BA+0x24	R/W	Timer0 Capture Input Noise Filter Register	0x0000_0000
TIMER0_PWMCTL	TMR01_BA+0x40	R/W	Timer0 PWM Control Register	0x0000_0000
TIMER0_PWMCLKSRC	TMR01_BA+0x44	R/W	Timer0 PWM Counter Clock Source Register	0x0000_0000
TIMER0_PWMCLKPSC	TMR01_BA+0x48	R/W	Timer0 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER0_PWMCNTCLR	TMR01_BA+0x4C	R/W	Timer0 PWM Clear Counter Register	0x0000_0000
TIMER0_PWMPERIOD	TMR01_BA+0x50	R/W	Timer0 PWM Period Register	0x0000_0000
TIMER0_PWMCMPDAT	TMR01_BA+0x54	R/W	Timer0 PWM Comparator Register	0x0000_0000
TIMER0_PWMDTCTL	TMR01_BA+0x58	R/W	Timer0 PWM Dead-Time Control Register	0x0000_0000
TIMER0_PWMCNT	TMR01_BA+0x5C	R	Timer0 PWM Counter Register	0x0000_0000
TIMER0_PWMMSKEN	TMR01_BA+0x60	R/W	Timer0 PWM Output Mask Enable Register	0x0000_0000
TIMER0_PWMMSK	TMR01_BA+0x64	R/W	Timer0 PWM Output Mask Data Control Register	0x0000_0000
TIMER0_PWMBNF	TMR01_BA+0x68	R/W	Timer0 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER0_PWMFAILBRK	TMR01_BA+0x6C	R/W	Timer0 PWM System Fail Brake Control Register	0x0000_0000
TIMER0_PWMBRKCTL	TMR01_BA+0x70	R/W	Timer0 PWM Brake Control Register	0x0000_0000
TIMER0_PWMPOLCTL	TMR01_BA+0x74	R/W	Timer0 PWM Pin Output Polar Control Register	0x0000_0000
TIMER0_PWMPOEN	TMR01_BA+0x78	R/W	Timer0 PWM Pin Output Enable Register	0x0000_0000
TIMER0_PWMSWBRK	TMR01_BA+0x7C	W	Timer0 PWM Software Trigger Brake Control Register	0x0000_0000

TIMER0_PWMINTEN0	TMR01_BA+0x80	R/W	Timer0 PWM Interrupt Enable Register 0	0x0000_0000
TIMER0_PWMINTEN1	TMR01_BA+0x84	R/W	Timer0 PWM Interrupt Enable Register 1	0x0000_0000
TIMER0_PWMINTSTS0	TMR01_BA+0x88	R/W	Timer0 PWM Interrupt Status Register 0	0x0000_0000
TIMER0_PWMINTSTS1	TMR01_BA+0x8C	R/W	Timer0 PWM Interrupt Status Register 1	0x0000_0000
TIMER0_PWMTRGCTL	TMR01_BA+0x90	R/W	Timer0 PWM Trigger Control Register	0x0000_0000
TIMER0_PWMSCTL	TMR01_BA+0x94	R/W	Timer0 PWM Synchronous Control Register	0x0000_0000
TIMER0_PWMSTRG	TMR01_BA+0x98	W	Timer0 PWM Synchronous Trigger Register	0x0000_0000
TIMER0_PWMSTATUS	TMR01_BA+0x9C	R/W	Timer0 PWM Status Register	0x0000_0000
TIMER0_PWMPBUF	TMR01_BA+0xA0	R	Timer0 PWM Period Buffer Register	0x0000_0000
TIMER0_PWMCMPBUF	TMR01_BA+0xA4	R	Timer0 PWM Comparator Buffer Register	0x0000_0000
TIMER0_PWMIFA	TMR01_BA+0xA8	R/W	Timer0 PWM Interrupt Flag Accumulator Register	0x0000_0000
TIMER0_PWMAINTSTS	TMR01_BA+0xAC	R/W	Timer0 PWM Accumulator Interrupt Flag Register	0x0000_0000
TIMER0_PWMAINTEN	TMR01_BA+0xB0	R/W	Timer0 PWM Accumulator Interrupt Enable Register	0x0000_0000
TIMER0_PWMAPDMACTL	TMR01_BA+0xB4	R/W	Timer0 PWM Accumulator PDMA Control Register	0x0000_0000
TIMER0_PWMEXTETCTL	TMR01_BA+0xB8	R/W	Timer0 PWM External Event Trigger Control Register	0x0000_0000
TIMER1_CTL	TMR01_BA+0x100	R/W	Timer1 Control Register	0x0000_0005
TIMER1_CMP	TMR01_BA+0x104	R/W	Timer1 Comparator Register	0x0000_0000
TIMER1_INTSTS	TMR01_BA+0x108	R/W	Timer1 Interrupt Status Register	0x0000_0000
TIMER1_CNT	TMR01_BA+0x10C	R/W	Timer1 Data Register	0x0000_0000
TIMER1_CAP	TMR01_BA+0x110	R	Timer1 Capture Data Register	0x0000_0000
TIMER1_EXTCTL	TMR01_BA+0x114	R/W	Timer1 External Control Register	0x0000_0000
TIMER1_EINTSTS	TMR01_BA+0x118	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TIMER1_TRGCTL	TMR01_BA+0x11C	R/W	Timer1 Trigger Control Register	0x0000_0000
TIMER1_ALTCTL	TMR01_BA+0x120	R/W	Timer1 Alternative Control Register	0x0000_0000
TIMER1_CAPNF	TMR01_BA+0x124	R/W	Timer1 Capture Input Noise Filter Register	0x0000_0000
TIMER1_PWMCTL	TMR01_BA+0x140	R/W	Timer1 PWM Control Register	0x0000_0000
TIMER1_PWMCLKSRC	TMR01_BA+0x144	R/W	Timer1 PWM Counter Clock Source Register	0x0000_0000
TIMER1_PWMCLKPSC	TMR01_BA+0x148	R/W	Timer1 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER1_PWMCNTCLR	TMR01_BA+0x14C	R/W	Timer1 PWM Clear Counter Register	0x0000_0000
TIMER1_PWMPERIOD	TMR01_BA+0x150	R/W	Timer1 PWM Period Register	0x0000_0000
TIMER1_PWMCMPDAT	TMR01_BA+0x154	R/W	Timer1 PWM Comparator Register	0x0000_0000

TIMER1_PWMDTCTL	TMR01_BA+0x158	R/W	Timer1 PWM Dead-Time Control Register	0x0000_0000
TIMER1_PWMCNT	TMR01_BA+0x15C	R	Timer1 PWM Counter Register	0x0000_0000
TIMER1_PWMMSKEN	TMR01_BA+0x160	R/W	Timer1 PWM Output Mask Enable Register	0x0000_0000
TIMER1_PWMMSK	TMR01_BA+0x164	R/W	Timer1 PWM Output Mask Data Control Register	0x0000_0000
TIMER1_PWMBNF	TMR01_BA+0x168	R/W	Timer1 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER1_PWMFAILBRK	TMR01_BA+0x16C	R/W	Timer1 PWM System Fail Brake Control Register	0x0000_0000
TIMER1_PWMBRKCTL	TMR01_BA+0x170	R/W	Timer1 PWM Brake Control Register	0x0000_0000
TIMER1_PWMPOLCTL	TMR01_BA+0x174	R/W	Timer1 PWM Pin Output Polar Control Register	0x0000_0000
TIMER1_PWMPOEN	TMR01_BA+0x178	R/W	Timer1 PWM Pin Output Enable Register	0x0000_0000
TIMER1_PWMSWBRK	TMR01_BA+0x17C	W	Timer1 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER1_PWMINTEN0	TMR01_BA+0x180	R/W	Timer1 PWM Interrupt Enable Register 0	0x0000_0000
TIMER1_PWMINTEN1	TMR01_BA+0x184	R/W	Timer1 PWM Interrupt Enable Register 1	0x0000_0000
TIMER1_PWMINTSTS0	TMR01_BA+0x188	R/W	Timer1 PWM Interrupt Status Register 0	0x0000_0000
TIMER1_PWMINTSTS1	TMR01_BA+0x18C	R/W	Timer1 PWM Interrupt Status Register 1	0x0000_0000
TIMER1_PWMTRGCTL	TMR01_BA+0x190	R/W	Timer1 PWM Trigger Control Register	0x0000_0000
TIMER1_PWMSCTL	TMR01_BA+0x194	R/W	Timer1 PWM Synchronous Control Register	0x0000_0000
TIMER1_PWMSTATUS	TMR01_BA+0x19C	R/W	Timer1 PWM Status Register	0x0000_0000
TIMER1_PWMPBUF	TMR01_BA+0x1A0	R	Timer1 PWM Period Buffer Register	0x0000_0000
TIMER1_PWMCMPBUF	TMR01_BA+0x1A4	R	Timer1 PWM Comparator Buffer Register	0x0000_0000
TIMER1_PWMIFA	TMR01_BA+0x1A8	R/W	Timer1 PWM Interrupt Flag Accumulator Register	0x0000_0000
TIMER1_PWMAINTSTS	TMR01_BA+0x1AC	R/W	Timer1 PWM Accumulator Interrupt Flag Register	0x0000_0000
TIMER1_PWMAINTEN	TMR01_BA+0x1B0	R/W	Timer1 PWM Accumulator Interrupt Enable Register	0x0000_0000
TIMER1_PWMAPDMACTL	TMR01_BA+0x1B4	R/W	Timer1 PWM Accumulator PDMA Control Register	0x0000_0000
TIMER1_PWMEXTETCTL	TMR01_BA+0x1B8	R/W	Timer1 PWM External Event Trigger Control Register	0x0000_0000
TIMER2_CTL	TMR23_BA+0x00	R/W	Timer2 Control Register	0x0000_0005
TIMER2_CMP	TMR23_BA+0x04	R/W	Timer2 Comparator Register	0x0000_0000
TIMER2_INTSTS	TMR23_BA+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TIMER2_CNT	TMR23_BA+0x0C	R/W	Timer2 Data Register	0x0000_0000
TIMER2_CAP	TMR23_BA+0x10	R	Timer2 Capture Data Register	0x0000_0000
TIMER2_EXTCTL	TMR23_BA+0x14	R/W	Timer2 External Control Register	0x0000_0000
TIMER2_EINTSTS	TMR23_BA+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000

TIMER2_TRGCTL	TMR23_BA+0x1C	R/W	Timer2 Trigger Control Register	0x0000_0000
TIMER2_ALTCTL	TMR23_BA+0x20	R/W	Timer2 Alternative Control Register	0x0000_0000
TIMER2_CAPNF	TMR23_BA+0x24	R/W	Timer2 Capture Input Noise Filter Register	0x0000_0000
TIMER2_PWMCTL	TMR23_BA+0x40	R/W	Timer2 PWM Control Register	0x0000_0000
TIMER2_PWMCLKSRC	TMR23_BA+0x44	R/W	Timer2 PWM Counter Clock Source Register	0x0000_0000
TIMER2_PWMCLKPSC	TMR23_BA+0x48	R/W	Timer2 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER2_PWMCNTCLR	TMR23_BA+0x4C	R/W	Timer2 PWM Clear Counter Register	0x0000_0000
TIMER2_PWMPERIOD	TMR23_BA+0x50	R/W	Timer2 PWM Period Register	0x0000_0000
TIMER2_PWMCMPDAT	TMR23_BA+0x54	R/W	Timer2 PWM Comparator Register	0x0000_0000
TIMER2_PWMDTCTL	TMR23_BA+0x58	R/W	Timer2 PWM Dead-Time Control Register	0x0000_0000
TIMER2_PWMCNT	TMR23_BA+0x5C	R	Timer2 PWM Counter Register	0x0000_0000
TIMER2_PWMMSKEN	TMR23_BA+0x60	R/W	Timer2 PWM Output Mask Enable Register	0x0000_0000
TIMER2_PWMMSK	TMR23_BA+0x64	R/W	Timer2 PWM Output Mask Data Control Register	0x0000_0000
TIMER2_PWMBNF	TMR23_BA+0x68	R/W	Timer2 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER2_PWMFAILBRK	TMR23_BA+0x6C	R/W	Timer2 PWM System Fail Brake Control Register	0x0000_0000
TIMER2_PWMBRKCTL	TMR23_BA+0x70	R/W	Timer2 PWM Brake Control Register	0x0000_0000
TIMER2_PWMPOLCTL	TMR23_BA+0x74	R/W	Timer2 PWM Pin Output Polar Control Register	0x0000_0000
TIMER2_PWMPOEN	TMR23_BA+0x78	R/W	Timer2 PWM Pin Output Enable Register	0x0000_0000
TIMER2_PWMSWBRK	TMR23_BA+0x7C	W	Timer2 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER2_PWMINTEN0	TMR23_BA+0x80	R/W	Timer2 PWM Interrupt Enable Register 0	0x0000_0000
TIMER2_PWMINTEN1	TMR23_BA+0x84	R/W	Timer2 PWM Interrupt Enable Register 1	0x0000_0000
TIMER2_PWMINTSTS0	TMR23_BA+0x88	R/W	Timer2 PWM Interrupt Status Register 0	0x0000_0000
TIMER2_PWMINTSTS1	TMR23_BA+0x8C	R/W	Timer2 PWM Interrupt Status Register 1	0x0000_0000
TIMER2_PWMTRGCTL	TMR23_BA+0x90	R/W	Timer2 PWM Trigger Control Register	0x0000_0000
TIMER2_PWMSCTL	TMR23_BA+0x94	R/W	Timer2 PWM Synchronous Control Register	0x0000_0000
TIMER2_PWMSTRG	TMR23_BA+0x98	W	Timer2 PWM Synchronous Trigger Register	0x0000_0000
TIMER2_PWMSTATUS	TMR23_BA+0x9C	R/W	Timer2 PWM Status Register	0x0000_0000
TIMER2_PWMPBUF	TMR23_BA+0xA0	R	Timer2 PWM Period Buffer Register	0x0000_0000
TIMER2_PWMCMPBUF	TMR23_BA+0xA4	R	Timer2 PWM Comparator Buffer Register	0x0000_0000
TIMER2_PWMIFA	TMR23_BA+0xA8	R/W	Timer2 PWM Interrupt Flag Accumulator Register	0x0000_0000
TIMER2_PWMAINTSTS	TMR23_BA+0xAC	R/W	Timer2 PWM Accumulator Interrupt Flag Register	0x0000_0000

TIMER2_PWMAINTEN	TMR23_BA+0xB0	R/W	Timer2 PWM Accumulator Interrupt Enable Register	0x0000_0000
TIMER2_PWMAPDMACTL	TMR23_BA+0xB4	R/W	Timer2 PWM Accumulator PDMA Control Register	0x0000_0000
TIMER2_PWMEXTETCTL	TMR23_BA+0xB8	R/W	Timer2 PWM External Event Trigger Control Register	0x0000_0000
TIMER3_CTL	TMR23_BA+0x100	R/W	Timer3 Control Register	0x0000_0005
TIMER3_CMP	TMR23_BA+0x104	R/W	Timer3 Comparator Register	0x0000_0000
TIMER3_INTSTS	TMR23_BA+0x108	R/W	Timer3 Interrupt Status Register	0x0000_0000
TIMER3_CNT	TMR23_BA+0x10C	R/W	Timer3 Data Register	0x0000_0000
TIMER3_CAP	TMR23_BA+0x110	R	Timer3 Capture Data Register	0x0000_0000
TIMER3_EXTCTL	TMR23_BA+0x114	R/W	Timer3 External Control Register	0x0000_0000
TIMER3_EINTSTS	TMR23_BA+0x118	R/W	Timer3 External Interrupt Status Register	0x0000_0000
TIMER3_TRGCTL	TMR23_BA+0x11C	R/W	Timer3 Trigger Control Register	0x0000_0000
TIMER3_ALTCTL	TMR23_BA+0x120	R/W	Timer3 Alternative Control Register	0x0000_0000
TIMER3_CAPNF	TMR23_BA+0x124	R/W	Timer3 Capture Input Noise Filter Register	0x0000_0000
TIMER3_PWMCTL	TMR23_BA+0x140	R/W	Timer3 PWM Control Register	0x0000_0000
TIMER3_PWMCLKSRC	TMR23_BA+0x144	R/W	Timer3 PWM Counter Clock Source Register	0x0000_0000
TIMER3_PWMCLKPSC	TMR23_BA+0x148	R/W	Timer3 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER3_PWMCNTCLR	TMR23_BA+0x14C	R/W	Timer3 PWM Clear Counter Register	0x0000_0000
TIMER3_PWMPERIOD	TMR23_BA+0x150	R/W	Timer3 PWM Period Register	0x0000_0000
TIMER3_PWMCMPDAT	TMR23_BA+0x154	R/W	Timer3 PWM Comparator Register	0x0000_0000
TIMER3_PWMDTCTL	TMR23_BA+0x158	R/W	Timer3 PWM Dead-Time Control Register	0x0000_0000
TIMER3_PWMCNT	TMR23_BA+0x15C	R	Timer3 PWM Counter Register	0x0000_0000
TIMER3_PWMMSKEN	TMR23_BA+0x160	R/W	Timer3 PWM Output Mask Enable Register	0x0000_0000
TIMER3_PWMMSK	TMR23_BA+0x164	R/W	Timer3 PWM Output Mask Data Control Register	0x0000_0000
TIMER3_PWMBNF	TMR23_BA+0x168	R/W	Timer3 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER3_PWMFAILBRK	TMR23_BA+0x16C	R/W	Timer3 PWM System Fail Brake Control Register	0x0000_0000
TIMER3_PWMBRKCTL	TMR23_BA+0x170	R/W	Timer3 PWM Brake Control Register	0x0000_0000
TIMER3_PWMPOLCTL	TMR23_BA+0x174	R/W	Timer3 PWM Pin Output Polar Control Register	0x0000_0000
TIMER3_PWMPOEN	TMR23_BA+0x178	R/W	Timer3 PWM Pin Output Enable Register	0x0000_0000
TIMER3_PWMSWBRK	TMR23_BA+0x17C	W	Timer3 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER3_PWMINTEN0	TMR23_BA+0x180	R/W	Timer3 PWM Interrupt Enable Register 0	0x0000_0000
TIMER3_PWMINTEN1	TMR23_BA+0x184	R/W	Timer3 PWM Interrupt Enable Register 1	0x0000_0000

TIMER3_PWMINTSTS0	TMR23_BA+0x188	R/W	Timer3 PWM Interrupt Status Register 0	0x0000_0000
TIMER3_PWMINTSTS1	TMR23_BA+0x18C	R/W	Timer3 PWM Interrupt Status Register 1	0x0000_0000
TIMER3_PWMTRGCTL	TMR23_BA+0x190	R/W	Timer3 PWM Trigger Control Register	0x0000_0000
TIMER3_PWMSCTL	TMR23_BA+0x194	R/W	Timer3 PWM Synchronous Control Register	0x0000_0000
TIMER3_PWMSTATUS	TMR23_BA+0x19C	R/W	Timer3 PWM Status Register	0x0000_0000
TIMER3_PWMPBUF	TMR23_BA+0x1A0	R	Timer3 PWM Period Buffer Register	0x0000_0000
TIMER3_PWMCMPBUF	TMR23_BA+0x1A4	R	Timer3 PWM Comparator Buffer Register	0x0000_0000
TIMER3_PWMIFA	TMR23_BA+0x1A8	R/W	Timer3 PWM Interrupt Flag Accumulator Register	0x0000_0000
TIMER3_PWMAINTSTS	TMR23_BA+0x1AC	R/W	Timer3 PWM Accumulator Interrupt Flag Register	0x0000_0000
TIMER3_PWMAINTEN	TMR23_BA+0x1B0	R/W	Timer3 PWM Accumulator Interrupt Enable Register	0x0000_0000
TIMER3_PWMAPDMACTL	TMR23_BA+0x1B4	R/W	Timer3 PWM Accumulator PDMA Control Register	0x0000_0000
TIMER3_PWMEXTETCTL	TMR23_BA+0x1B8	R/W	Timer3 PWM External Event Trigger Control Register	0x0000_0000

6.10.8 Register Description

Timer Control Register (TIMERx_CTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_CTL	TMR01_BA+0x00	R/W	Timer0 Control Register	0x0000_0005
TIMER1_CTL	TMR01_BA+0x100	R/W	Timer1 Control Register	0x0000_0005
TIMER2_CTL	TMR23_BA+0x00	R/W	Timer2 Control Register	0x0000_0005
TIMER3_CTL	TMR23_BA+0x100	R/W	Timer3 Control Register	0x0000_0005

31	30	29	28	27	26	25	24
ICEDEBUG	CNTEN	INTEN	OPMODE		Reserved	ACTSTS	EXTCNTEN
23	22	21	20	19	18	17	16
WKEN	CAPSRC	TGPINSEL	PERIOSEL	INTRGEN	Reserved		
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PSC							

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement effects TIMER counting. TIMER counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. TIMER counter will keep going no matter CPU is held by ICE or not. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	**CNTEN**	**Timer Counting Enable Bit** 0 = Stops/Suspends counting. 1 = Starts counting. **Note 1:** In stop status, and then set CNTEN to 1 will enable the 24-bit up counter to keep counting from the last stop counting value. **Note 2:** This bit is auto-cleared by hardware in one-shot mode (TIMERx_CTL[28:27] = 00) when the timer time-out interrupt flag TIF (TIMERx_INTSTS[0]) is generated. **Note 3:** Set enable/disable this bit needs 2 * TMR_CLK period to become active, user can read ACTSTS (TIMERx_CTL[25]) to check enable/disable command is completed or not.
[29]	**INTEN**	**Timer Time-out Interrupt Enable Bit** 0 = Timer time-out interrupt Disabled. 1 = Timer time-out interrupt Enabled. **Note:** If this bit is enabled, when the timer time-out interrupt flag TIF is set to 1, the timer interrupt signal is generated and inform to CPU.
[28:27]	**OPMODE**	**Timer Counting Mode Select** 00 = The timer controller is operated in One-shot mode.

		01 = The timer controller is operated in Periodic mode. 10 = The timer controller is operated in Toggle-output mode. 11 = The timer controller is operated in Continuous Counting mode.
[26]	**Reserved**	Reserved.
[25]	**ACTSTS**	**Timer Active Status Bit (Read Only)** This bit indicates the 24-bit up counter status. 0 = 24-bit up counter is not active. 1 = 24-bit up counter is active. **Note:** This bit may active when CNT 0 transition to CNT 1.
[24]	**EXTCNTEN**	**Event Counter Mode Enable Bit** This bit is for event counting function enabled. 0 = Event counter mode Disabled. 1 = Event counter mode Enabled. **Note:** When timer is used as an event counter, this bit should be set to 1 and select PCLK as timer clock source.
[23]	**WKEN**	**Wake-up Function Enable Bit** If this bit is set to 1, while timer interrupt flag TIF (TIMERx_INTSTS[0]) is 1 and INTEN (TIMERx_CTL[29]) is enabled, the timer interrupt signal will generate a wake-up trigger event to CPU. 0 = Wake-up function Disabled if timer interrupt signal generated. 1 = Wake-up function Enabled if timer interrupt signal generated.
[22]	**CAPSRC**	**Capture Source Selection** 0 = Capture Function source is from TMx_EXT (x= 0~3) pin. 1 = Capture Function source is from internal ACMP output signal, internal clock source (HIRC, LIRC) or external clock (HXT, LXT). **Note:** When CAPSRC = 1, user can set ICAPSEL (TIMERx_EXTCTL[10:8]) to decide which internal ACMP output signal or which clock is as timer capture source.
[21]	**TGPINSEL**	**Toggle-output Pin Select** 0 = Toggle mode output to TMx (Timer Event Counter Pin). 1 = Toggle mode output to TMx_EXT (Timer External Capture Pin).
[20]	**PERIOSEL**	**Periodic Mode Behavior Selection Enable Bit** 0 = The behavior selection in periodic mode is Disabled. When user updates CMPDAT while timer is running in periodic mode, CNT will be reset to default value. 1 = The behavior selection in periodic mode is Enabled. When user updates CMPDAT while timer is running in periodic mode, the limitations as bellows list, If updated CMPDAT value > CNT, CMPDAT will be updated and CNT keep running continually. If updated CMPDAT value = CNT, timer time-out interrupt will be asserted immediately. If updated CMPDAT value < CNT, CNT will be reset to default value.
[19]	**INTRGEN**	**Inter-Timer Trigger Mode Enable Bit** Setting this bit will enable the inter-timer trigger capture function. The Timer0/2 will be in event counter mode and counting with external clock source or event. Also, Timer1/3 will be in trigger-counting mode of capture function. 0 = Inter-Timer Trigger Capture mode Disabled. 1 = Inter-Timer Trigger Capture mode Enabled. **Note:** For Timer1/3, this bit is ignored and the read back value is always 0.

[18:8]	**Reserved**	Reserved.
[7:0]	**PSC**	**Prescale Counter** Timer input clock or event source is divided by (PSC+1) before it is fed to the timer up counter. If this field is 0 (PSC = 0), then there is no scaling. **Note:** Update prescale counter value will reset internal 8-bit prescale counter and 24-bit up counter value.

Timer Comparator Register (TIMERx_CMP)

Register	Offset	R/W	Description	Reset Value
TIMER0_CMP	TMR01_BA+0x04	R/W	Timer0 Comparator Register	0x0000_0000
TIMER1_CMP	TMR01_BA+0x104	R/W	Timer1 Comparator Register	0x0000_0000
TIMER2_CMP	TMR23_BA+0x04	R/W	Timer2 Comparator Register	0x0000_0000
TIMER3_CMP	TMR23_BA+0x104	R/W	Timer3 Comparator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPDAT							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CMPDAT**	**Timer Comparator Value** CMPDAT is a 24-bit compared value register. When the internal 24-bit up counter value is equal to CMPDAT value, the TIF (TIMERx_INTSTS [0] Timer Interrupt Flag) will be set to 1. Time-out period = (Period of timer clock input) * (8-bit PSC + 1) * (24-bit CMPDAT). **Note 1:** Never write 0x0 or 0x1 in CMPDAT field, or the core will run into unknown state. **Note 2:** When timer is operating at continuous counting mode, the 24-bit up counter will keep counting continuously even if user writes a new value into CMPDAT field. But if timer is operating at other modes, the 24-bit up counter will restart counting from 0 and using newest CMPDAT value to be the timer compared value while user writes a new value into CMPDAT field.

Timer Interrupt Status Register (TIMERx_INTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_INTSTS	TMR01_BA+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TIMER1_INTSTS	TMR01_BA+0x108	R/W	Timer1 Interrupt Status Register	0x0000_0000
TIMER2_INTSTS	TMR23_BA+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TIMER3_INTSTS	TMR23_BA+0x108	R/W	Timer3 Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**TWKF**	**TIF**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**TWKF**	**Timer Wake-up Flag** This bit indicates the interrupt wake-up flag status of timer. 0 = Timer does not cause CPU wake-up. 1 = CPU wake-up from Idle or Power-down mode if timer time-out interrupt signal generated. **Note:** This bit is cleared by writing 1 to it.
[0]	**TIF**	**Timer Interrupt Flag** This bit indicates the interrupt flag status of timer while 24-bit timer up counter CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value. 0 = No effect. 1 = CNT value matches the CMPDAT value. **Note:** This bit is cleared by writing 1 to it.

Timer Data Register (TIMERx_CNT)

Register	Offset	R/W	Description	Reset Value
TIMER0_CNT	TMR01_BA+0x0C	R/W	Timer0 Data Register	0x0000_0000
TIMER1_CNT	TMR01_BA+0x10C	R/W	Timer1 Data Register	0x0000_0000
TIMER2_CNT	TMR23_BA+0x0C	R/W	Timer2 Data Register	0x0000_0000
TIMER3_CNT	TMR23_BA+0x10C	R/W	Timer3 Data Register	0x0000_0000

31	30	29	28	27	26	25	24
RSTACT	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CNT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT							

Bits	Description	
[31]	**RSTACT**	**Timer Data Register Reset Active (Read Only)** This bit indicates if the counter reset operation active. When user writes this CNT register, timer starts to reset its internal 24-bit timer up-counter to 0 and reload 8-bit pre-scale counter. At the same time, timer set this flag to 1 to indicate the counter reset operation is in progress. Once the counter reset operation done, timer clear this bit to 0 automatically. 0 = Reset operation is done. 1 = Reset operation triggered by writing TIMERx_CNT is in progress.
[30:24]	**Reserved**	Reserved.
[23:0]	**CNT**	**Timer Data Register** Read operation. Read this register to get CNT value. For example: If EXTCNTEN (TIMERx_CTL[24]) is 0, user can read CNT value for getting current 24-bit counter value. If EXTCNTEN (TIMERx_CTL[24]) is 1, user can read CNT value for getting current 24-bit event input counter value. Write operation. Writing any value to this register will reset current CNT value to 0 and reload internal 8-bit prescale counter.

Timer Capture Data Register (TIMERx_CAP)

Register	Offset	R/W	Description	Reset Value
TIMER0_CAP	TMR01_BA+0x10	R	Timer0 Capture Data Register	0x0000_0000
TIMER1_CAP	TMR01_BA+0x110	R	Timer1 Capture Data Register	0x0000_0000
TIMER2_CAP	TMR23_BA+0x10	R	Timer2 Capture Data Register	0x0000_0000
TIMER3_CAP	TMR23_BA+0x110	R	Timer3 Capture Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CAPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CAPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAPDAT							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CAPDAT**	**Timer Capture Data Register** When CAPEN (TIMERx_EXTCTL[3]) bit is set, the transition on the capture source matches the CAPEDGE (TIMERx_EXTCTL[14:12]) setting, CAPIF (TIMERx_EINTSTS[0]) will be set to 1 and the current timer counter value CNT (TIMERx_CNT[23:0]) will be auto-loaded into this CAPDAT field.

Timer External Control Register (TIMERx_EXTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_EXTCTL	TMR01_BA+0x14	R/W	Timer0 External Control Register	0x0000_0000
TIMER1_EXTCTL	TMR01_BA+0x114	R/W	Timer1 External Control Register	0x0000_0000
TIMER2_EXTCTL	TMR23_BA+0x14	R/W	Timer2 External Control Register	0x0000_0000
TIMER3_EXTCTL	TMR23_BA+0x114	R/W	Timer3 External Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CAPDIVSC				Reserved			
23	22	21	20	19	18	17	16
Reserved						ECNTSSEL	
15	14	13	12	11	10	9	8
Reserved	CAPEDGE			Reserved	ICAPSEL		
7	6	5	4	3	2	1	0
CNTDBEN	CAPDBEN	CAPIEN	CAPFUNCS	CAPEN	Reserved		CNTPHASE

Bits	Description	
[31:28]	**CAPDIVSC**	**Timer Capture Source Divider Scale** These bits indicate the divide scale for capture source divider. 0000 = Capture source/1. 0001 = Capture source/2. 0010 = Capture source/4. 0011 = Capture source/8. 0100 = Capture source/16. 0101 = Capture source/32. 0110 = Capture source/64. 0111 = Capture source/128. 1000 = Capture source/256. 1001~1111 = Reserved. **Note:** Sets CAPSRC (TIMERx_CTL[22]) and ICAPSEL (TIMERx_EXTCTL[10:8]) to select capture source.
[27:18]	**Reserved**	Reserved.
[17:16]	**ECNTSSEL**	**Event Counter Source Selection to Trigger Event Counter Function** 00 = Event Counter input source is from external TMx (x= 0~3) pin. 01 = Event Counter input source is from internal USB internal SOF output signal. 10 = Event Counter input source is from internal ACMP0 output signal. 11 = Event Counter input source is from internal ACMP1 output signal.
[15]	**Reserved**	Reserved.
[14:12]	**CAPEDGE**	**Timer Capture Edge Detect**

		When the first capture event is generated, the CNT (TIMERx_CNT[23:0]) will be reset to 0 and first CAPDAT (TIMERx_CAP[23:0]) should be to 0. 000 = Capture event occurred when detect falling edge transfer on capture source. 001 = Capture event occurred when detect rising edge transfer on capture source. 010 = Capture event occurred when detect both falling and rising edge transfer on capture source, and the first capture event occurred at falling edge transfer. 011 = Capture event occurred when detect both rising and falling edge transfer on capture source, and the first capture event occurred at rising edge transfer. 110 = First capture event occurred at falling edge, follows capture events are at rising edge transfer on capture source. 111 = First capture event occurred at rising edge, follows capture events are at falling edge transfer on capture source. 100, 101 = Reserved. **Note:** Set CAPSRC (TIMERx_CTL[22]) and ICAPSEL (TIMERx_EXTCTL[10:8]) to select capture source.
[11]	**Reserved**	Reserved.
[10:8]	**ICAPSEL**	**Internal Capture Source Select** 000 = Capture Function source is from internal ACMP0 output signal. 001 = Capture Function source is from internal ACMP1 output signal. 010 = Capture Function source is from HXT. 011 = Capture Function source is from LXT. 100 = Capture Function source is from HIRC. 101 = Capture Function source is from LIRC. 110, 111 = Reserved. **Note:** These bits only available when CAPSRC (TIMERx_CTL[22]) is 1.
[7]	**CNTDBEN**	**Timer External Counter Pin De-bounce Enable Bit** 0 = TMx (x= 0~3) pin de-bounce Disabled. 1 = TMx (x= 0~3) pin de-bounce Enabled. **Note:** If this bit is enabled, the edge detection of TMx pin is detected with de-bounce circuit.
[6]	**CAPDBEN**	**Timer Capture De-bounce Enable Bit** 0 = TMx_EXT (x= 0~3) pin de-bounce or ACMP output de-bounce Disabled. 1 = TMx_EXT (x= 0~3) pin de-bounce or ACMP output de-bounce Enabled. **Note:** If this bit is enabled, the edge detection of TMx_EXT pin or ACMP output is detected with de-bounce circuit.
[5]	**CAPIEN**	**Timer Capture Interrupt Enable Bit** 0 = TMx_EXT (x= 0~3) pin, ACMP, internal clock, or external clock detection Interrupt Disabled. 1 = TMx_EXT (x= 0~3) pin, ACMP, internal clock, or external clock detection Interrupt Enabled. **Note:** CAPIEN is used to enable timer capture interrupt. If CAPIEN enabled, timer will rise an interrupt when CAPIF (TIMERx_EINTSTS[0]) is 1. For example, while CAPIEN = 1, CAPEN = 1, and CAPEDGE = 00, a 1 to 0 transition on the TMx_EXT pin, ACMP, internal clock, or external clock will cause the CAPIF to be set then the interrupt signal is generated and sent to NVIC to inform CPU.
[4]	**CAPFUNCS**	**Capture Function Selection** 0 = Capture Mode Enabled. 1 = Capture and Reset Mode Enabled. **Note 1:** When CAPFUNCS is 0 and CAPIF becomes 1, the current 24-bit timer counter value (CNT value) will be saved to CAPDAT field. **Note 2:** When CAPFUNCS is 1 and CAPIF becomes 1, the current 24-bit timer counter value (CNT value) will be saved to CAPDAT field then CNT value will be reset to 0

		immediately.
[3]	**CAPEN**	**Timer Capture Function Enable Bit** This bit enables the capture input function. 0 = Timer capture function Disabled. 1 = Timer capture function Enabled.
[2:1]	**Reserved**	Reserved.
[0]	**CNTPHASE**	**Timer External Count Phase** This bit indicates the detection phase of external counting pin TMx (x= 0~3). 0 = A falling edge of external counting pin will be counted. 1 = A rising edge of external counting pin will be counted.

Timer External Interrupt Status Register (TIMERx_EINTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_EINTSTS	TMR01_BA+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TIMER1_EINTSTS	TMR01_BA+0x118	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TIMER2_EINTSTS	TMR23_BA+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TIMER3_EINTSTS	TMR23_BA+0x118	R/W	Timer3 External Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**CAPIFOV**	**CAPIF**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**CAPIFOV**	**Capture Latch Interrupt Flag Overrun Status (Read Only)** 0 = Capture latch happened when the corresponding CAPIF is 0. 1 = Capture latch happened when the corresponding CAPIF is 1. **Note:** This bit will be cleared automatically when user clears corresponding CAPIF.
[0]	**CAPIF**	**Timer Capture Interrupt Flag** This bit indicates the timer capture interrupt flag status. 0 = TMx_EXT (x= 0~3) pin, ACMP, internal clock, or external clock capture interrupt did not occur. 1 = TMx_EXT (x= 0~3) pin, ACMP, internal clock, or external clock capture interrupt occurred. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** When CAPEN (TIMERx_EXTCTL[3]) bit is set, the transition on the capture source matches the CAPEDGE (TIMERx_EXTCTL[14:12]) setting, this bit will be set to 1 by hardware. **Note 3:** There is a new incoming capture event detected before CPU clearing the CAPIF status. If the above condition occurred, the timer will keep register TIMERx_CAP unchanged and drop the new capture value.

Timer Trigger Control Register (TIMERx_TRGCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_TRGCTL	TMR01_BA+0x1C	R/W	Timer0 Trigger Control Register	0x0000_0000
TIMER1_TRGCTL	TMR01_BA+0x11C	R/W	Timer1 Trigger Control Register	0x0000_0000
TIMER2_TRGCTL	TMR23_BA+0x1C	R/W	Timer2 Trigger Control Register	0x0000_0000
TIMER3_TRGCTL	TMR23_BA+0x11C	R/W	Timer3 Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**3**	**4**	**3**	**2**	**1**	**0**
Reserved			**TRGPDMA**	**TRGDAC**	**TRGEADC**	**TRGPWM**	**TRGSSEL**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**TRGPDMA**	**Trigger PDMA Enable Bit** If this bit is set to 1, each timer time-out event or capture event can be triggered PDMA transfer. 0 = Timer interrupt trigger PDMA Disabled. 1 = Timer interrupt trigger PDMA Enabled. **Note:** If TRGSSEL (TIMERx_TRGCTL[0]) = 0, time-out interrupt signal will trigger PDMA transfer. If TRGSSEL (TIMERx_TRGCTL[0]) = 1, capture interrupt signal will trigger PDMA transfer.
[3]	**TRGDAC**	**Trigger DAC Enable Bit** If this bit is set to 1, timer time-out interrupt or capture interrupt can be triggered DAC conversion. 0 = Timer interrupt trigger DAC Disabled. 1 = Timer interrupt trigger DAC Enabled. **Note:** If TRGSSEL (TIMERx_TRGCTL[0]) = 0, time-out interrupt signal will trigger DAC conversion. If TRGSSEL (TIMERx_TRGCTL[0]) = 1, capture interrupt signal will trigger DAC conversion.
[2]	**TRGEADC**	**Trigger EADC Enable Bit** If this bit is set to 1, each timer time-out event or capture event can be triggered EADC conversion. 0 = Timer interrupt trigger EADC Disabled. 1 = Timer interrupt trigger EADC Enabled. **Note:** If TRGSSEL (TIMERx_TRGCTL[0]) = 0, time-out interrupt signal will trigger EADC

		conversion. If TRGSSEL (TIMERx_TRGCTL[0]) = 1, capture interrupt signal will trigger EADC conversion.
[1]	**TRGPWM**	**Trigger EPWM and BPWM Enable Bit** If this bit is set to 1, each timer time-out event or capture event can be as EPWM and BPWM counter clock source. 0 = Timer interrupt signal as EPWM and BPWM counter clock source Disabled. 1 = Timer interrupt signal as EPWM and BPWM counter clock source Enabled. **Note:** If TRGSSEL (TIMERx_TRGCTL[0]) = 0, time-out interrupt signal as EPWM and BPWM counter clock source. If TRGSSEL (TIMERx_TRGCTL[0]) = 1, capture interrupt signal as EPWM and BPWM counter clock source.
[0]	**TRGSSEL**	**Trigger Source Select Bit** This bit is used to select internal trigger source is form timer time-out interrupt signal or capture interrupt signal. 0 = Time-out interrupt signal is used to internal trigger EPWM, BPWM, PDMA, DAC, and EADC. 1 = Capture interrupt signal is used to internal trigger EPWM, BPWM, PDMA, DAC, and EADC.

Timer Alternative Control Register (TIMERx_ALTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_ALTCTL	TMR01_BA+0x20	R/W	Timer0 Alternative Control Register	0x0000_0000
TIMER1_ALTCTL	TMR01_BA+0x120	R/W	Timer1 Alternative Control Register	0x0000_0000
TIMER2_ALTCTL	TMR23_BA+0x20	R/W	Timer2 Alternative Control Register	0x0000_0000
TIMER3_ALTCTL	TMR23_BA+0x120	R/W	Timer3 Alternative Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**FUNCSEL**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**FUNCSEL**	**Function Selection** 0 = Timer controller is used as timer function. 1 = Timer controller is used as PWM function. **Note:** When the timer is used as PWM, the clock source of time controller will be forced to PCLKx automatically.

Timer Capture Input Noise Filter Register (TIMERx_CAPNF)

Register	Offset	R/W	Description	Reset Value
TIMER0_CAPNF	TMR01_BA+0x24	R/W	Timer0 Capture Input Noise Filter Register	0x0000_0000
TIMER1_CAPNF	TMR01_BA+0x124	R/W	Timer1 Capture Input Noise Filter Register	0x0000_0000
TIMER2_CAPNF	TMR23_BA+0x24	R/W	Timer2 Capture Input Noise Filter Register	0x0000_0000
TIMER3_CAPNF	TMR23_BA+0x124	R/W	Timer3 Capture Input Noise Filter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					CAPNFCNT		
7	6	5	4	3	2	1	0
Reserved	CAPNFSEL			Reserved			CAPNFEN

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:8]	**CAPNFCNT**	**Capture Edge Detector Noise Filter Count** These bits control the capture filter counter to count from 0 to CAPNFCNT.
[7]	**Reserved**	Reserved.
[6:4]	**CAPNFSEL**	**Capture Edge Detector Noise Filter Clock Selection** 000 = Noise filter clock is PCLKx. 001 = Noise filter clock is PCLKx/2. 010 = Noise filter clock is PCLKx/4. 011 = Noise filter clock is PCLKx/8. 100 = Noise filter clock is PCLKx/16. 101 = Noise filter clock is PCLKx/32. 110 = Noise filter clock is PCLKx/64. 111 = Noise filter clock is PCLKx/128.
[3:1]	**Reserved**	Reserved.
[0]	**CAPNFEN**	**Capture Noise Filter Enable** 0 = Capture Noise Filter function Disabled. 1 = Capture Noise Filter function Enabled.

Timer PWM Control Register (TIMERx_PWMCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCTL	TMR01_BA+0x40	R/W	Timer0 PWM Control Register	0x0000_0000
TIMER1_PWMCTL	TMR01_BA+0x140	R/W	Timer1 PWM Control Register	0x0000_0000
TIMER2_PWMCTL	TMR23_BA+0x40	R/W	Timer2 PWM Control Register	0x0000_0000
TIMER3_PWMCTL	TMR23_BA+0x140	R/W	Timer3 PWM Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DBGNACK	DBGHALT	Reserved					
23	22	21	20	19	18	17	16
Reserved							OUTMODE
15	14	13	12	11	10	9	8
Reserved						IMMLDEN	CTRLD
7	6	5	4	3	2	1	0
Reserved				CNTMODE	CNTTYPE		CNTEN

Bits	Description	
[31]	**DBGNACK**	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement effects PWM output. PWM output pin will be forced as tri-state while ICE debug mode acknowledged. 1 = ICE debug mode acknowledgement disabled. PWM output pin will keep output no matter ICE debug mode acknowledged or not. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[30]	**DBGHALT**	**ICE Debug Mode Counter Halt (Write Protect)** If debug mode counter halt is enabled, PWM counter will keep current value until exit ICE debug mode. 0 = ICE debug mode counter halt disable. 1 = ICE debug mode counter halt enable. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[29:17]	**Reserved**	Reserved.
[16]	**OUTMODE**	**PWM Output Mode** This bit controls the output mode of corresponding PWM channel. 0 = PWM independent mode. 1 = PWM complementary mode.
[15:10]	**Reserved**	Reserved.
[9]	**IMMLDEN**	**Immediately Load Enable Bit** 0 = PERIOD will load to PBUF when current PWM period is completed no matter CTRLD is enabled/disabled. If CTRLD is disabled, CMP will load to CMPBUF when current PWM period is completed; if CTRLD is enabled in up-down count type, CMP will load to CMPBUF at the center point of current period. 1 = PERIOD/CMP will load to PBUF/CMPBUF immediately when user writes

		PERIOD/CMP and current counter count is completed. **Note:** If IMMLDEN is enabled, CTRLD will be invalid.
[8]	**CTRLD**	**Center Re-load** In up-down count type, PERIOD will load to PBUF when current PWM period is completed always and CMP will load to CMPBUF at the center point of current period.
[7:4]	**Reserved**	Reserved.
[3]	**CNTMODE**	**PWM Counter Mode** 0 = Auto-reload mode. 1 = One-shot mode.
[2:1]	**CNTTYPE**	**PWM Counter Behavior Type** 00 = Up count type. 01 = Down count type. 10 = Up-down count type. 11 = Reserved.
[0]	**CNTEN**	**PWM Counter Enable Bit** 0 = PWM counter and clock prescale Stop Running. 1 = PWM counter and clock prescale Start Running.

Timer PWM Counter Clock Source Register (TIMERx_PWMCLKSRC)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCLKSRC	TMR01_BA+0x44	R/W	Timer0 PWM Counter Clock Source Register	0x0000_0000
TIMER1_PWMCLKSRC	TMR01_BA+0x144	R/W	Timer1 PWM Counter Clock Source Register	0x0000_0000
TIMER2_PWMCLKSRC	TMR23_BA+0x44	R/W	Timer2 PWM Counter Clock Source Register	0x0000_0000
TIMER3_PWMCLKSRC	TMR23_BA+0x144	R/W	Timer3 PWM Counter Clock Source Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**CLKSRC**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**CLKSRC**	**PWM Counter Clock Source Select** The PWM counter clock source can be selected from TMRx_CLK or internal timer time-out or capture event. 000 = TMRx_CLK. 001 = Internal TIMER0 time-out or capture event. 010 = Internal TIMER1 time-out or capture event. 011 = Internal TIMER2 time-out or capture event. 100 = Internal TIMER3 time-out or capture event. Others = Reserved. **Note:** If TIMER0 PWM function is enabled, the PWM counter clock source can be selected from TMR0_CLK, TIMER1 interrupt events, TIMER2 interrupt events, or TIMER3 interrupt events.

Timer PWM Counter Clock Pre-scale Register (TIMERx_PWMCLKPSC)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCLKPSC	TMR01_BA+0x48	R/W	Timer0 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER1_PWMCLKPSC	TMR01_BA+0x148	R/W	Timer1 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER2_PWMCLKPSC	TMR23_BA+0x48	R/W	Timer2 PWM Counter Clock Pre-scale Register	0x0000_0000
TIMER3_PWMCLKPSC	TMR23_BA+0x148	R/W	Timer3 PWM Counter Clock Pre-scale Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				CLKPSC			
7	6	5	4	3	2	1	0
CLKPSC							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CLKPSC**	**PWM Counter Clock Pre-scale** The active clock of PWM counter is decided by counter clock prescale and divided by (CLKPSC + 1). If CLKPSC is 0, then there is no scaling in PWM counter clock source.

Timer PWM Clear Counter Register (TIMERx_PWMCNTCLR)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCNTCLR	TMR01_BA+0x4C	R/W	Timer0 PWM Clear Counter Register	0x0000_0000
TIMER1_PWMCNTCLR	TMR01_BA+0x14C	R/W	Timer1 PWM Clear Counter Register	0x0000_0000
TIMER2_PWMCNTCLR	TMR23_BA+0x4C	R/W	Timer2 PWM Clear Counter Register	0x0000_0000
TIMER3_PWMCNTCLR	TMR23_BA+0x14C	R/W	Timer3 PWM Clear Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTCLR**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTCLR**	**Clear PWM Counter Control Bit** It is automatically cleared by hardware. 0 = No effect. 1 = Clear 16-bit PWM counter to 0x10000 in up and up-down count type and reset counter value to PERIOD in down count type.

Timer PWM Period Register (TIMERx_PWMPERIOD)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMPERIOD	TMR01_BA+0x50	R/W	Timer0 PWM Period Register	0x0000_0000
TIMER1_PWMPERIOD	TMR01_BA+0x150	R/W	Timer1 PWM Period Register	0x0000_0000
TIMER2_PWMPERIOD	TMR23_BA+0x50	R/W	Timer2 PWM Period Register	0x0000_0000
TIMER3_PWMPERIOD	TMR23_BA+0x150	R/W	Timer3 PWM Period Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PERIOD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PERIOD							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PERIOD**	**PWM Period Register** In up count type: PWM counter counts from 0 to PERIOD, and restarts from 0. In down count type: PWM counter counts from PERIOD to 0, and restarts from PERIOD. In up-down count type: PWM counter counts from 0 to PERIOD, then decrements to 0 and repeats again. In up and down count type: PWM period time = (PERIOD + 1) * (CLKPSC + 1) * TMRx_PWMCLK. In up-down count type: PWM period time = (2 * PERIOD) * (CLKPSC+ 1) * TMRx_PWMCLK. **Note:** User should take care DIRF (TIMERx_PWMCNT[16]) bit in up/down/up-down count type to monitor current counter direction in each count type.

Timer PWM Comparator Register (TIMERx_PWMCMPDAT)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCMPDAT	TMR01_BA+0x54	R/W	Timer0 PWM Comparator Register	0x0000_0000
TIMER1_PWMCMPDAT	TMR01_BA+0x154	R/W	Timer1 PWM Comparator Register	0x0000_0000
TIMER2_PWMCMPDAT	TMR23_BA+0x54	R/W	Timer2 PWM Comparator Register	0x0000_0000
TIMER3_PWMCMPDAT	TMR23_BA+0x154	R/W	Timer3 PWM Comparator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMP							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMP**	**PWM Comparator Register** PWM CMP is used to compare with PWM CNT to generate PWM output waveform, interrupt events and trigger EADC to start conversion.

Timer PWM Dead-time Control Register (TIMERx_PWMDTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMDTCTL	TMR01_BA+0x58	R/W	Timer0 PWM Dead-Time Control Register	0x0000_0000
TIMER1_PWMDTCTL	TMR01_BA+0x158	R/W	Timer1 PWM Dead-Time Control Register	0x0000_0000
TIMER2_PWMDTCTL	TMR23_BA+0x58	R/W	Timer2 PWM Dead-Time Control Register	0x0000_0000
TIMER3_PWMDTCTL	TMR23_BA+0x158	R/W	Timer3 PWM Dead-Time Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							DTCKSEL
23	22	21	20	19	18	17	16
Reserved							DTEN
15	14	13	12	11	10	9	8
Reserved				DTCNT			
7	6	5	4	3	2	1	0
DTCNT							

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**DTCKSEL**	**Dead-time Clock Select (Write Protect)** 0 = Dead-time clock source from TMRx_PWMCLK without counter clock prescale. 1 = Dead-time clock source from TMRx_PWMCLK with counter clock prescale. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[23:17]	**Reserved**	Reserved.
[16]	**DTEN**	**Enable Dead-time Insertion for PWMx_CH0 and PWMx_CH1 (Write Protect)** Dead-time insertion function is only active when PWM complementary mode is enabled. If dead- time insertion is inactive, the outputs of PWMx_CH0 and PWMx_CH1 are complementary without any delay. 0 = Dead-time insertion Disabled on the pin pair. 1 = Dead-time insertion Enabled on the pin pair. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[15:12]	**Reserved**	Reserved.
[11:0]	**DTCNT**	**Dead-time Counter (Write Protect)** The dead-time can be calculated from the following two formulas: Dead-time = (DTCNT[11:0] + 1) * TMRx_PWMCLK, if DTCKSEL is 0. Dead-time = (DTCNT[11:0] + 1) * TMRx_PWMCLK * (CLKPSC + 1), if DTCKSEL is 1. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Counter Register (TIMERx_PWMCNT)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCNT	TMR01_BA+0x5C	R	Timer0 PWM Counter Register	0x0000_0000
TIMER1_PWMCNT	TMR01_BA+0x15C	R	Timer1 PWM Counter Register	0x0000_0000
TIMER2_PWMCNT	TMR23_BA+0x5C	R	Timer2 PWM Counter Register	0x0000_0000
TIMER3_PWMCNT	TMR23_BA+0x15C	R	Timer3 PWM Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**DIRF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT							

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DIRF**	**PWM Counter Direction Indicator Flag (Read Only)** 0 = Counter is active in down count. 1 = Counter is active up count.
[15:0]	**CNT**	**PWM Counter Value Register (Read Only)** User can monitor CNT to know the current counter value in 16-bit period counter.

Timer PWM Output Mask Enable Register (TIMERx_PWMMSKEN)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMMSKEN	TMR01_BA+0x60	R/W	Timer0 PWM Output Mask Enable Register	0x0000_0000
TIMER1_PWMMSKEN	TMR01_BA+0x160	R/W	Timer1 PWM Output Mask Enable Register	0x0000_0000
TIMER2_PWMMSKEN	TMR23_BA+0x60	R/W	Timer2 PWM Output Mask Enable Register	0x0000_0000
TIMER3_PWMMSKEN	TMR23_BA+0x160	R/W	Timer3 PWM Output Mask Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						MSKEN1	MSKEN0

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**MSKEN1**	**PWMx_CH1 Output Mask Enable Bit** The PWMx_CH1 output signal will be masked when this bit is enabled. The PWMx_CH1 will output MSKDAT1 (TIMERx_PWMMSK[1]) data. 0 = PWMx_CH1 output signal is non-masked. 1 = PWMx_CH1 output signal is masked and output MSKDAT1 data.
[0]	**MSKEN0**	**PWMx_CH0 Output Mask Enable Bit** The PWMx_CH0 output signal will be masked when this bit is enabled. The PWMx_CH0 will output MSKDAT0 (TIMERx_PWMMSK[0]) data. 0 = PWMx_CH0 output signal is non-masked. 1 = PWMx_CH0 output signal is masked and output MSKDAT0 data.

Timer PWM Output Mask Data Control Register (TIMERx_PWMMSK)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMMSK	TMR01_BA+0x64	R/W	Timer0 PWM Output Mask Data Control Register	0x0000_0000
TIMER1_PWMMSK	TMR01_BA+0x164	R/W	Timer1 PWM Output Mask Data Control Register	0x0000_0000
TIMER2_PWMMSK	TMR23_BA+0x64	R/W	Timer2 PWM Output Mask Data Control Register	0x0000_0000
TIMER3_PWMMSK	TMR23_BA+0x164	R/W	Timer3 PWM Output Mask Data Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						MSKDAT1	MSKDAT0

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**MSKDAT1**	**PWMx_CH1 Output Mask Data Control Bit** This bit is used to control the output state of PWMx_CH1 pin when PWMx_CH1 output mask function is enabled (MSKEN1 = 1). 0 = Output logic Low to PWMx_CH1. 1 = Output logic High to PWMx_CH1.
[0]	**MSKDAT0**	**PWMx_CH0 Output Mask Data Control Bit** This bit is used to control the output state of PWMx_CH0 pin when PWMx_CH0 output mask function is enabled (MSKEN0 = 1). 0 = Output logic Low to PWMx_CH0. 1 = Output logic High to PWMx_CH0.

Timer PWM Brake Pin Noise Filter Register (TIMERx_PWMBNF)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMBNF	TMR01_BA+0x68	R/W	Timer0 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER1_PWMBNF	TMR01_BA+0x168	R/W	Timer1 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER2_PWMBNF	TMR23_BA+0x68	R/W	Timer2 PWM Brake Pin Noise Filter Register	0x0000_0000
TIMER3_PWMBNF	TMR23_BA+0x168	R/W	Timer3 PWM Brake Pin Noise Filter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						BKPINSRC	
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
BRKPINV	BRKFCNT			BRKNFSEL			BRKNFEN

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**BKPINSRC**	**Brake Pin Source Select** 00 = Brake pin source comes from PWM0_BRAKE0 pin. 01 = Brake pin source comes from PWM0_BRAKE1 pin. 10 = Brake pin source comes from PWM1_BRAKE0 pin. 11 = Brake pin source comes from PWM1_BRAKE1 pin.
[15:8]	**Reserved**	Reserved.
[7]	**BRKPINV**	**Brake Pin Detection Value Control Bit** 0 = Brake pin event will be detected if PWMx_BRAKEy pin status transfer from low to high in edge-detect, or pin status is high in level-detect. 1 = Brake pin event will be detected if PWMx_BRAKEy pin status transfer from high to low in edge-detect, or pin status is low in level-detect .
[6:4]	**BRKFCNT**	**Brake Pin Noise Filter Count** The fields are used to control the active noise filter sample time. Once noise filter sample time = (Period time of BRKNFSEL) * (BRKFCNT + 1).
[3:1]	**BRKNFSEL**	**Brake Pin Noise Filter Clock Selection** 000 = Noise filter clock is PCLKx. 001 = Noise filter clock is PCLKx/2. 010 = Noise filter clock is PCLKx/4. 011 = Noise filter clock is PCLKx/8. 100 = Noise filter clock is PCLKx/16. 101 = Noise filter clock is PCLKx/32. 110 = Noise filter clock is PCLKx/64.

		111 = Noise filter clock is PCLKx/128.
[0]	**BRKNFEN**	**Brake Pin Noise Filter Enable Bit** 0 = Pin noise filter detect of PWMx_BRAKEy Disabled. 1 = Pin noise filter detect of PWMx_BRAKEy Enabled.

Timer PWM System Fail Brake Control Register (TIMERx_PWMFAILBRK)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMFAILBRK	TMR01_BA+0x6C	R/W	Timer0 PWM System Fail Brake Control Register	0x0000_0000
TIMER1_PWMFAILBRK	TMR01_BA+0x16C	R/W	Timer1 PWM System Fail Brake Control Register	0x0000_0000
TIMER2_PWMFAILBRK	TMR23_BA+0x6C	R/W	Timer2 PWM System Fail Brake Control Register	0x0000_0000
TIMER3_PWMFAILBRK	TMR23_BA+0x16C	R/W	Timer3 PWM System Fail Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CORBRKEN	RAMBRKEN	BODBRKEN	CSSBRKEN

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CORBRKEN**	**Core Lockup Detection Trigger PWM Brake Function Enable Bit** 0 = Brake Function triggered by core lockup event Disabled. 1 = Brake Function triggered by core lockup event Enabled.
[2]	**RAMBRKEN**	**SRAM Parity Error Detection Trigger PWM Brake Function Enable Bit** 0 = Brake Function triggered by SRAM parity error detection Disabled. 1 = Brake Function triggered by SRAM parity error detection Enabled.
[1]	**BODBRKEN**	**Brown-out Detection Trigger PWM Brake Function Enable Bit** 0 = Brake Function triggered by BOD event Disabled. 1 = Brake Function triggered by BOD event Enabled.
[0]	**CSSBRKEN**	**Clock Security System Detection Trigger PWM Brake Function Enable Bit** 0 = Brake Function triggered by clock fail detection Disabled. 1 = Brake Function triggered by clock fail detection Enabled.

Timer PWM Brake Control Register (TIMERx_PWMBRKCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMBRKCTL	TMR01_BA+0x70	R/W	Timer0 PWM Brake Control Register	0x0000_0000
TIMER1_PWMBRKCTL	TMR01_BA+0x170	R/W	Timer1 PWM Brake Control Register	0x0000_0000
TIMER2_PWMBRKCTL	TMR23_BA+0x70	R/W	Timer2 PWM Brake Control Register	0x0000_0000
TIMER3_PWMBRKCTL	TMR23_BA+0x170	R/W	Timer3 PWM Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				BRKAODD		BRKAEVEN	
15	14	13	12	11	10	9	8
SYSLBEN	Reserved		BRKPLEN	Reserved		CPO1LBEN	CPO0LBEN
7	6	5	4	3	2	1	0
SYSEBEN	Reserved		BRKPEEN	Reserved		CPO1EBEN	CPO0EBEN

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:18]	**BRKAODD**	**PWM Brake Action Select for PWMx_CH1 (Write Protect)** 00 = PWMx_BRAKEy brake event will not affect PWMx_CH1 output. 01 = PWMx_CH1 output tri-state when PWMx_BRAKEy brake event happened. 10 = PWMx_CH1 output low level when PWMx_BRAKEy brake event happened. 11 = PWMx_CH1 output high level when PWMx_BRAKEy brake event happened. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[17:16]	**BRKAEVEN**	**PWM Brake Action Select for PWMx_CH0 (Write Protect)** 00 = PWMx_BRAKEy brake event will not affect PWMx_CH0 output. 01 = PWMx_CH0 output tri-state when PWMx_BRAKEy brake event happened. 10 = PWMx_CH0 output low level when PWMx_BRAKEy brake event happened. 11 = PWMx_CH0 output high level when PWMx_BRAKEy brake event happened. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[15]	**SYSLBEN**	**Enable System Fail As Level-detect Brake Source (Write Protect)** 0 = System fail condition as level-detect brake source Disabled. 1 = System fail condition as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[14:13]	**Reserved**	Reserved.
[12]	**BRKPLEN**	**Enable PWMx_BRAKEy Pin As Level-detect Brake Source (Write Protect)** 0 = PWMx_BRAKEy pin event as level-detect brake source Disabled. 1 = PWMx_BRAKEy pin event as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

[11:10]	**Reserved**	Reserved.
[9]	**CPO1LBEN**	**Enable Internal ACMP1_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = Internal ACMP1_O signal as level-detect brake source Disabled. 1 = Internal ACMP1_O signal as level-detect brake source Enabled. **Note 1:** Only internal ACMP1_O signal from low to high will be detected as brake event. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[8]	**CPO0LBEN**	**Enable Internal ACMP0_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = Internal ACMP0_O signal as level-detect brake source Disabled. 1 = Internal ACMP0_O signal as level-detect brake source Enabled. **Note 1:** Only internal ACMP0_O signal from low to high will be detected as brake event. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[7]	**SYSEBEN**	**Enable System Fail As Edge-detect Brake Source (Write Protect)** 0 = System fail condition as edge-detect brake source Disabled. 1 = System fail condition as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[6:5]	**Reserved**	Reserved.
[4]	**BRKPEEN**	**Enable PWMx_BRAKEy Pin As Edge-detect Brake Source (Write Protect)** 0 = PWMx_BRAKEy pin event as edge-detect brake source Disabled. 1 = PWMx_BRAKEy pin event as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[3:2]	**Reserved**	Reserved.
[1]	**CPO1EBEN**	**Enable Internal ACMP1_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = Internal ACMP1_O signal as edge-detect brake source Disabled. 1 = Internal ACMP1_O signal as edge-detect brake source Enabled. **Note 1:** Only internal ACMP1_O signal from low to high will be detected as brake event. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[0]	**CPO0EBEN**	**Enable Internal ACMP0_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = Internal ACMP0_O signal as edge-detect brake source Disabled. 1 = Internal ACMP0_O signal as edge-detect brake source Enabled. **Note 1:** Only internal ACMP0_O signal from low to high will be detected as brake event. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Pin Output Polar Control Register (TIMERx_PWMPOLCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMPOLCTL	TMR01_BA+0x74	R/W	Timer0 PWM Pin Output Polar Control Register	0x0000_0000
TIMER1_PWMPOLCTL	TMR01_BA+0x174	R/W	Timer1 PWM Pin Output Polar Control Register	0x0000_0000
TIMER2_PWMPOLCTL	TMR23_BA+0x74	R/W	Timer2 PWM Pin Output Polar Control Register	0x0000_0000
TIMER3_PWMPOLCTL	TMR23_BA+0x174	R/W	Timer3 PWM Pin Output Polar Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**PINV1**	**PINV0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**PINV1**	**PWMx_CH1 Output Pin Polar Control Bit** The bit is used to control polarity state of PWMx_CH1 output pin. 0 = PWMx_CH1 output pin polar inverse Disabled. 1 = PWMx_CH1 output pin polar inverse Enabled.
[0]	**PINV0**	**PWMx_CH0 Output Pin Polar Control Bit** The bit is used to control polarity state of PWMx_CH0 output pin. 0 = PWMx_CH0 output pin polar inverse Disabled. 1 = PWMx_CH0 output pin polar inverse Enabled.

Timer PWM Pin Output Enable Register (TIMERx_PWMPOEN)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMPOEN	TMR01_BA+0x78	R/W	Timer0 PWM Pin Output Enable Register	0x0000_0000
TIMER1_PWMPOEN	TMR01_BA+0x178	R/W	Timer1 PWM Pin Output Enable Register	0x0000_0000
TIMER2_PWMPOEN	TMR23_BA+0x78	R/W	Timer2 PWM Pin Output Enable Register	0x0000_0000
TIMER3_PWMPOEN	TMR23_BA+0x178	R/W	Timer3 PWM Pin Output Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**POEN1**	**POEN0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**POEN1**	**PWMx_CH1 Output Pin Enable Bit** 0 = PWMx_CH1 pin at tri-state mode. 1 = PWMx_CH1 pin in output mode.
[0]	**POEN0**	**PWMx_CH0 Output Pin Enable Bit** 0 = PWMx_CH0 pin at tri-state mode. 1 = PWMx_CH0 pin in output mode.

Timer PWM Software Trigger Brake Control Register (TIMERx_PWMSWBRK)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMSWBRK	TMR01_BA+0x7C	W	Timer0 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER1_PWMSWBRK	TMR01_BA+0x17C	W	Timer1 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER2_PWMSWBRK	TMR23_BA+0x7C	W	Timer2 PWM Software Trigger Brake Control Register	0x0000_0000
TIMER3_PWMSWBRK	TMR23_BA+0x17C	W	Timer3 PWM Software Trigger Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**BRKLTRG**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**BRKETRG**

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BRKLTRG**	**Software Trigger Level-detect Brake Source (Write Only) (Write Protect)** Write 1 to this bit will trigger PWM level-detect brake source, then BRKLIF0 and BRKLIF1 will be set to 1 automatically in TIMERx_PWMINTSTS1 register. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:1]	**Reserved**	Reserved.
[0]	**BRKETRG**	**Software Trigger Edge-detect Brake Source (Write Only) (Write Protect)** Write 1 to this bit will trigger PWM edge-detect brake source, then BRKEIF0 and BRKEIF1 will be set to 1 automatically in TIMERx_PWMINTSTS1 register. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Interrupt Enable Register 0 (TIMERx_PWMINTEN0)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMINTEN0	TMR01_BA+0x80	R/W	Timer0 PWM Interrupt Enable Register 0	0x0000_0000
TIMER1_PWMINTEN0	TMR01_BA+0x180	R/W	Timer1 PWM Interrupt Enable Register 0	0x0000_0000
TIMER2_PWMINTEN0	TMR23_BA+0x80	R/W	Timer2 PWM Interrupt Enable Register 0	0x0000_0000
TIMER3_PWMINTEN0	TMR23_BA+0x180	R/W	Timer3 PWM Interrupt Enable Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CMPDIEN	CMPUIEN	PIEN	ZIEN

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CMPDIEN**	**PWM Compare Down Count Interrupt Enable Bit** 0 = Compare down count interrupt Disabled. 1 = Compare down count interrupt Enabled.
[2]	**CMPUIEN**	**PWM Compare Up Count Interrupt Enable Bit** 0 = Compare up count interrupt Disabled. 1 = Compare up count interrupt Enabled.
[1]	**PIEN**	**PWM Period Point Interrupt Enable Bit** 0 = Period point interrupt Disabled. 1 = Period point interrupt Enabled. **Note:** In up-down count type, period point means the center point of current PWM period.
[0]	**ZIEN**	**PWM Zero Point Interrupt Enable Bit** 0 = Zero point interrupt Disabled. 1 = Zero point interrupt Enabled.

Timer PWM Interrupt Enable Register 1 (TIMERx_PWMINTEN1)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMINTEN1	TMR01_BA+0x84	R/W	Timer0 PWM Interrupt Enable Register 1	0x0000_0000
TIMER1_PWMINTEN1	TMR01_BA+0x184	R/W	Timer1 PWM Interrupt Enable Register 1	0x0000_0000
TIMER2_PWMINTEN1	TMR23_BA+0x84	R/W	Timer2 PWM Interrupt Enable Register 1	0x0000_0000
TIMER3_PWMINTEN1	TMR23_BA+0x184	R/W	Timer3 PWM Interrupt Enable Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							BRKLIEN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							BRKEIEN

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BRKLIEN**	**PWM Level-detect Brake Interrupt Enable Bit (Write Protect)** 0 = PWM level-detect brake interrupt Disabled. 1 = PWM level-detect brake interrupt Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:1]	**Reserved**	Reserved.
[0]	**BRKEIEN**	**PWM Edge-detect Brake Interrupt Enable Bit (Write Protect)** 0 = PWM edge-detect brake interrupt Disabled. 1 = PWM edge-detect brake interrupt Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Interrupt Status Register 0 (TIMERx_PWMINTSTS0)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMINTSTS0	TMR01_BA+0x88	R/W	Timer0 PWM Interrupt Status Register 0	0x0000_0000
TIMER1_PWMINTSTS0	TMR01_BA+0x188	R/W	Timer1 PWM Interrupt Status Register 0	0x0000_0000
TIMER2_PWMINTSTS0	TMR23_BA+0x88	R/W	Timer2 PWM Interrupt Status Register 0	0x0000_0000
TIMER3_PWMINTSTS0	TMR23_BA+0x188	R/W	Timer3 PWM Interrupt Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CMPDIF	CMPUIF	PIF	ZIF

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CMPDIF**	**PWM Compare Down Count Interrupt Flag** This bit is set by hardware when TIMERx_PWM counter in down count direction and reaches CMP. **Note 1:** If CMP is equal to PERIOD, there is no CMPDIF flag in down count type. **Note 2:** This bit is cleared by writing 1 to it.
[2]	**CMPUIF**	**PWM Compare Up Count Interrupt Flag** This bit is set by hardware when TIMERx_PWM counter in up count direction and reaches CMP. **Note 1:** If CMP is equal to PERIOD, there is no CMPUIF flag in up count type and up-down count type. **Note 2:** This bit is cleared by writing 1 to it.
[1]	**PIF**	**PWM Period Point Interrupt Flag** This bit is set by hardware when TIMERx_PWM counter reaches PERIOD. **Note 1:** In up-down count type, PIF flag means the center point flag of current PWM period. **Note 2:** This bit is cleared by writing 1 to it.
[0]	**ZIF**	**PWM Zero Point Interrupt Flag** This bit is set by hardware when TIMERx_PWM counter reaches 0. **Note:** This bit is cleared by writing 1 to it.

Timer PWM Interrupt Status Register 1 (TIMERx_PWMINTSTS1)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMINTSTS1	TMR01_BA+0x8C	R/W	Timer0 PWM Interrupt Status Register 1	0x0000_0000
TIMER1_PWMINTSTS1	TMR01_BA+0x18C	R/W	Timer1 PWM Interrupt Status Register 1	0x0000_0000
TIMER2_PWMINTSTS1	TMR23_BA+0x8C	R/W	Timer2 PWM Interrupt Status Register 1	0x0000_0000
TIMER3_PWMINTSTS1	TMR23_BA+0x18C	R/W	Timer3 PWM Interrupt Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						BRKLSTS1	BRKLSTS0
23	22	21	20	19	18	17	16
Reserved						BRKESTS1	BRKESTS0
15	14	13	12	11	10	9	8
Reserved						BRKLIF1	BRKLIF0
7	6	5	4	3	2	1	0
Reserved						BRKEIF1	BRKEIF0

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**BRKLSTS1**	**Level-detect Brake Status of PWMx_CH1 (Read Only)** 0 = PWMx_CH1 level-detect brake state is released. 1 = PWMx_CH1 at level-detect brake state. **Note:** If TIMERx_PWM level-detect brake source has released, both PWMx_CH0 and PWMx_CH1 will release brake state when current PWM period finished and resume PWMx_CH0 and PWMx_CH1 output waveform start from next full PWM period.
[24]	**BRKLSTS0**	**Level-detect Brake Status of PWMx_CH0 (Read Only)** 0 = PWMx_CH0 level-detect brake state is released. 1 = PWMx_CH0 at level-detect brake state. **Note:** If TIMERx_PWM level-detect brake source has released, both PWMx_CH0 and PWMx_CH1 will release brake state when current PWM period finished and resume PWMx_CH0 and PWMx_CH1 output waveform start from next full PWM period.
[23:18]	**Reserved**	Reserved.
[17]	**BRKESTS1**	**Edge-detect Brake Status of PWMx_CH1 (Read Only)** 0 = PWMx_CH1 edge-detect brake state is released. 1 = PWMx_CH1 at edge-detect brake state. **Note:** User can set BRKEIF1 1 to clear BRKEIF1 flag and PWMx_CH1 will release brake state when current PWM period finished and resume PWMx_CH1 output waveform start from next full PWM period.
[16]	**BRKESTS0**	**Edge -detect Brake Status of PWMx_CH0 (Read Only)** 0 = PWMx_CH0 edge-detect brake state is released. 1 = PWMx_CH0 at edge-detect brake state. **Note:** User can set BRKEIF0 1 to clear BRKEIF0 flag and PWMx_CH0 will release brake state when current PWM period finished and resume PWMx_CH0 output waveform start

		from next full PWM period.
[15:10]	**Reserved**	Reserved.
[9]	**BRKLIF1**	**Level-detect Brake Interrupt Flag on PWMx_CH1 (Write Protect)** 0 = PWMx_CH1 level-detect brake event do not happened. 1 = PWMx_CH1 level-detect brake event happened. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[8]	**BRKLIF0**	**Level-detect Brake Interrupt Flag on PWMx_CH0 (Write Protect)** 0 = PWMx_CH0 level-detect brake event do not happened. 1 = PWMx_CH0 level-detect brake event happened. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:2]	**Reserved**	Reserved.
[1]	**BRKEIF1**	**Edge-detect Brake Interrupt Flag PWMx_CH1 (Write Protect)** 0 = PWMx_CH1 edge-detect brake event do not happened. 1 = PWMx_CH1 edge-detect brake event happened. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.
[0]	**BRKEIF0**	**Edge-detect Brake Interrupt Flag on PWMx_CH0 (Write Protect)** 0 = PWMx_CH0 edge-detect brake event do not happened. 1 = PWMx_CH0 edge-detect brake event happened. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** This bit is write protected. Refer to SYS_REGLCTL register.

Timer PWM Trigger Control Register (TIMERx_PWMTRGCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMTRGCTL	TMR01_BA+0x90	R/W	Timer0 PWM Trigger Control Register	0x0000_0000
TIMER1_PWMTRGCTL	TMR01_BA+0x190	R/W	Timer1 PWM Trigger Control Register	0x0000_0000
TIMER2_PWMTRGCTL	TMR23_BA+0x90	R/W	Timer2 PWM Trigger Control Register	0x0000_0000
TIMER3_PWMTRGCTL	TMR23_BA+0x190	R/W	Timer3 PWM Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
TRGEADC	Reserved				TRGSEL		

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**TRGEADC**	**PWM Counter Event Trigger EADC Conversion Enable Bit** 0 = PWM counter event trigger EADC conversion Disabled. 1 = PWM counter event trigger EADC conversion Enabled. **Note:** Set TRGSEL (TIMERx_PWMTRGCTL[2:0]) to select PWM trigger conversion source.
[6:3]	**Reserved**	Reserved.
[2:0]	**TRGSEL**	**PWM Counter Event Source Select to Trigger Conversion** 000 = Trigger conversion at zero point (ZIF). 001 = Trigger conversion at period point (PIF). 010 = Trigger conversion at zero or period point (ZIF or PIF). 011 = Trigger conversion at compare up count point (CMPUIF). 100 = Trigger conversion at compare down count point (CMPDIF). Others = Reserved.

Timer PWM Synchronous Control Register (TIMERx_PWMSCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMSCTL	TMR01_BA+0x94	R/W	Timer0 PWM Synchronous Control Register	0x0000_0000
TIMER1_PWMSCTL	TMR01_BA+0x194	R/W	Timer1 PWM Synchronous Control Register	0x0000_0000
TIMER2_PWMSCTL	TMR23_BA+0x94	R/W	Timer2 PWM Synchronous Control Register	0x0000_0000
TIMER3_PWMSCTL	TMR23_BA+0x194	R/W	Timer3 PWM Synchronous Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							SYNCSRC
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						SYNCMODE	

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**SYNCSRC**	**PWM Synchronous Counter Start/Clear Source Select** 0 = Counter synchronous start/clear by trigger TIMER0_PWMSTRG STRGEN bit. 1 = Counter synchronous start/clear by trigger TIMER2_PWMSTRG STRGEN bit. **Note 1:** If TIMER0/1/2/3 PWM counter synchronous source are from TIMER0, TIMER0_PWMSCTL[8], TIMER1_PWMSCTL[8], TIMER2_PWMSCTL[8] and TIMER3_PWMSCTL[8] should be 0. **Note 2:** If TIMER0/1/ PWM counter synchronous source are from TIMER0, TIMER0_PWMSCTL[8] and TIMER1_PWMSCTL[8] should be set 0, and TIMER2/3/ PWM counter synchronous source are from TIMER2, TIMER2_PWMSCTL[8] and TIMER3_PWMSCTL[8] should be set 1.
[7:2]	**Reserved**	Reserved.
[1:0]	**SYNCMODE**	**PWM Synchronous Mode Enable Select** 00 = PWM synchronous function Disabled. 01 = PWM synchronous counter start function Enabled. 10 = Reserved. 11 = PWM synchronous counter clear function Enabled.

Timer PWM Synchronous Trigger Register (TIMERx_PWMSTRG)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMSTRG	TMR01_BA+0x98	W	Timer0 PWM Synchronous Trigger Register	0x0000_0000
TIMER2_PWMSTRG	TMR23_BA+0x98	W	Timer2 PWM Synchronous Trigger Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**STRGEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**STRGEN**	**PWM Counter Synchronous Trigger Enable Bit (Write Only)** PMW counter synchronous function is used to make selected PWM channels (including TIMER0/1/2/3 PWM, TIMER0/1 PWM and TIMER2/3 PWM) start counting or clear counter at the same time according to TIMERx_PWMSCTL setting. **Note:** This bit is only available in TIMER0 and TIMER2.

Timer PWM Status Register (TIMERx_PWMSTATUS)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMSTATUS	TMR01_BA+0x9C	R/W	Timer0 PWM Status Register	0x0000_0000
TIMER1_PWMSTATUS	TMR01_BA+0x19C	R/W	Timer1 PWM Status Register	0x0000_0000
TIMER2_PWMSTATUS	TMR23_BA+0x9C	R/W	Timer2 PWM Status Register	0x0000_0000
TIMER3_PWMSTATUS	TMR23_BA+0x19C	R/W	Timer3 PWM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**EADCTRGF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTMAXF**

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**EADCTRGF**	**Trigger EADC Start Conversion Flag** 0 = PWM counter event trigger EADC start conversion did not occurred. 1 = PWM counter event trigger EADC start conversion occurred. **Note:** This bit is cleared by writing 1 to it.
[15:1]	**Reserved**	Reserved.
[0]	**CNTMAXF**	**PWM Counter Equal to 0xFFFF Flag** 0 = The PWM counter value never reached its maximum value 0xFFFF. 1 = The PWM counter value has reached its maximum value. **Note:** This bit is cleared by writing 1 to it.

Timer PWM Period Buffer Register (TIMERx_PWMPBUF)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMPBUF	TMR01_BA+0xA0	R	Timer0 PWM Period Buffer Register	0x0000_0000
TIMER1_PWMPBUF	TMR01_BA+0x1A0	R	Timer1 PWM Period Buffer Register	0x0000_0000
TIMER2_PWMPBUF	TMR23_BA+0xA0	R	Timer2 PWM Period Buffer Register	0x0000_0000
TIMER3_PWMPBUF	TMR23_BA+0x1A0	R	Timer3 PWM Period Buffer Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PBUF**	**PWM Period Buffer Register (Read Only)** Used as PERIOD active register.

Timer PWM Comparator Buffer Register (TIMERx_PWMCMPBUF)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMCMPBUF	TMR01_BA+0xA4	R	Timer0 PWM Comparator Buffer Register	0x0000_0000
TIMER1_PWMCMPBUF	TMR01_BA+0x1A4	R	Timer1 PWM Comparator Buffer Register	0x0000_0000
TIMER2_PWMCMPBUF	TMR23_BA+0xA4	R	Timer2 PWM Comparator Buffer Register	0x0000_0000
TIMER3_PWMCMPBUF	TMR23_BA+0x1A4	R	Timer3 PWM Comparator Buffer Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPBUF**	**PWM Comparator Buffer Register (Read Only)** Used as CMP active register.

Timer PWM Interrupt Flag Accumulator Register (TIMERx_PWMIFA)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMIFA	TMR01_BA+0xA8	R/W	Timer0 PWM Interrupt Flag Accumulator Register	0x0000_0000
TIMER1_PWMIFA	TMR01_BA+0x1A8	R/W	Timer1 PWM Interrupt Flag Accumulator Register	0x0000_0000
TIMER2_PWMIFA	TMR23_BA+0xA8	R/W	Timer2 PWM Interrupt Flag Accumulator Register	0x0000_0000
TIMER3_PWMIFA	TMR23_BA+0x1A8	R/W	Timer3 PWM Interrupt Flag Accumulator Register	0x0000_0000

31	30	29	28	27	26	25	24
IFAEN	Reserved	IFASEL		Reserved			STPMOD
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IFACNT							
7	6	5	4	3	2	1	0
IFACNT							

Bits	Description	
[31]	**IFAEN**	**PWM Interrupt Flag Accumulator Enable Bit** 0 = PWM interrupt flag accumulator function Disabled. 1 = PWM interrupt flag accumulator function Enabled.
[30]	**Reserved**	Reserved.
[29:28]	**IFASEL**	**PWM Interrupt Flag Accumulator Source Select** 00 = Accumulate at each PWM zero point. 01 = Accumulate at each PWM period point. 10 = Accumulate at each PWM up-count compared point. 11 = Accumulate at each PWM down-count compared point.
[27:25]	**Reserved**	Reserved.
[24]	**STPMOD**	**PWM Accumulator Stop Mode Enable Bit** 0 = PWM interrupt accumulator event to stop counting Disabled. 1 = PWM interrupt accumulator event to stop counting Enabled.
[23:16]	**Reserved**	Reserved.
[15:0]	**IFACNT**	**PWM Interrupt Flag Accumulator Counter** This field sets the count number which defines (IFACNT+1) times of specify PWM interrupt occurs to set IFAIF bit to request the PWM accumulator interrupt. PWM accumulator flag (IFAIF) will be set in every (IFACNT+1) times during the the PWM counter operation.

Timer PWM Accumulator Interrupt Status Register (TIMERx_PWMAINTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMAINTSTS	TMR01_BA+0xAC	R/W	Timer0 PWM Accumulator Interrupt Flag Register	0x0000_0000
TIMER1_PWMAINTSTS	TMR01_BA+0x1AC	R/W	Timer1 PWM Accumulator Interrupt Flag Register	0x0000_0000
TIMER2_PWMAINTSTS	TMR23_BA+0xAC	R/W	Timer2 PWM Accumulator Interrupt Flag Register	0x0000_0000
TIMER3_PWMAINTSTS	TMR23_BA+0x1AC	R/W	Timer3 PWM Accumulator Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**IFAIF**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**IFAIF**	**PWM Interrupt Flag Accumulator Interrupt Flag** This bit is set by hardware when the accumulator value reaches (IFACNT+1) **Note 1:** This bit is cleared by writing 1 to it. **Note 2:** If APDMAEN (TIMERx_PWMAPDMACTL[0]) is set, this bit will be auto clear after PDMA transfer done.

Timer PWM Accumulator Interrupt Enable Register (TIMERx_PWMAINTEN)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMAINTEN	TMR01_BA+0xB0	R/W	Timer0 PWM Accumulator Interrupt Enable Register	0x0000_0000
TIMER1_PWMAINTEN	TMR01_BA+0x1B0	R/W	Timer1 PWM Accumulator Interrupt Enable Register	0x0000_0000
TIMER2_PWMAINTEN	TMR23_BA+0xB0	R/W	Timer2 PWM Accumulator Interrupt Enable Register	0x0000_0000
TIMER3_PWMAINTEN	TMR23_BA+0x1B0	R/W	Timer3 PWM Accumulator Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**IFAIEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**IFAIEN**	**PWM Interrupt Flag Accumulator Interrupt Enable Bit** 0 = Interrupt Flag Accumulator interrupt Disabled. 1 = Interrupt Flag Accumulator interrupt Enabled.

Timer PWM Accumulator PDMA Control Register (TIMERx_PWMAPDMACTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMAPDMACTL	TMR01_BA+0xB4	R/W	Timer0 PWM Accumulator PDMA Control Register	0x0000_0000
TIMER1_PWMAPDMACTL	TMR01_BA+0x1B4	R/W	Timer1 PWM Accumulator PDMA Control Register	0x0000_0000
TIMER2_PWMAPDMACTL	TMR23_BA+0xB4	R/W	Timer2 PWM Accumulator PDMA Control Register	0x0000_0000
TIMER3_PWMAPDMACTL	TMR23_BA+0x1B4	R/W	Timer3 PWM Accumulator PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**APDMAEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**APDMAEN**	**PWM Accumulator PDMA Enable Bit** 0 = PWM interrupt accumulator event to trigger PDMA transfer Disabled. 1 = PWM interrupt accumulator event to trigger PDMA transfer Enabled.

Timer PWM External Event Trigger Control Register (TIMERx_PWMEXTETCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_PWMEXTETCTL	TMR01_BA+0xB8	R/W	Timer0 PWM External Event Trigger Control Register	0x0000_0000
TIMER1_PWMEXTETCTL	TMR01_BA+0x1B8	R/W	Timer1 PWM External Event Trigger Control Register	0x0000_0000
TIMER2_PWMEXTETCTL	TMR23_BA+0xB8	R/W	Timer2 PWM External Event Trigger Control Register	0x0000_0000
TIMER3_PWMEXTETCTL	TMR23_BA+0x1B8	R/W	Timer3 PWM External Event Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				EXTTRGS			
7	6	5	4	3	2	1	0
Reserved		CNTACTS		Reserved			EXTETEN

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**EXTTRGS**	**External Trigger Pin Selection** 0000 = Trigger source form INT0 pin. 0001 = Trigger source form INT1 pin. 0010 = Trigger source form INT2 pin. 0011 = Trigger source form INT3 pin. 0100 = Trigger source form INT4 pin. 0101 = Trigger source form INT5 pin. 0110 = Trigger source form INT6 pin. 0111 = Trigger source form INT7 pin. Other = Reserved.
[5:4]	**CNTACTS**	**Counter Action Selection** 00 = External pin event to trigger PWM counter reset. 01 = External pin event to trigger PWM counter start. 10 = External pin event to trigger PWM counter reset and start. 11 = Reserved.
[3:1]	**Reserved**	Reserved.
[0]	**EXTETEN**	**External Pin Event Trigger Enable Bit** 0 = External pin event trigger function Disabled. 1 = External pin event trigger function Enabled.

6.11 Basic PWM Generator and Capture Timer (BPWM)

6.11.1 Overview

The chip provides two BPWM generators — BPWM0 and BPWM1 as shown in Figure 6.11-1. Each BPWM supports 6 channels of BPWM output or input capture. There is a 12-bit prescaler to support flexible clock to the 16-bit BPWM counter with 16-bit comparator. The BPWM counter supports up, down and up-down counter types, all 6 channels share one counter. BPWM uses the comparator compared with counter to generate events. These events are used to generate BPWM pulse, interrupt and trigger signal for EADC0/1/2 to start conversion. For BPWM output control unit, it supports polarity output, independent pin mask and tri-state output enable.

The BPWM generator also supports input capture function to latch BPWM counter value to corresponding register when input channel has a rising transition, falling transition or both transition is happened.

6.11.2 Features

6.11.2.1 BPWM Function Features

- Supports maximum clock frequency up to maximum PLL frequency.
- Supports up to two BPWM modules; each module provides 6 output channels
- Supports independent mode for BPWM output/Capture input channel
- Supports 12-bit prescalar from 1 to 4096
- Supports 16-bit resolution BPWM counter; each module provides 1 BPWM counter
  - Up, down and up/down counter operation type
- Supports mask function and tri-state enable for each BPWM pin
- Supports interrupt in the following events:
  - BPWM counter matches 0, period value or compared value
- Supports trigger EADC0/1/2 in the following events:
  - BPWM counter matches 0, period value or compared value

6.11.2.2 Capture Function Features

- Supports up to 12 capture input channels with 16-bit resolution
- Supports rising or falling capture condition
- Supports input rising/falling capture interrupt
- Supports rising/falling capture with counter reload option

6.11.3 Block Diagram

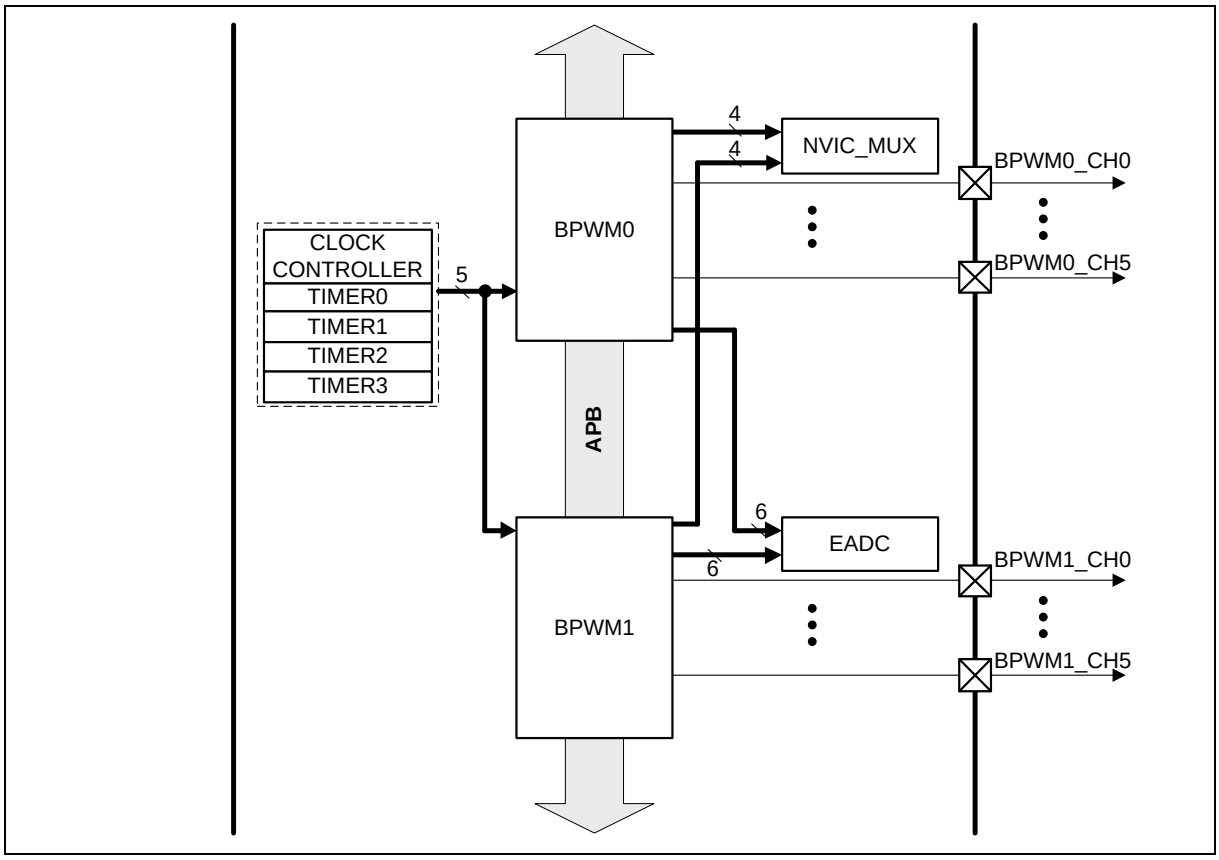


Figure 6.11-1 BPWM Generator Overview Block Diagram

Each BPWM generator has only one clock source inputs and can be selected from BPWM Clock or four TIMER trigger BPWM outputs as shown in Figure 6.11-2 by ECLKSRC0 (BPWM_CLKSRC[2:0]) for BPWM_CLK0. In general case, BPWM0 Clock must be selected from PCLK0 by setting BPWM0SEL (CLK_CLKSEL2[8]) to 1 and BPWM1 Clock from PCLK1 by setting BPWM1SEL (CLK_CLKSEL2[9]) to 1.

When operating in maximum PLL clock frequency as shown in Figure 6.11-3, and Table 6.11-1 BPWM Clock Source Control Registers Setting Table

. BPWM0 and BPWM1 clock must be selected to PLL clock by setting BPWM0SEL (CLK_CLKSEL2[8]) and BPWM1SEL (CLK_CLKSEL2[9]) to 0.

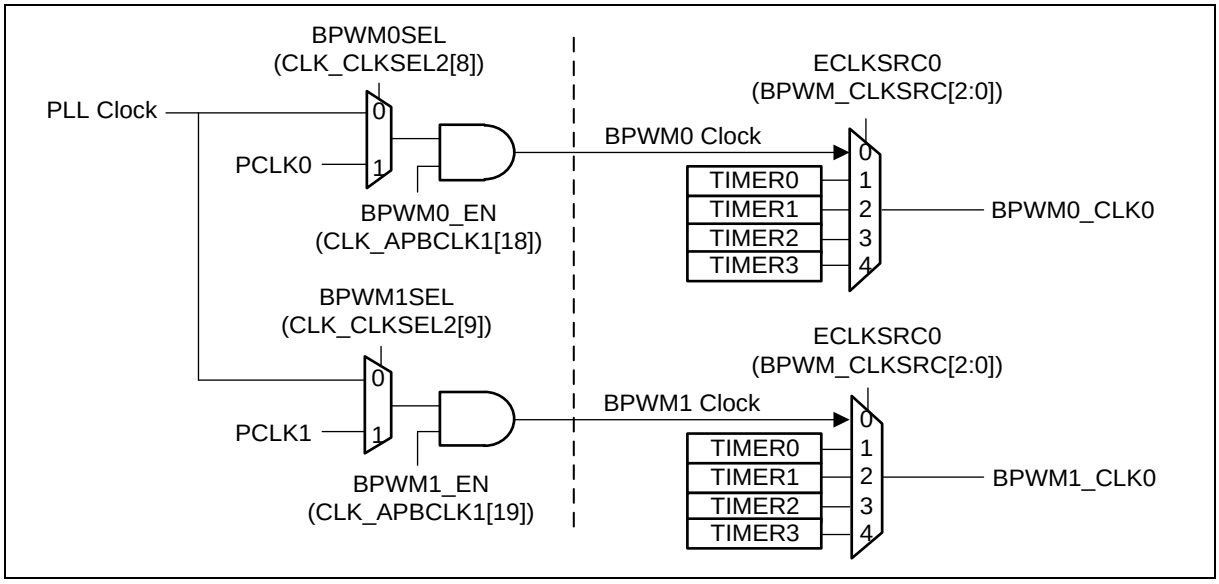


Figure 6.11-2 BPWM Clock Source Control

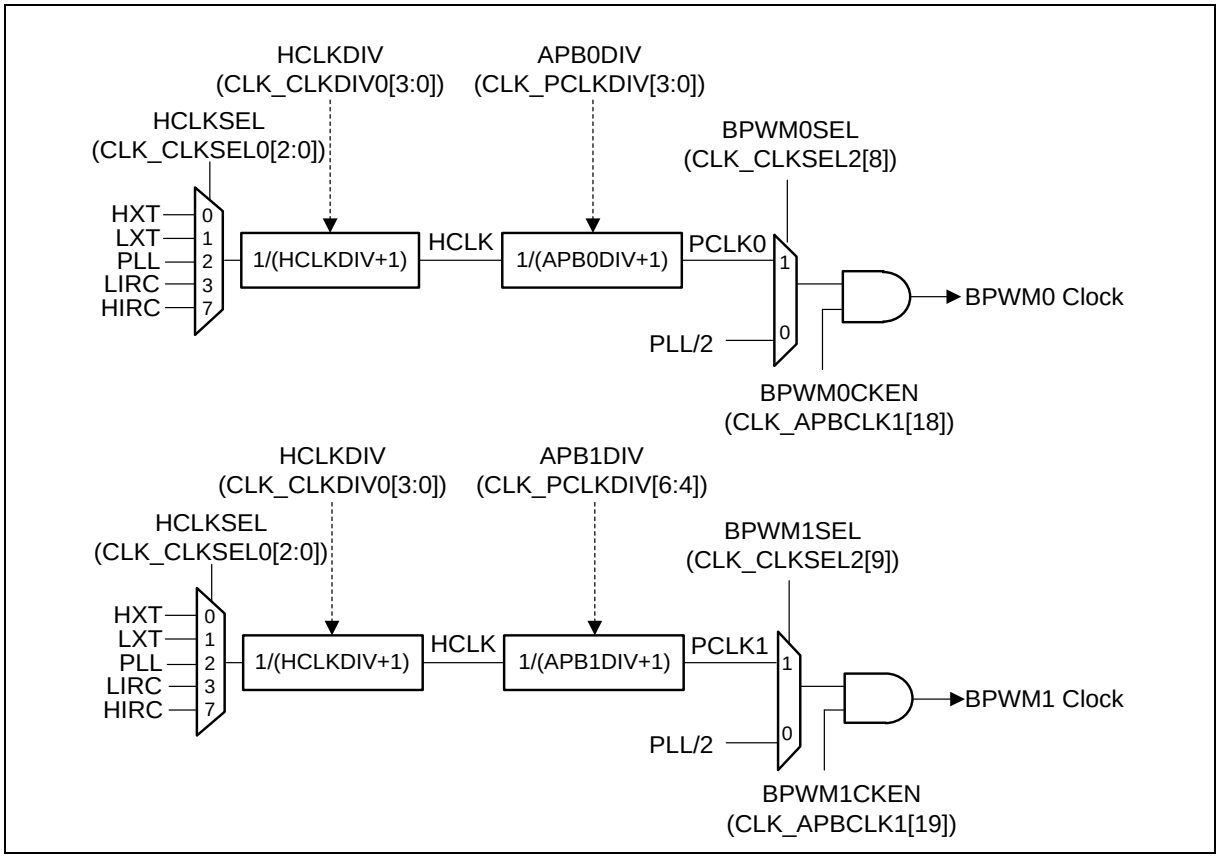


Figure 6.11-3 BPWM Clock Source Control

Frequency Ratio PCLK:BPWM Clock	HCLK	PCLK	BPWM Clock	HCLKSEL CLK_CLKSEL0[2:0]	HCLKDIV CLK_CLKDIV0[3:0]	APBnDIV (CLK_PCLKDIVn [2+4n:4n]), N Denotes 0 Or 1	BPWMnSEL (CLK_CLKSEL2[N+8]), N Denotes 0 Or 1
1:1	HCLK	PCLK	PCLK	Don't care	Don't care	Don't care	1
1:2	PLL	HCLK/ 2	HCLK	2	0	1	0

Table 6.11-1 BPWM Clock Source Control Registers Setting Table

Figure 6.11-4 illustrates the architecture of BPWM Independent mode. All six channels share the same counter. When the counter counts to 0, PERIOD (BPWM_PERIOD[15:0]) or equal to the comparator, events will be generated. These events are passed to the corresponding generators to generate BPWM pulse, interrupt signal and trigger signal for EADC0/1/2 to start conversion. Output control is used to change the BPWM pulse output state.

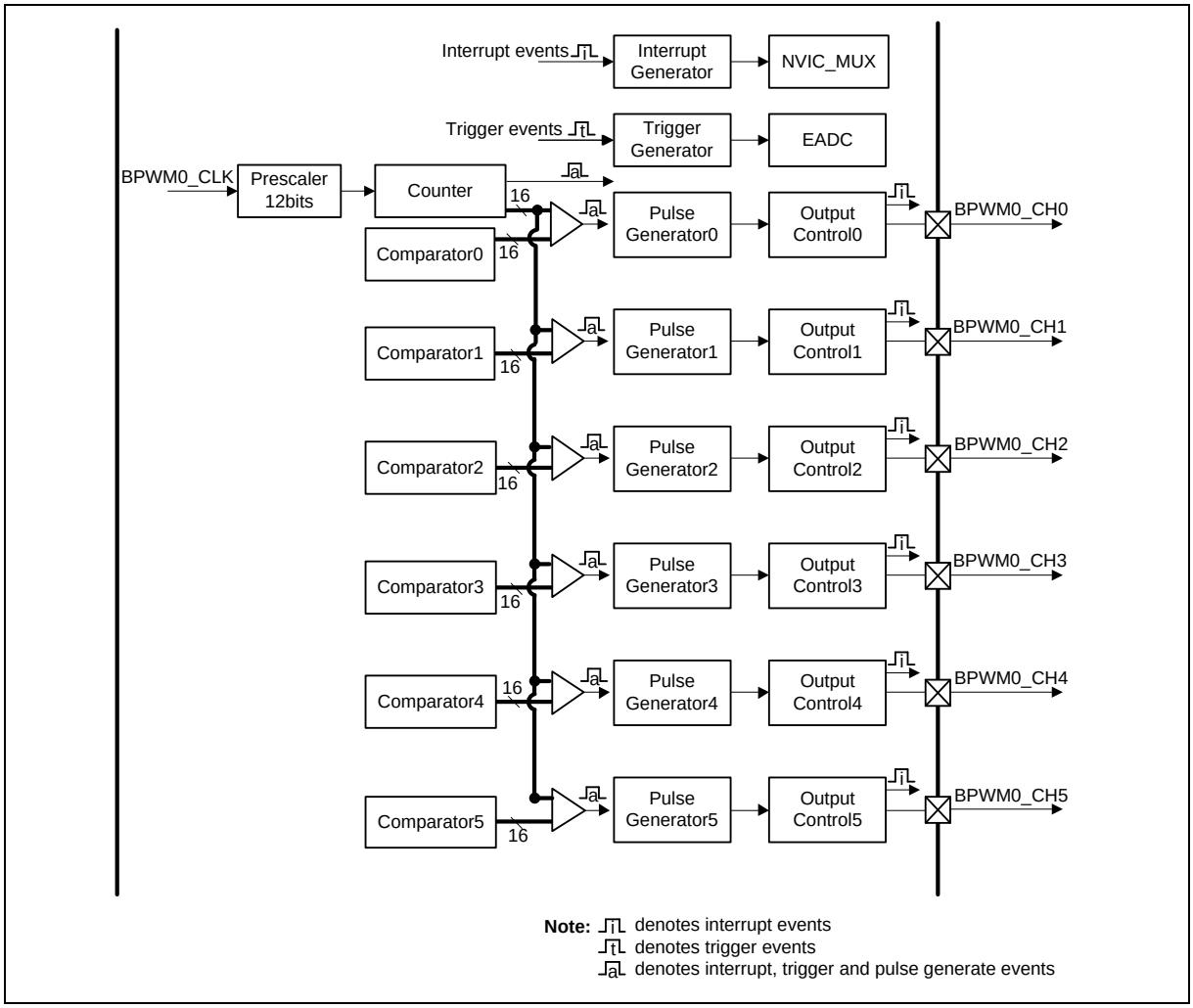


Figure 6.11-4 BPWM Independent Mode Architecture Diagram

6.11.4 Basic Configuration

6.11.4.1 BPWM0 Basic Configuration

- Clock Source Configuration

- Select the source of BPWM0 peripheral clock on BPWM0SEL (CLK_CLKSEL2[8]).
- Enable BPWM0 peripheral clock in BPWM0CKEN (CLK_APBCLK1[18]).

- Reset Configuration
  - Reset BPWM0 controller in BPWM0RST (SYS_IPRST2[18]).

6.11.4.2 BPWM1 Basic Configuration

- Clock Source Configuration
  - Select the source of BPWM1 peripheral clock on BPWM1SEL (CLK_CLKSEL2[9]).
  - Enable BPWM1 peripheral clock in BPWM1CKEN (CLK_APBCLK1[19]).
- Reset Configuration
  - Reset BPWM1 controller in BPWM1RST (SYS_IPRST2[19]).

6.11.5 Functional Description

6.11.5.1 BPWM Prescaler

The BPWM prescaler is used to divide clock source, prescaler counting CLKPSC +1 times, and BPWM counter only count once. The prescale is set by CLKPSC (BPWM_CLKPSC[11:0]). Figure 6.11-5 shows an example of BPWM channel 0 CLKPSC waveform. The prescale counter will reload CLKPSC in the beginning of the next prescale counter down-count.

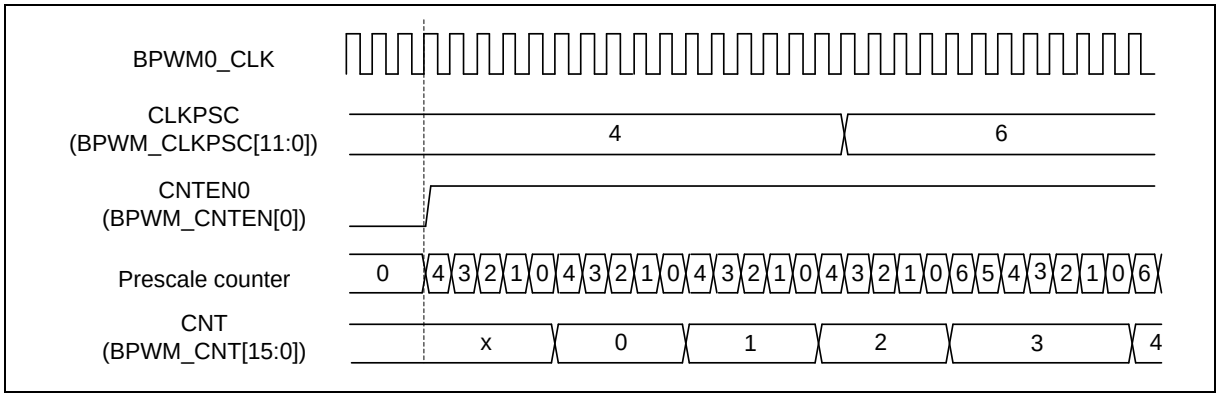


Figure 6.11-5 BPWM_CH0 CLKPSC Waveform

6.11.5.2 BPWM Counter

BPWM has one counter, and supports 3 counter types operation: Up Counter, Down Counter and Up-Down Counter types.

For BPWM channel0, CNT(BPWM_CNT[15:0]) can clear to 0x00 by CNTCLR0 (BPWM_CNTCLR[0]) when prescale counter down count to 0, and CNTCLR0(BPWM_CNTCLR[0]) will be set 0 by hardware automatically.

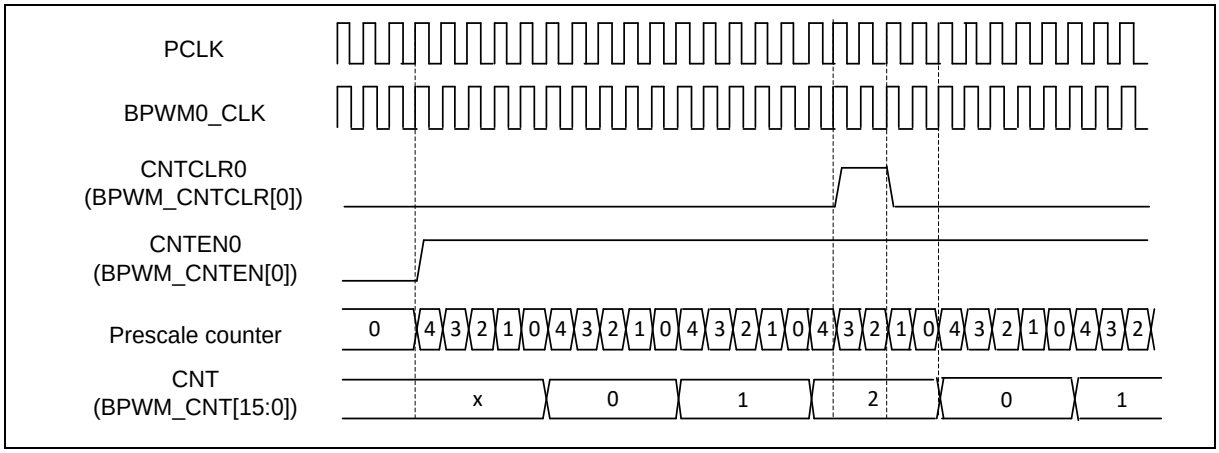


Figure 6.11-6 BPWM Counter Clear Waveform

6.11.5.3 Up Counter Type

In the up counter operation, the 16 bits BPWM counter is an up counter and starts up-counting from 0 to PERIOD (BPWM_PERIOD) to finish a BPWM period. The current counter value can be found by reading the CNT (BPWM_CNT[15:0]). BPWM generates zero point event when counter counts to 0 and generates period point event when counting to PERIOD. An example of the period time in up counter type, the BPWM period time = (PERIOD+1) * (CLKPSC+1) * BPWMx_CLK clock time, as shown in Figure 6.11-7.

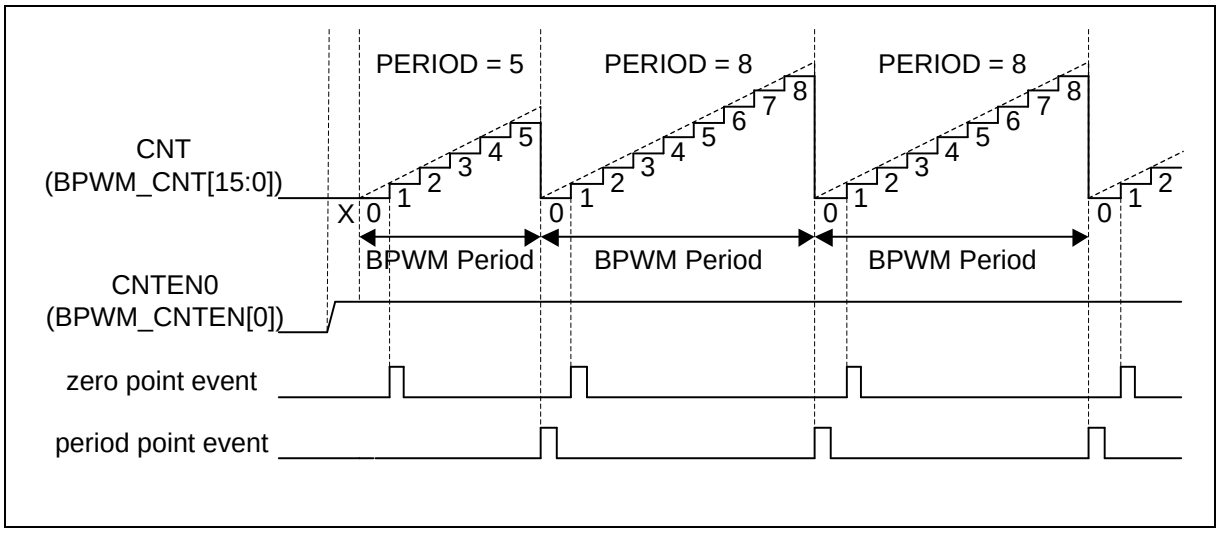


Figure 6.11-7 BPWM Up Counter Type

6.11.5.4 Down Counter Type

In the down counter operation, the 16 bits BPWM counter is a down counter and starts down-counting from PERIOD to 0 to finish a BPWM period. The current counter value can be found by reading the CNT. BPWM generates zero point event when counter counts to 0 and generates period point event when counting to PERIOD. An example of the period time in down counter type, the BPWM period time = (PERIOD+1) * (CLKPSC+1) * BPWMx_CLK clock time, as shown in Figure 6.11-8.

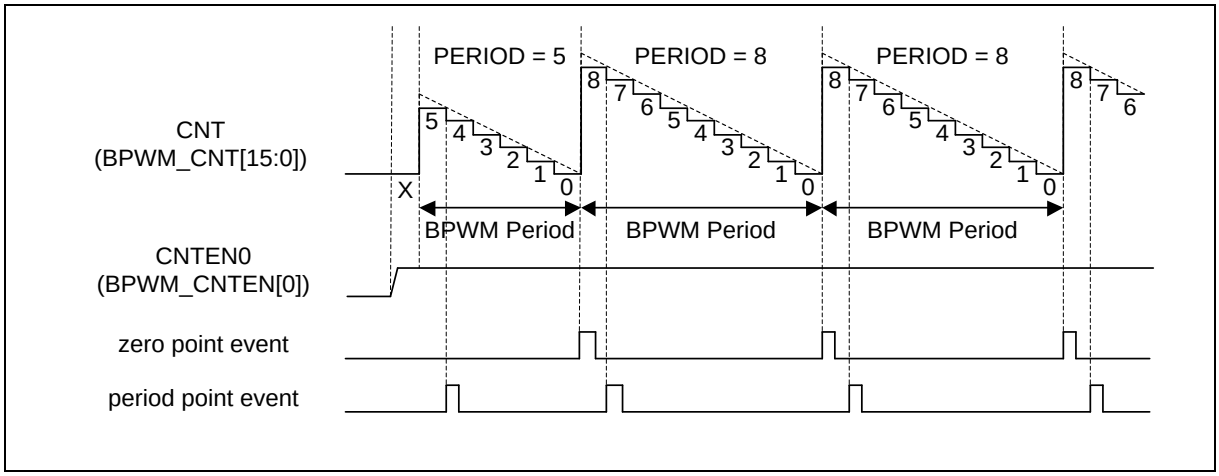
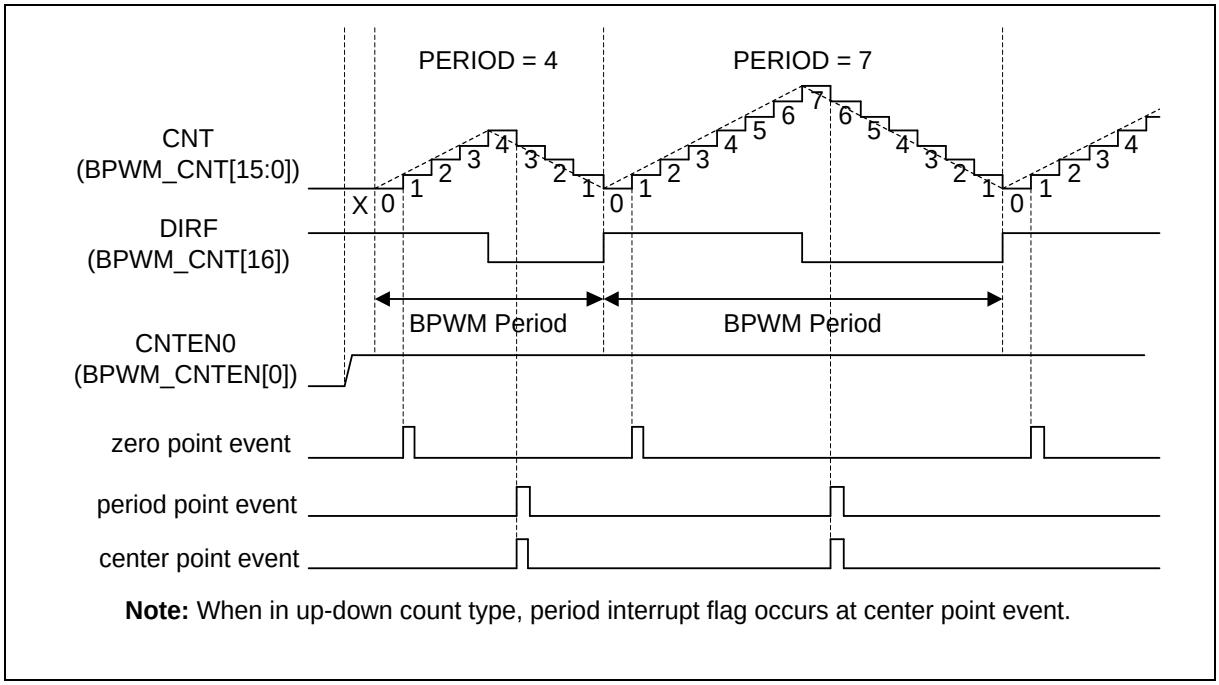


Figure 6.11-8 BPWM Down Counter Type

6.11.5.5 Up-Down Counter Type

In the up-down counter operation, the 16 bits BPWM counter is an up-down counter and starts counting-up from 0 to PERIOD and then starts counting down to 0 to finish a BPWM period. The current counter value can be found by reading the CNT. BPWM generates zero point event when counter counts to 0 and generates center point event when counting to PERIOD. An example of the period time in up-down counter type, the BPWM period time = (2xPERIOD) * (CLKPSC+1) * BPWMx_CLK clock time,as shown in Figure 6.11-9. The DIRF (BPWM_CNT[16]) is counter direction indicator flag, where high is up counting, and low is down counting.



Note: When in up-down count type, period interrupt flag occurs at center point event.

Figure 6.11-9 BPWM Up-Down Counter Type

6.11.5.6 BPWM Comparator

The CMPDAT (BPWM_CMPDATn[15:0]) is a basic comparator register of BPWM channel n; each channel only has one CMPDAT. The CMPDAT's value is continuously compared to the counter value. When the counter is equal to compared register, BPWM generates an event and uses the event to generate BPWM pulse, interrupt or use to trigger EADC0/1/2. In up-down counter type, two events will

be generated in a BPWM period as shown in Figure 6.11-10.

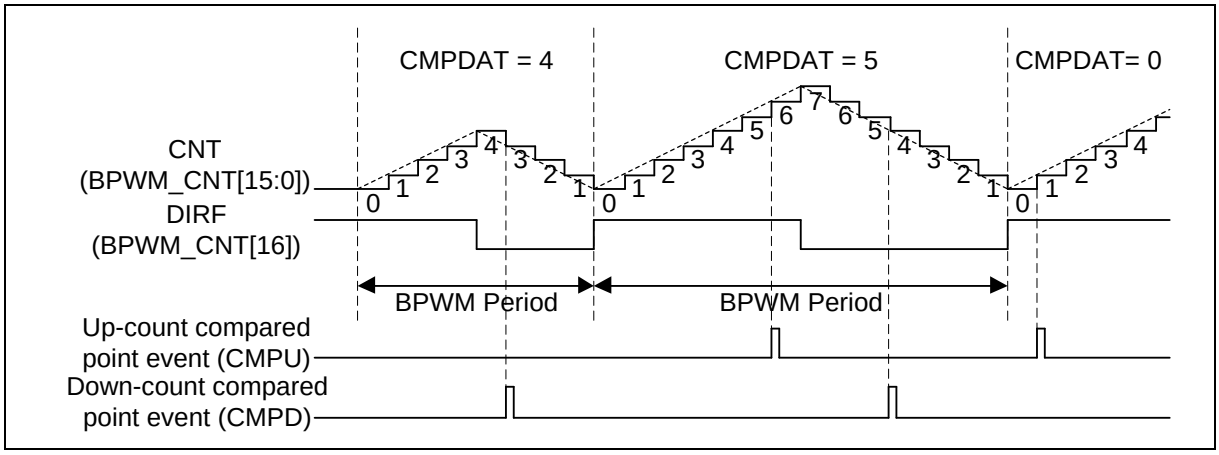


Figure 6.11-10 BPWM CMPDAT Events in Up-Down Counter Type

6.11.5.7 Period Loading Mode

Period Loading mode is the default loading mode. It has lowest priority in loading modes. PERIOD and CMPDAT will both load to their buffer while a period is completed. For example, after BPWM counter up counts from 0 to PERIOD in up-counter operation or down counts from PERIOD to 0 in the down-counter operation or up counts from 0 to PERIOD and then down counts to 0 in up-down counter operation.

Figure 6.11-11 shows period loading timing of up-count operation, where PERIOD DATA0 denotes the initial data of PERIOD, PERIOD DATA1 denotes the first updated PERIOD data by software and so on, CMPDAT also follows this rule. The following describes steps sequence of Figure 6.11-11. User can know the PERIOD and CMPDAT update condition, by watching BPWM period and CMPU event.

1. Software writes CMPDAT DATA1 to CMPDAT at point 1.
2. Hardware loads CMPDAT DATA1 to CMPBUF at the end of PWM period at point 2.
3. Software writes PERIOD DATA1 to PERIOD at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of PWM period at point 4.
5. Software writes PERIOD DATA2 to PERIOD at point 5.
6. Hardware loads PERIOD DATA2 to PBUF at the end of PWM period at point 6.

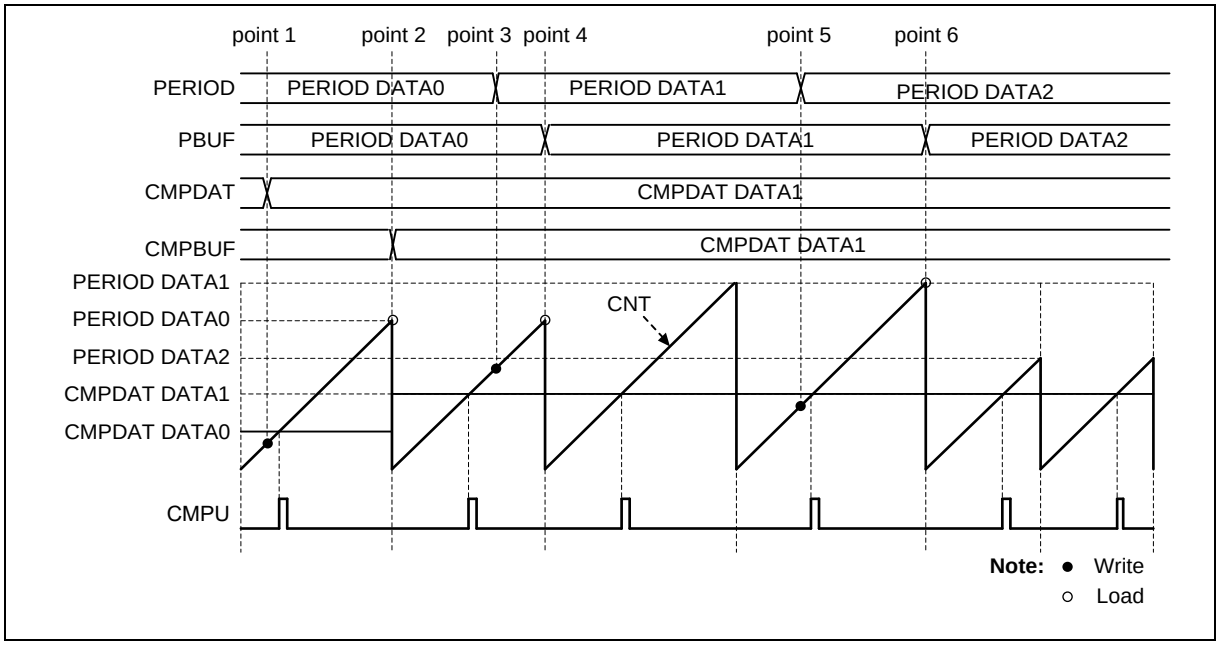


Figure 6.11-11 Period Loading Mode with Up-Counter Type

6.11.5.8 Immediately Loading Mode

If the IMMLDENn (BPWM_CTL0[21:16]) bit which corresponds to BPWM channel n is set to 1, software will load a value to buffer from PERIOD and CMPDAT immediately while software updates PERIOD or CMPDAT. If the update PERIOD value is less than current counter value, counter will count to 0xFFFF, when counter count to 0xFFFF and prescale count to 0, the flag CNTMAXF0(BPWM_STATUS[0]) will raise, and then counter will count wraparound. Immediately loading mode has the highest priority. If IMMLDENn has been set, other loading mode for channel n will become invalid. Figure 6.11-12 shows an example and its steps sequence is described below.

1. Software writes CMPDAT DATA1 and hardware immediately loading CMPDAT DATA1 to CMPBUF at point 1.
2. Software writes PERIOD DATA1 which is greater than current counter value at point 2; counter will continue counting until equal to PERIOD DATA1 to finish a period loading.
3. Software writes PERIOD DATA2 which is less than the current counter value at point 3; counter will continue counting to its maximum value 0xFFFF and count wraparound from 0 to PERIOD DATA2 to finish this period loading.

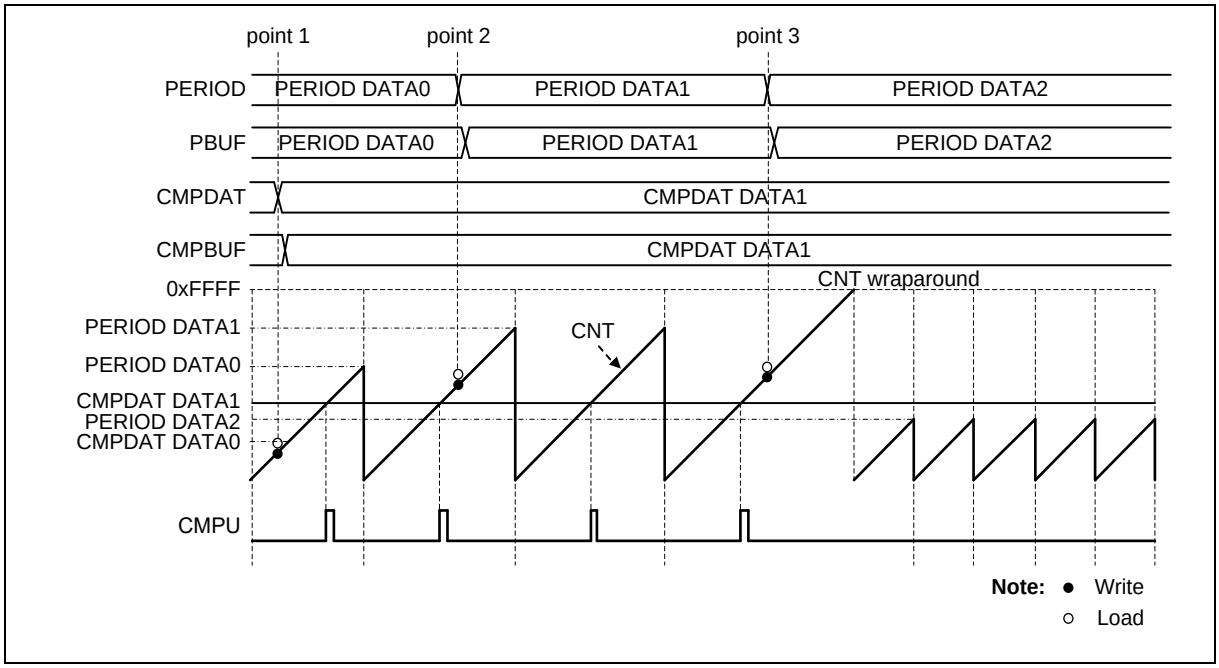


Figure 6.11-12 Immediately Loading Mode with Up-Counter Type

6.11.5.9 Center Loading Mode

If the CTRLDn (BPWM_CTL0[5:0]) bit which corresponds to BPWM channel n is set to 1 and in up-down counter type, CMPDAT will load to CMPBUFn in center of a period, that is, counter counts to PERIOD. PERIOD loading timing is the same as period loading mode. Figure 6.11-13 shows an example and its steps sequence is described below.

1. Software writes CMPDAT DATA1 at point 1.
2. Hardware loads CMPDAT DATA1 to CMPBUF at center of PWM period at point 2.
3. Software writes PERIOD DATA1 at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of PWM period at point 4.
5. Software writes CMPDAT DATA2 at point 5.
6. Hardware loads CMPDAT DATA2 to CMPBUF at center of PWM period at point 6.
7. Software writes PERIOD DATA2 at point 7.
8. Hardware loads PERIOD DATA2 to PBUF at the end of PWM period at point 8.

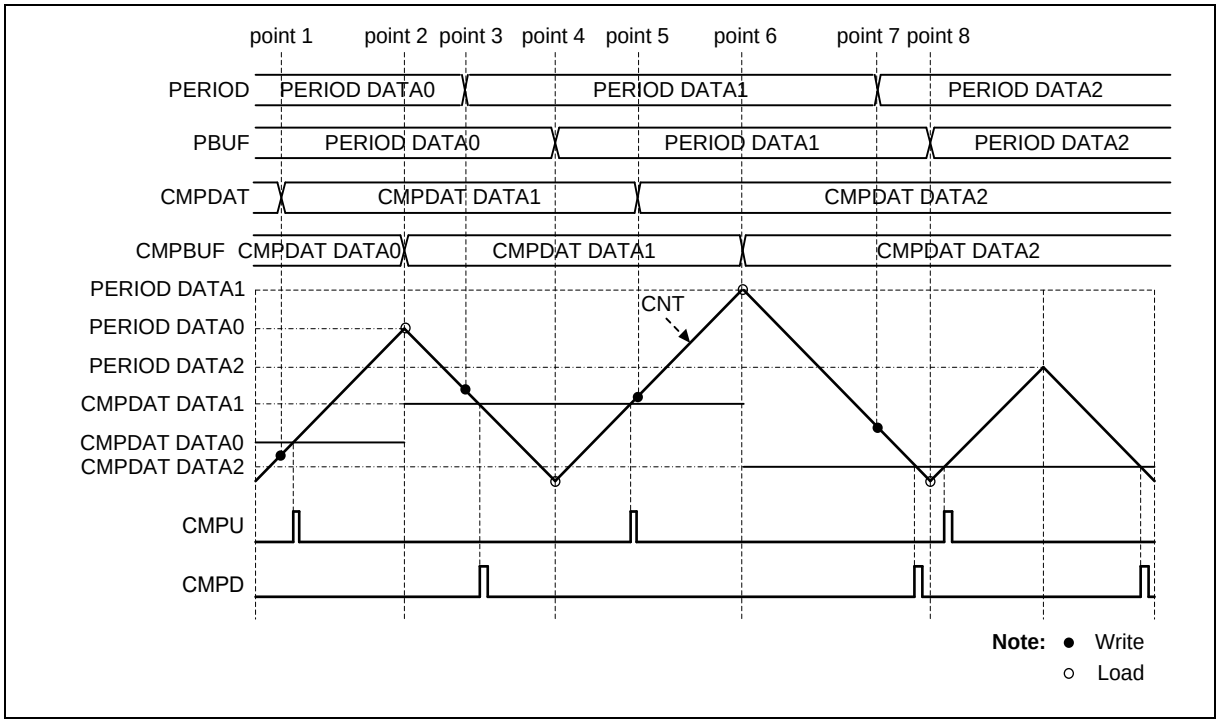


Figure 6.11-13 Center Loading Mode with Up-Down-Counter Type

6.11.5.10 BPWM Pulse Generator

The BPWM pulse generator uses counter and comparator events to generate BPWM pulse. The events are: zero point, period point in up counter type and down counter type, center point in up-down counter type and counter equal to comparator point in three types. As to up-down counter type, there are two counter equal comparator points, one at up count another at down count.

Each event point can decide BPWM waveform to do nothing (X), set Low (L), set High (H) or toggle (T) by setting BPWM_WGCTL0 and BPWM_WGCTL1 registers. Using these points can easily generate asymmetric BPWM pulse or variant waveform as shown in Figure 6.11-14. In the figure, there is a comparator n to generate BPWM pulse, where n denotes channel number 0 to 5. CMPU denotes CNT is equal to CMPDAT when counting up, and CMPD denotes CNT is equal to CMPDAT when counting down.

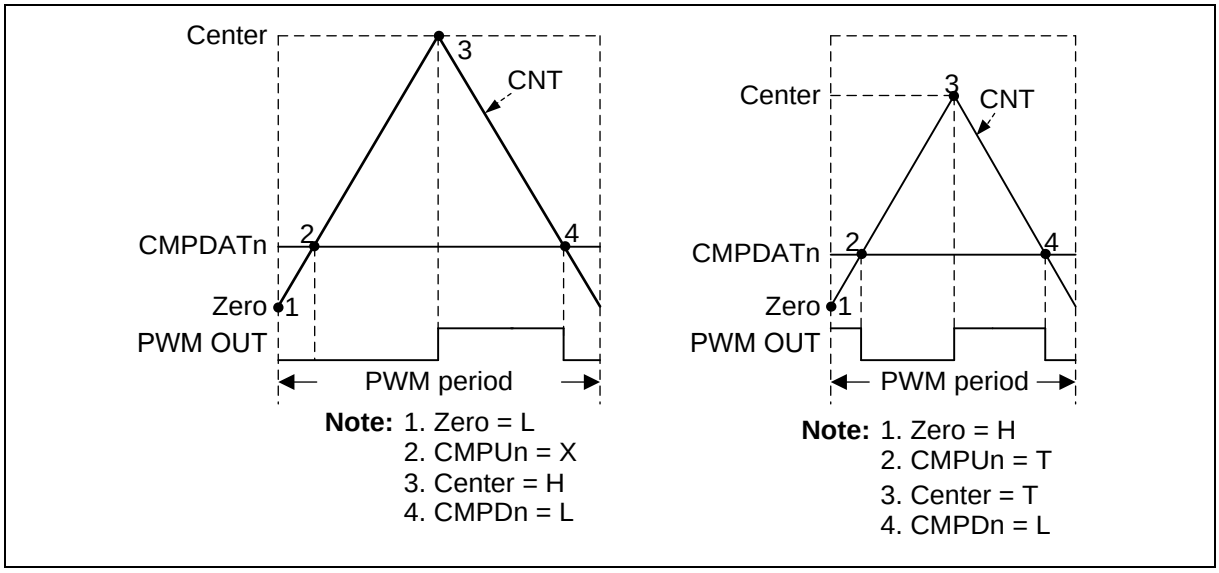


Figure 6.11-14 BPWM Pulse Generation (Left: Asymmetric Pulse, Right: Variety Pulse)

The generation events may be sometimes set to the same value, as the reason, events priority between different counter types are list below, up counter type (Table 6.11-2), down counter type (Table 6.11-3) and up-down counter type (Table 6.11-4). By using event priority, user can easily generate 0% to 100% duty pulse as shown in Figure 6.11-15.

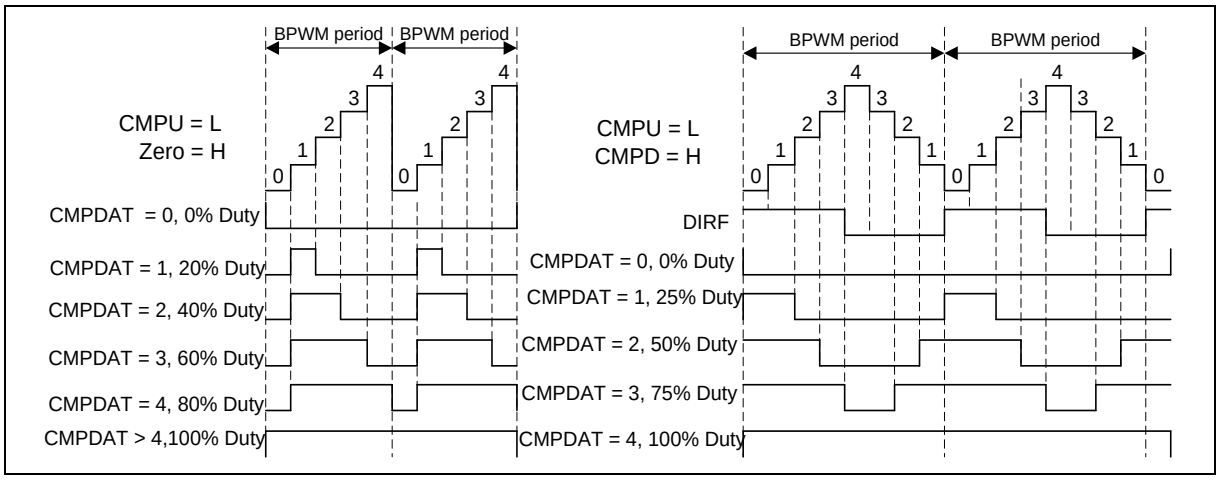


Figure 6.11-15 BPWM 0% to 100% Pulse Generation (Left: Up Counter Type, Right: Up-down Counter Type)

Priority	Up Event
1 (Highest)	Period event (CNT = PERIOD)
2	Compare up event(CNT = CMPUn)
3 (Lowest)	Zero event (CNT = 0)

Table 6.11-2 BPWM Pulse Generation Event Priority for Up-Counter

Priority	Down Event
1 (Highest)	Zero event (CNT = 0)

2	Compare down event (CNT = CMPDn)
3 (Lowest)	Period event (CNT = PERIOD)

Table 6.11-3 BPWM Pulse Generation Event Priority for Down-Counter

Priority	Up Event	Down Event
1 (Highest)	Compare up event (CNT = CMPUn)	Compare down event (CNT = CMPDn)
2 (Lowest)	Zero event (CNT = 0)	Period (center) event (CNT =PERIOD)

Table 6.11-4 BPWM Pulse Generation Event Priority for Up-Down-Counter

6.11.5.11 Synchronous function

To start BPWM and EPWM counters in the same time, user have to set the BPWM Synchronous Start Control Register (BPWM_SSCTL[0]) to enable the channel counters which are planned to start counting together, and select the SSRC(BPWM_SSCTL[9:8]) to choose the Synchronous Start source, followed by setting the BPWM Synchronous Start Trigger Register CNTSEN (BPWM_SSTRG[0]).

6.11.5.12 BPWM Output Control

After BPWM pulse generation, there are three steps to control the output of BPWM channels. There are Mask, Pin Polarity and Output Enable three steps as shown in Figure 6.11-16.

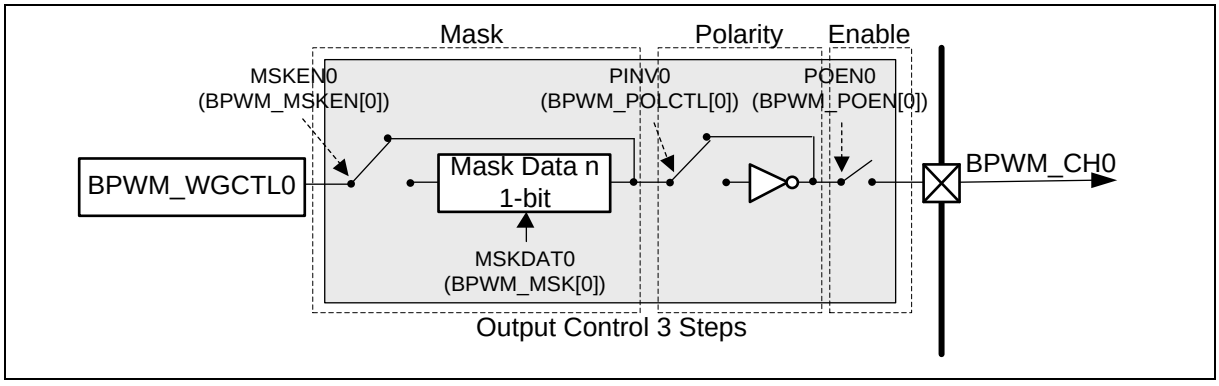


Figure 6.11-16 BPWM_CH0 Output Control 3 Steps

6.11.5.13 BPWM Mask Output Function

Each of the BPWM output channels can be manually overridden by using the appropriate bits in the BPWM Mask Enable Control Register (BPWM_MSKEN) and BPWM Masked Data Register (BPWM_MSK) to drive the BPWM channel outputs to specified logic states independent of the duty cycle comparison units. The BPWM mask bits are useful when controlling various types of Electrically Commutated Motor (ECM) like a BLDC motor. The BPWM_MSKEN register contains six bits, MSKENn(BPWM_MSKEN[5:0]) determine which BPWM channel output will be overridden, MSKENn(BPWM_MSKEN[5:0]) bits are active-high. The BPWM_MSK register contains six bits, MSKDATn(BPWM_MSK[5:0]) determine the state of the BPWM channel output when the channel is masked via the MSKDAT bits. Figure 6.11-17 shows an example of how BPWM mask control can be used for the override feature.

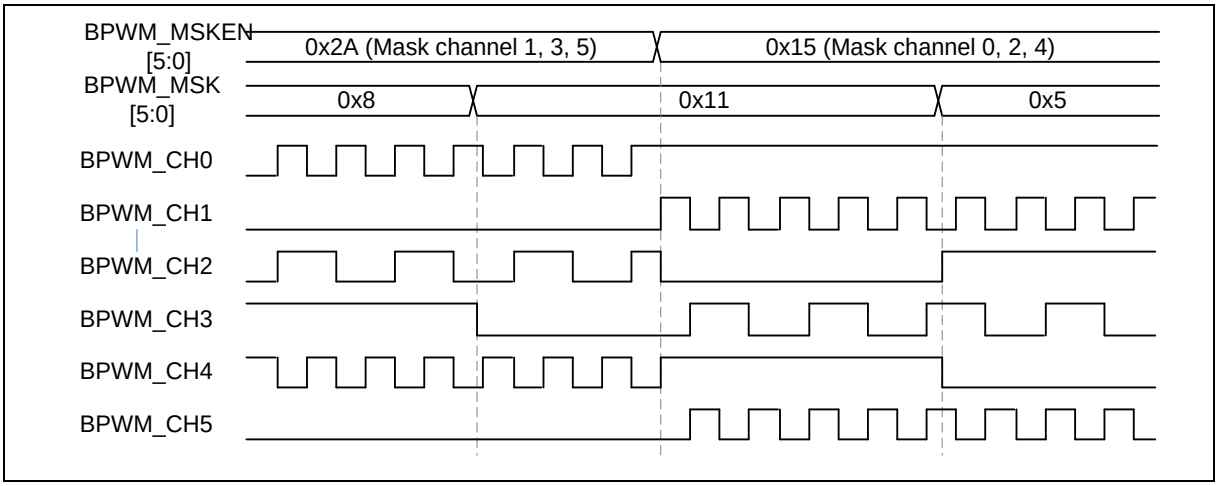
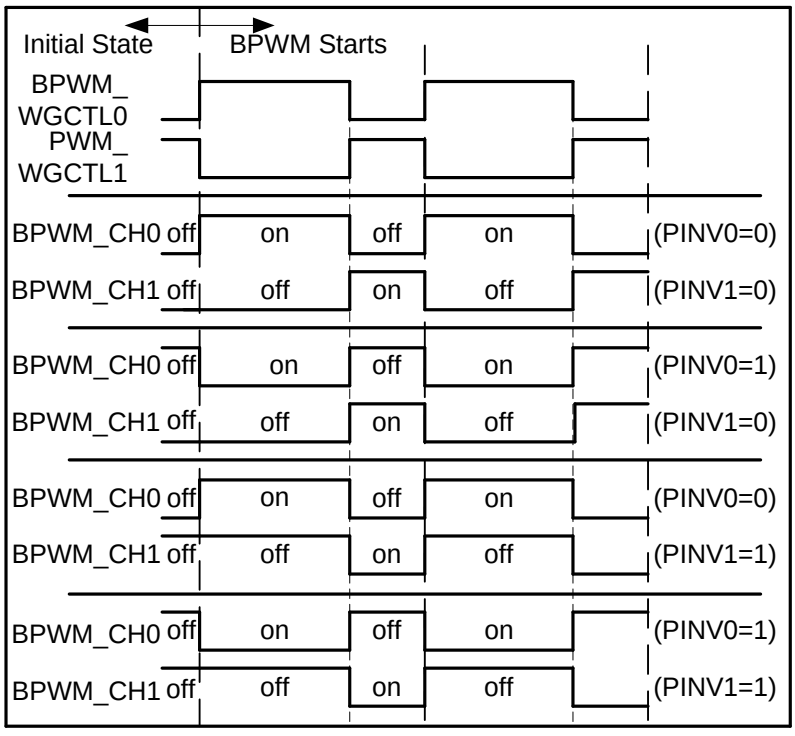


Figure 6.11-17 Mask Control Waveform Illustration

6.11.5.14 Polarity Control

Each BPWM port from BPWM_CH0 to BPWM_CH5 has an independent polarity control module to configure the polarity of the active state of BPWM output. By default, the BPWM output is active high. This implies the BPWM OFF state is low and ON state is high. This definition is variable through setting BPWM Negative Polarity Control Register (BPWM_POLCTL), for each individual BPWM channel. Figure 6.11-18 shows the initial state before BPWM starts with different polarity settings.



Note: PINVx: Negative Polarity control bits; It controls the PWM output initial state and polarity, x denotes 0 or 1.

Figure 6.11-18 Initial State and Polarity Control

6.11.5.15 BPWM Interrupt Generator

There are two independent interrupts for each BPWM as shown in Figure 6.11-19.

BPWM interrupt (BPWM_INT) comes from BPWM complementary pair events. The counter can generate the Zero point Interrupt Flag ZIF0 (BPWM_INTSTS0[0]) and the Period point Interrupt Flag PIF0 (BPWM_INTSTS0[8]). When BPWM channel n's counter equals to the comparator value stored in BPWM_CMPDATn, the different interrupt flags will be triggered depending on the counting direction. If the matching occurs at up-count direction, the Up Interrupt Flag CMPUIFn (BPWM_INTSTS0[21:16]) is set and if matching at the opposite direction, the Down Interrupt Flag CMPDIFn (BPWM_INTSTS0[29:24]) is set. If the correspond interrupt enable bits are set, the trigger events will generates interrupt signals.

Another interrupt is the capture interrupt (CAP_INT). It shares the BPWM_INT vector in NVIC, CAP_INT can be generated when the CAPRIFn (BPWM_CAPIF[5:0]) is triggered and the Capture Rising Interrupt Enable bit CAPRIENn (BPWM_CAPIEN[5:0]) is set to 1. Or in the falling edge condition, the CAPFIFn (BPWM_CAPIF[13:8]) can be triggered when the Capture Falling Interrupt Enable bit CAPFIENn (BPWM_CAPIEN[13:8]) is set to 1.

Figure 6.11-19 demonstrates the architecture of the BPWM interrupts.

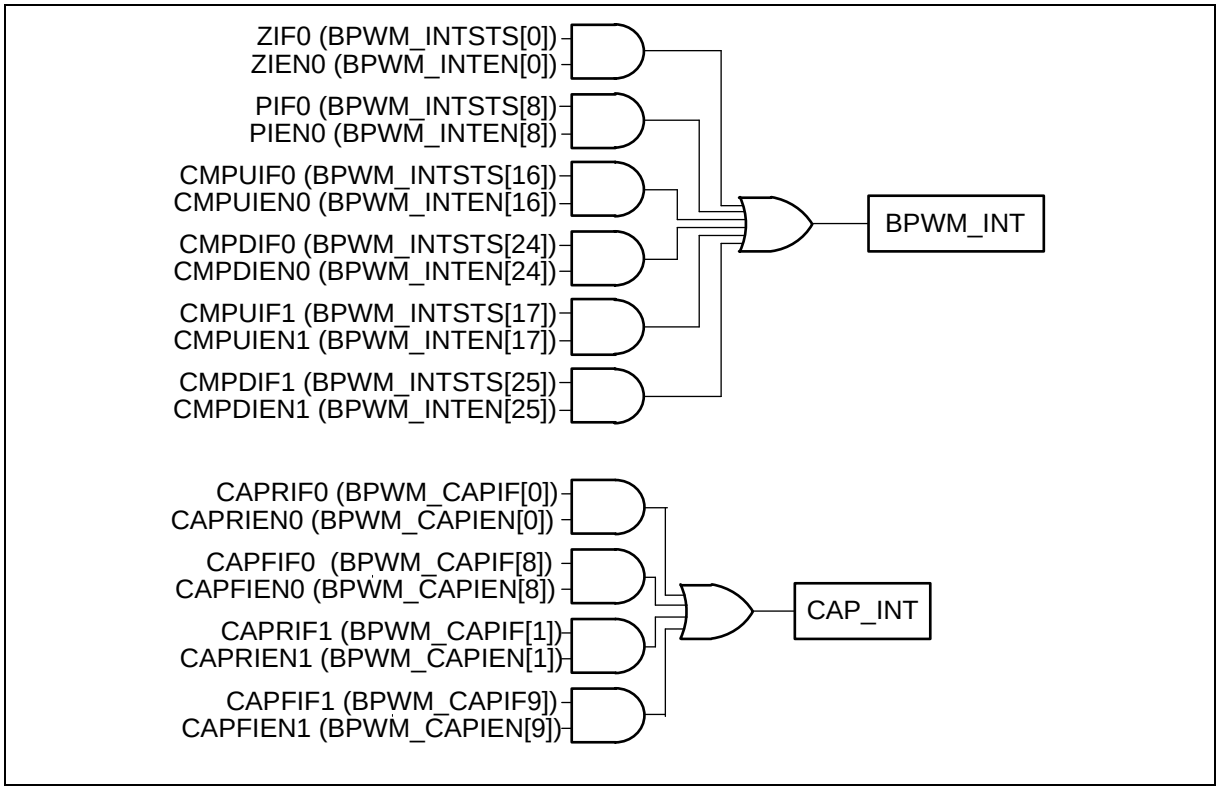


Figure 6.11-19 BPWM_CH0 and BPWM_CH1 Pair Interrupt Architecture Diagram

6.11.5.16 BPWM Trigger EADC0/1/2 Generator

BPWM can be one of the EADC0/1/2 conversion trigger source. Each BPWM pair channels share the same trigger source. Setting TRGSELn is to select the trigger sources, where TRGSELn is TRGSEL0, TRGSEL1, …, and TRGSEL5, which are located in BPWM_EADCTS0[3:0], BPWM_EADCTS0[11:8], BPWM_EADCTS0[19:16], BPWM_EADCTS0[27:24], BPWM_EADCTS1[3:0] and BPWM_EADCTS1[11:8], respectively. Setting TRGENn is to enable the trigger output to EADC0/1/2, where TRGENn is TRGEN0, TRGEN1, …, TRGEN5, which are located in BPWM_EADCTS0[7], BPWM_EADCTS0[15], BPWM_EADCTS0[23], BPWM_EADCTS0[31], BPWM_EADCTS1[7] and BPWM_EADCTS1[15], respectively. The number n (n = 0,1, ..,5) denotes BPWM channel number.

There are 7 BPWM events can be selected as the trigger source for one pair of channels. Figure 6.11-20 is an example of BPWM_CH0 and BPWM_CH1. BPWM can trigger EADC0/1/2 to start conversion in different timings by setting PERIOD and CMPDAT. Figure 6.11-22 is the trigger EADC0/1/2 timing waveform in the up-down counter type.

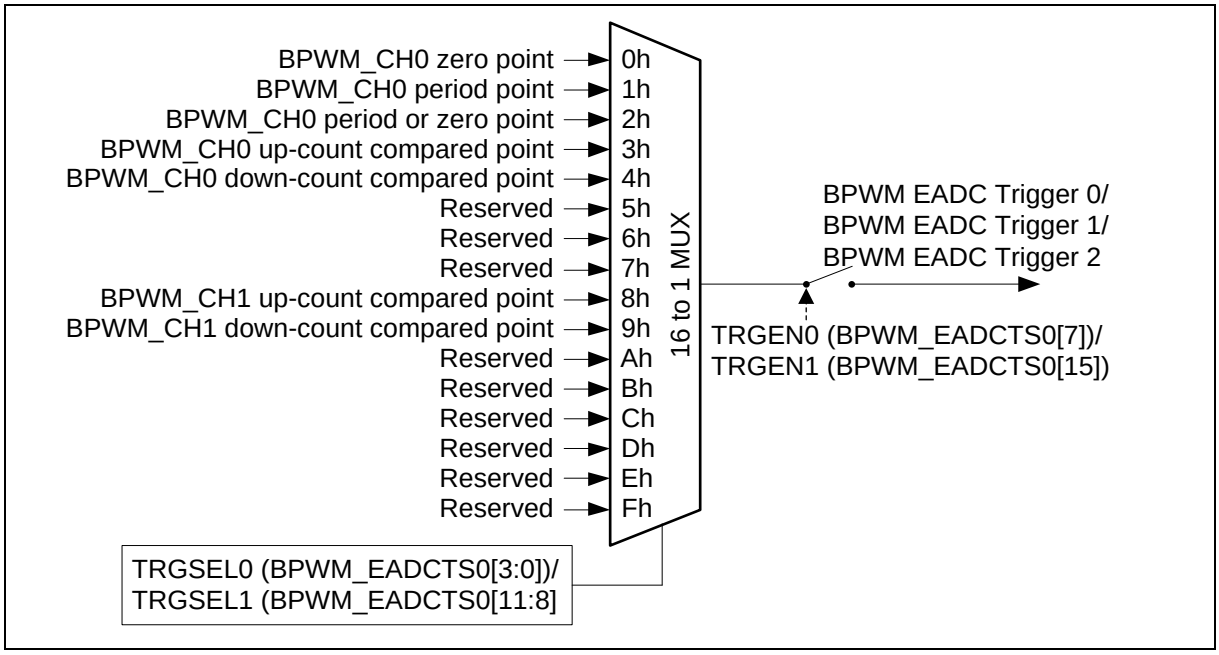


Figure 6.11-20 BPWM_CH0 and BPWM_CH1 Pair Trigger EADC0/1/2 Source Block Diagram

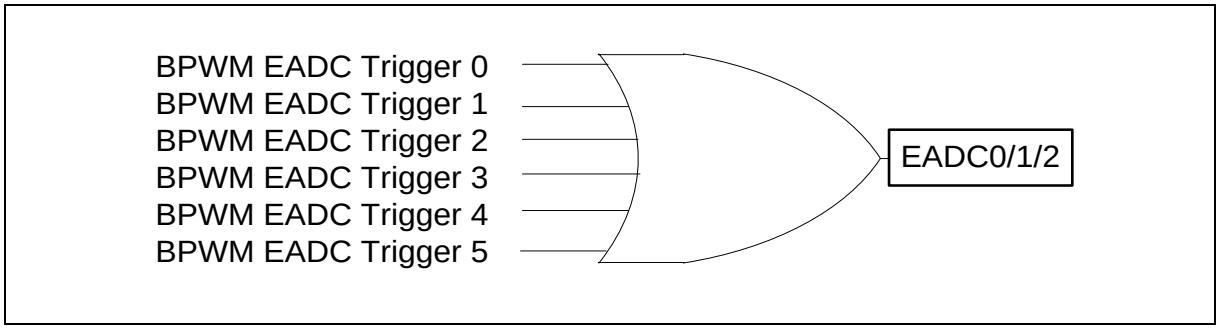


Figure 6.11-21 BPWM CH0~ CH5 Trigger EADC0/1/2 Block Diagram

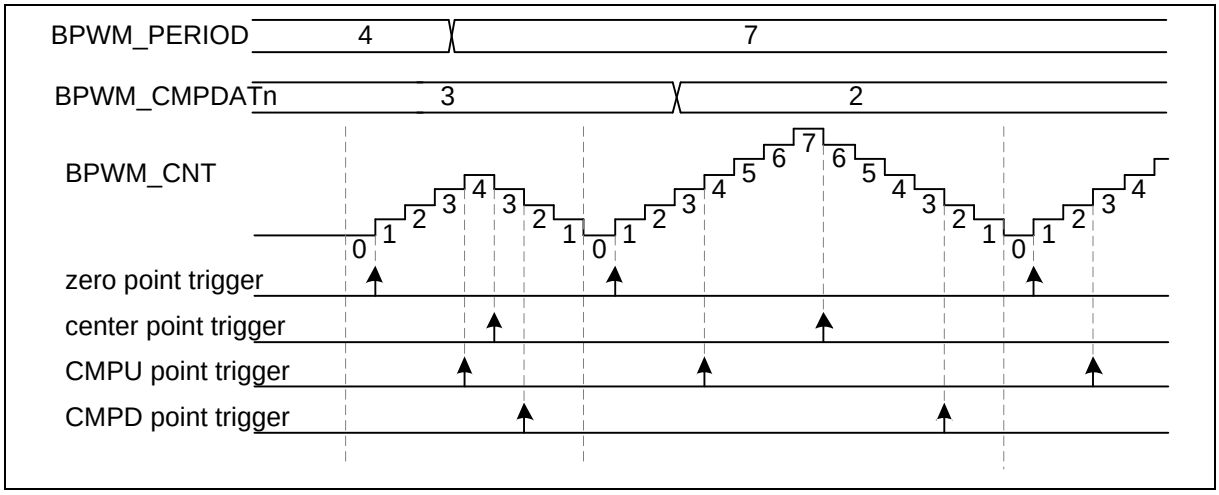


Figure 6.11-22 BPWM Trigger EADC0/1/2 in Up-Down Counter Type Timing Waveform

6.11.5.17 BPWM Trigger EADC Generator

BPWM can be one of the EADC conversion trigger source. Each BPWM pair channels share the same trigger source. Setting TRGSELn will select the trigger sources, where TRGSELn is TRGSEL0, TRGSEL1, …, and TRGSEL5, which are located in BPWM_EADCTS0[3:0], BPWM_EADCTS0[11:8], BPWM_EADCTS0[19:16], BPWM_EADCTS0[27:24], BPWM_EADCTS1[3:0] and BPWM_EADCTS1[11:8], respectively. Setting TRGENn will enable the trigger output to EADC, where TRGENn is TRGEN0, TRGEN1, …, TRGEN5, which are located in BPWM_EADCTS0[7], BPWM_EADCTS0[15], BPWM_EADCTS0[23], BPWM_EADCTS0[31], BPWM_EADCTS1[7] and BPWM_EADCTS1[15], respectively. The number n (n = 0,1, ..,5) denotes BPWM channel number.

There are 7 BPWM events that an be selected as the trigger source for one pair of channels. Figure 6.11-20 is an example of BPWM_CH0 and BPWM_CH1. BPWM can trigger EADC to start conversion in different timings by setting PERIOD and CMPDAT. Figure 6.11-22 is the trigger EADC timing waveform in the up-down counter type.

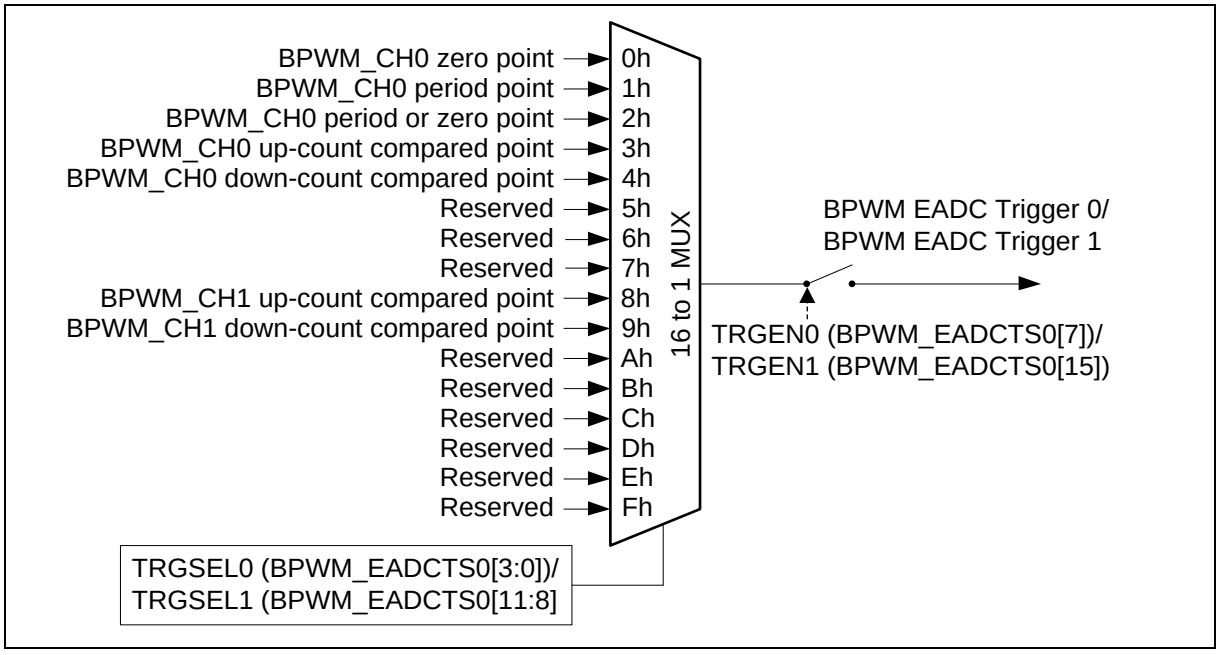


Figure 6.11-23 BPWM_CH0 and BPWM_CH1 Pair Trigger EADC Source Block Diagram

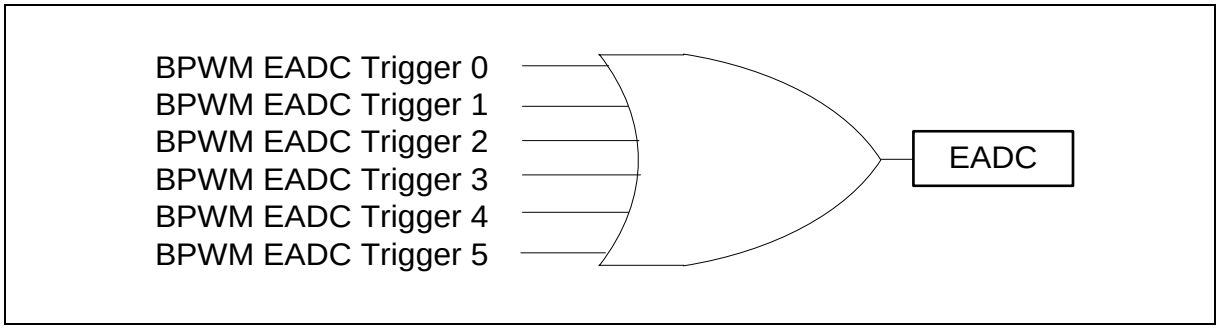


Figure 6.11-24 BPWM CH0~ CH5 Trigger EADC Block Diagram

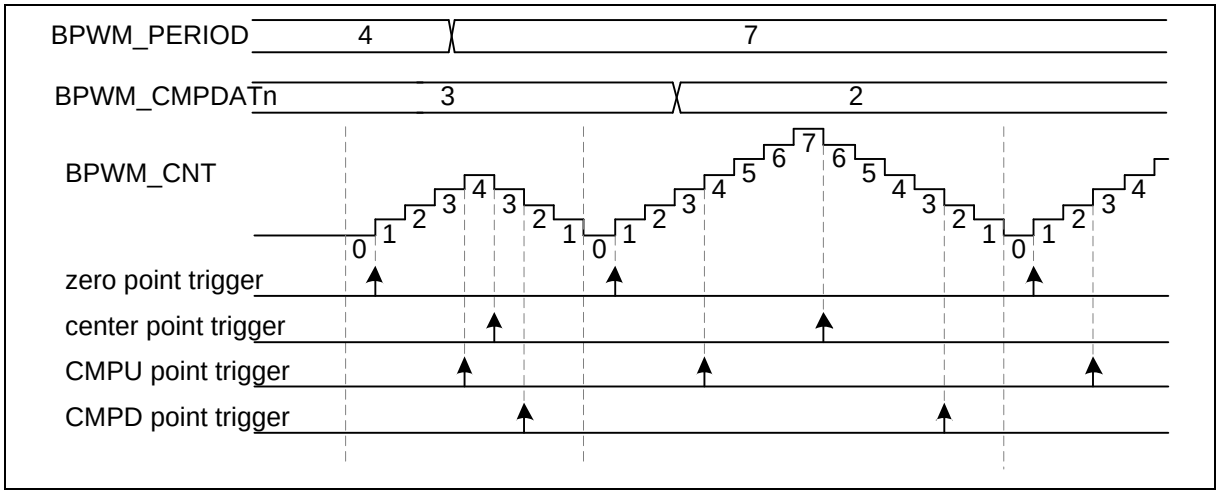


Figure 6.11-25 BPWM Trigger EADC in Up-Down Counter Type Timing Waveform

6.11.5.18 Capture Operation

The channels of the capture input and the BPWM output share the same pin and counter. The counter can operate in up or down counter type. The capture function will always latch the BPWM counter to the register RCAPDATn (BPWM_RCAPDATn[15:0]) or the register FCAPDATn (BPWM_FCAPDATn[15:0]) if the input channel has a rising transition or a falling transition, respectively. The capture function will also generate an interrupt CAP_INT (using BPWM_INT vector) if the rising or falling latch occurs and the corresponding channel n's rising or falling interrupt enable bits are set, where the CAPRIENn (BPWM_CAPIEN[5:0]) is for the rising edge and the CAPFIENn (BPWM_CAPIEN[13:8]) is for the falling edge. When rising or falling latch occurs, the corresponding BPWM counter may be reloaded with the value BPWM_PERIOD, depending on the setting of RCRLDENn or FCRLDENn (where RCRLDENn and FCRLDENn are located at BPWM_CAPCTL[21:16] and BPWM_CAPCTL[29:24], respectively). Note that the corresponding GPIO pins must be configured as the capture function by enable the CAPINENn (BPWM_CAPINEN[5:0]) for the corresponding capture channel n. Figure 6.11-26 is the capture block diagram of channel 0.

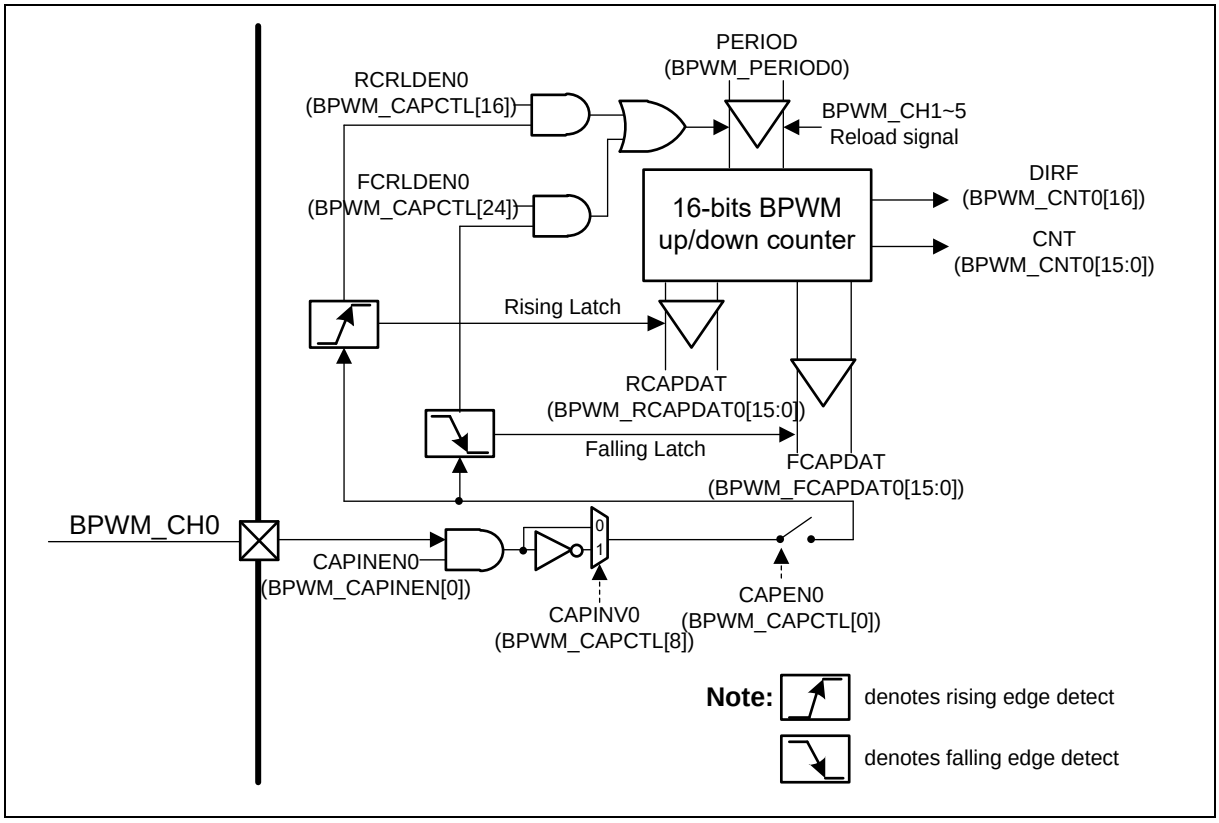


Figure 6.11-26 BPWM_CH0 Capture Block Diagram

Figure 6.11-27 illustrates the capture function timing. In this case, the capture counter is set as BPWM down counter type and the PERIOD is set to 8 so that the counter counts in the down direction, from 8 to 0. When detecting a falling edge at the capture input pin, the capture function latches counter value to the BPWM_FCAPDATn. When detecting the rising edge, it latches the counter value to the BPWM_RCAPDATn. In this timing diagram, when the falling edge is detected at the first time, the capture function will reload the counter value from the PERIOD setting because the FCRLDENn is enabled. But at the second time, the falling edge does not result in a reload because of the disabled FCRLDENn. In this example, the counter also reloads at the rising edge of the capture input because the RCRLDENn is enabled, too.

Moreover, if the case is setup as the up counter type, the counter will reload the value zero and count up to the value PERIOD. It is important that the counter is shared by all channels, so the counter reloads time also controlled by all channels' reload signals.

Figure 6.11-27 also illustrates the timing example for the interrupt and interrupt flag generation. When the rising edge at channel n is detected, the corresponding bit CAPRIFn (BPWM_CAPIF[5:0]) is set by hardware. Similarly, a falling edge detection at channel n causes the corresponding bit CAPFIFn (BPWM_CAPIF[13:8]) set by hardware. CAPRIFn (BPWM_CAPIF[5:0]) and CAPFIFn (BPWM_CAPIF[13:8]) can be cleared by software by writing '1'. If the CAPRIFn (BPWM_CAPIF[5:0]) is set and the CAPRIENn is enabled, the capture function generates an interrupt. If the CAPFIFn (BPWM_CAPIF[13:8]) is set and the CAPFIENn is enabled, the interrupt also happens.

A condition which is not shown in this figure is: if the rising latch happens again when the CAPRIFn(BPWM_CAPIF[5:0]) is already set, the Overrun status CRIFOVn (BPWM_CAPSTS[5:0]) will be set to 1 by hardware to indicate the CAPRIFn (BPWM_CAPIF[5:0]) overrunning. Also, if the falling latch happens again, the same hardware operation occurs for the interrupt flag CAPFIFn (BPWM_CAPIF[13:8]) and the Overrun status CFIFOVn (BPWM_CAPSTS[13:8]).

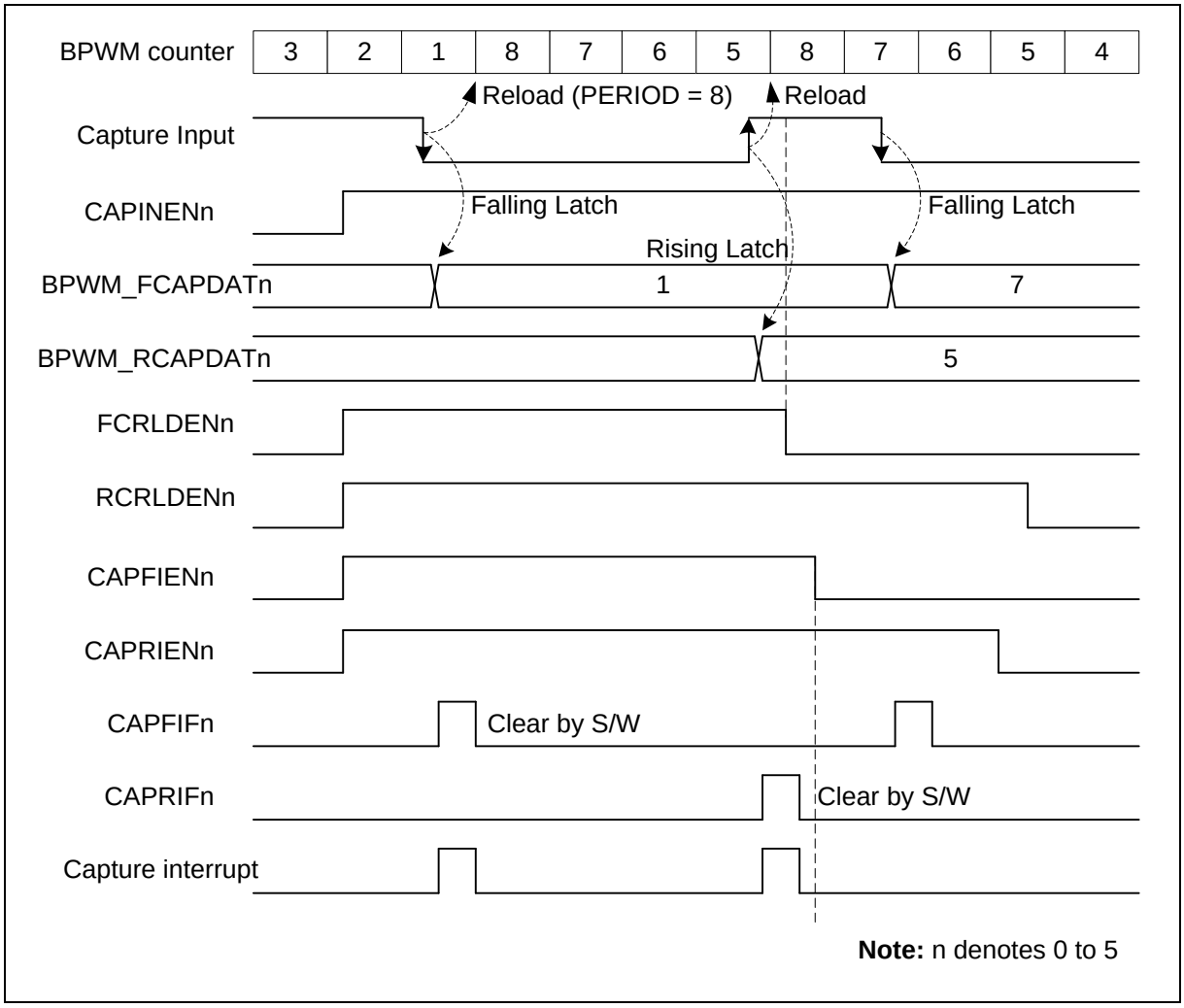


Figure 6.11-27 Capture Operation Waveform

The capture pulse width meeting the following conditions can be calculated according to the formula.

1. The capture positive or negative pulse width is shorter than a counter period.
2. The counter operates in down counter type.
3. The counter can be reloaded by both falling and rising capture events through setting FCRLDENn and RCRLDENn bits of BPWM_CAPCTL register to 1.

For the negative pulse case, the channel low pulse width is calculated as (BPWM_PERIOD + 1 - BPWM_RCAPDATn) BPWM counter time, where one BPWM counter time is (CLKPSC+1) * BPWMx_CLK clock time. In the case shown in Figure 6.11-27, low pulse width is 8+1-5 = 4 BPWM counter time.

For the positive pulse case, the channel high pulse width is calculated as (BPWM_PERIOD + 1 - BPWM_FCAPDATn) BPWM counter time, where one BPWM counter time is (CLKPSC+1) * BPWMx_CLK clock time. In the case shown in Figure 6.11-27, high pulse width is 8+1-7 = 2 BPWM counter time.

6.11.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
BPWM Base Address: **BPWM0_BA = 0x4005_A000** **BPWM1_BA = 0x4005_B000**				
BPWM_CTL0 **x=0, 1**	BPWMx_BA+0x00	R/W	BPWM Control Register 0	0x0000_0000
BPWM_CTL1 **x=0, 1**	BPWMx_BA+0x04	R/W	BPWM Control Register 1	0x0000_0000
BPWM_CLKSRC **x=0, 1**	BPWMx_BA+0x10	R/W	BPWM Clock Source Register	0x0000_0000
BPWM_CLKPSC **x=0, 1**	BPWMx_BA+0x14	R/W	BPWM Clock Prescale Register	0x0000_0000
BPWM_CNTEN **x=0, 1**	BPWMx_BA+0x20	R/W	BPWM Counter Enable Register	0x0000_0000
BPWM_CNTCLR **x=0, 1**	BPWMx_BA+0x24	R/W	BPWM Clear Counter Register	0x0000_0000
BPWM_PERIOD **x=0, 1**	BPWMx_BA+0x30	R/W	BPWM Period Register	0x0000_0000
BPWM_CMPDAT0 **x=0, 1**	BPWMx_BA+0x50	R/W	BPWM Comparator Register 0	0x0000_0000
BPWM_CMPDAT1 **x=0, 1**	BPWMx_BA+0x54	R/W	BPWM Comparator Register 1	0x0000_0000
BPWM_CMPDAT2 **x=0, 1**	BPWMx_BA+0x58	R/W	BPWM Comparator Register 2	0x0000_0000
BPWM_CMPDAT3 **x=0, 1**	BPWMx_BA+0x5C	R/W	BPWM Comparator Register 3	0x0000_0000
BPWM_CMPDAT4 **x=0, 1**	BPWMx_BA+0x60	R/W	BPWM Comparator Register 4	0x0000_0000
BPWM_CMPDAT5 **x=0, 1**	BPWMx_BA+0x64	R/W	BPWM Comparator Register 5	0x0000_0000
BPWM_CNT **x=0, 1**	BPWMx_BA+0x90	R	BPWM Counter Register	0x0000_0000
BPWM_WGCTL0 **x=0, 1**	BPWMx_BA+0xB0	R/W	BPWM Generation Register 0	0x0000_0000
BPWM_WGCTL1 **x=0, 1**	BPWMx_BA+0xB4	R/W	BPWM Generation Register 1	0x0000_0000
BPWM_MSKEN **x=0, 1**	BPWMx_BA+0xB8	R/W	BPWM Mask Enable Register	0x0000_0000
BPWM_MSK	BPWMx_BA+0xBC	R/W	BPWM Mask Data Register	0x0000_0000

x=0, 1				
BPWM_POLCTL **x=0, 1**	BPWMx_BA+0xD4	R/W	BPWM Pin Polar Inverse Register	0x0000_0000
BPWM_POEN **x=0, 1**	BPWMx_BA+0xD8	R/W	BPWM Output Enable Register	0x0000_0000
BPWM_INTEN **x=0, 1**	BPWMx_BA+0xE0	R/W	BPWM Interrupt Enable Register	0x0000_0000
BPWM_INTSTS **x=0, 1**	BPWMx_BA+0xE8	R/W	BPWM Interrupt Flag Register	0x0000_0000
BPWM_EADCTS0 **x=0, 1**	BPWMx_BA+0xF8	R/W	BPWM Trigger EADC0/1/2 Source Select Register 0	0x0000_0000
BPWM_EADCTS1 **x=0, 1**	BPWMx_BA+0xFC	R/W	BPWM Trigger EADC0/1/2 Source Select Register 1	0x0000_0000
BPWM_SSCTL **x=0, 1**	BPWMx_BA+0x110	R/W	BPWM Synchronous Start Control Register	0x0000_0000
BPWM_SSTRG **x=0, 1**	BPWMx_BA+0x114	W	BPWM Synchronous Start Trigger Register	0x0000_0000
BPWM_STATUS **x=0, 1**	BPWMx_BA+0x120	R/W	BPWM Status Register	0x0000_0000
BPWM_CAPINEN **x=0, 1**	BPWMx_BA+0x200	R/W	BPWM Capture Input Enable Register	0x0000_0000
BPWM_CAPCTL **x=0, 1**	BPWMx_BA+0x204	R/W	BPWM Capture Control Register	0x0000_0000
BPWM_CAPSTS **x=0, 1**	BPWMx_BA+0x208	R	BPWM Capture Status Register	0x0000_0000
BPWM_RCAPDAT0 **x=0, 1**	BPWMx_BA+0x20C	R	BPWM Rising Capture Data Register 0	0x0000_0000
BPWM_FCAPDAT0 **x=0, 1**	BPWMx_BA+0x210	R	BPWM Falling Capture Data Register 0	0x0000_0000
BPWM_RCAPDAT1 **x=0, 1**	BPWMx_BA+0x214	R	BPWM Rising Capture Data Register 1	0x0000_0000
BPWM_FCAPDAT1 **x=0, 1**	BPWMx_BA+0x218	R	BPWM Falling Capture Data Register 1	0x0000_0000
BPWM_RCAPDAT2 **x=0, 1**	BPWMx_BA+0x21C	R	BPWM Rising Capture Data Register 2	0x0000_0000
BPWM_FCAPDAT2 **x=0, 1**	BPWMx_BA+0x220	R	BPWM Falling Capture Data Register 2	0x0000_0000
BPWM_RCAPDAT3 **x=0, 1**	BPWMx_BA+0x224	R	BPWM Rising Capture Data Register 3	0x0000_0000
BPWM_FCAPDAT3 **x=0, 1**	BPWMx_BA+0x228	R	BPWM Falling Capture Data Register 3	0x0000_0000

BPWM_RCAPDAT4 **x=0, 1**	BPWMx_BA+0x22C	R	BPWM Rising Capture Data Register 4	0x0000_0000
BPWM_FCAPDAT4 **x=0, 1**	BPWMx_BA+0x230	R	BPWM Falling Capture Data Register 4	0x0000_0000
BPWM_RCAPDAT5 **x=0, 1**	BPWMx_BA+0x234	R	BPWM Rising Capture Data Register 5	0x0000_0000
BPWM_FCAPDAT5 **x=0, 1**	BPWMx_BA+0x238	R	BPWM Falling Capture Data Register 5	0x0000_0000
BPWM_CAPIEN **x=0, 1**	BPWMx_BA+0x250	R/W	BPWM Capture Interrupt Enable Register	0x0000_0000
BPWM_CAPIF **x=0, 1**	BPWMx_BA+0x254	R/W	BPWM Capture Interrupt Flag Register	0x0000_0000
BPWM_PBUF **x=0, 1**	BPWMx_BA+0x304	R	BPWM PERIOD Buffer	0x0000_0000
BPWM_CMPBUF0 **x=0, 1**	BPWMx_BA+0x31C	R	BPWM CMPDAT 0 Buffer	0x0000_0000
BPWM_CMPBUF1 **x=0, 1**	BPWMx_BA+0x320	R	BPWM CMPDAT 1 Buffer	0x0000_0000
BPWM_CMPBUF2 **x=0, 1**	BPWMx_BA+0x324	R	BPWM CMPDAT 2 Buffer	0x0000_0000
BPWM_CMPBUF3 **x=0, 1**	BPWMx_BA+0x328	R	BPWM CMPDAT 3 Buffer	0x0000_0000
BPWM_CMPBUF4 **x=0, 1**	BPWMx_BA+0x32C	R	BPWM CMPDAT 4 Buffer	0x0000_0000
BPWM_CMPBUF5 **x=0, 1**	BPWMx_BA+0x330	R	BPWM CMPDAT 5 Buffer	0x0000_0000

6.11.7 Register Description

BPWM Control Register 0 (BPWM_CTL0)

Register	Offset	R/W	Description	Reset Value
BPWM_CTL0	BPWMx_BA+0x00	R/W	BPWM Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
DBGTRIOFF	**DBGHALT**	**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**IMMLDEN5**	**IMMLDEN4**	**IMMLDEN3**	**IMMLDEN2**	**IMMLDEN1**	**IMMLDEN0**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CTRLD5**	**CTRLD4**	**CTRLD3**	**CTRLD2**	**CTRLD1**	**CTRLD0**

Bits	Description	
[31]	**DBGTRIOFF**	**ICE Debug Mode Acknowledge Disable (Write Protect)** 0 = ICE debug mode acknowledgement effects BPWM output. BPWM pin will be forced as tri-state while ICE debug mode acknowledged. 1 = ICE debug mode acknowledgement Disabled. BPWM pin will keep output no matter ICE debug mode acknowledged or not. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[30]	**DBGHALT**	**ICE Debug Mode Counter Halt (Write Protect)** If counter halt is enabled, BPWM all counters will keep current value until exit ICE debug mode. 0 = ICE debug mode counter halt Disabled. 1 = ICE debug mode counter halt Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[29:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**IMMLDENn**	**Immediately Load Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = PERIOD will load to PBUF at the end point of each period. CMPDAT will load to CMPBUF at the end point or center point of each period by setting CTRLD bit. 1 = PERIOD/CMPDAT will load to PBUF and CMPBUF immediately when software update PERIOD/CMPDAT. **Note:** If IMMLDENn is Enabled, CTRLDn will be invalid.
[15:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CTRLDn**	**Center Re-load** Each bit n controls the corresponding BPWM channel n. In up-down counter type, PERIOD will load to PBUF at the end point of each period. CMPDAT will load to CMPBUF at the center point of a period.

BPWM Control Register 1 (BPWM_CTL1)

Register	Offset	R/W	Description	Reset Value
BPWM_CTL1	BPWMx_BA+0x04	R/W	BPWM Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**CNTTYPE0**	

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1:0]	**CNTTYPE0**	**BPWM Counter Behavior Type 0** Each bit n controls corresponding BPWM channel n. 00 = Up counter type (supports in capture mode). 01 = Down count type (supports in capture mode). 10 = Up-down counter type. 11 = Reserved.

BPWM Clock Source Register (BPWM_CLKSRC)

Register	Offset	R/W	Description	Reset Value
BPWM_CLKSRC	BPWMx_BA+0x10	R/W	BPWM Clock Source Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**ECLKSRC0**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**ECLKSRC0**	**BPWM_CH01 External Clock Source Select** 000 = BPWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.

BPWM Clock Prescale Register (BPWM_CLKPSC)

Register	Offset	R/W	Description	Reset Value
BPWM_CLKPSC	BPWMx_BA+0x14	R/W	BPWM Clock Prescale Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CLKPSC**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLKPSC							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CLKPSC**	**BPWM Counter Clock Prescale** The clock of BPWM counter is decided by clock prescaler. Each BPWM pair share one BPWM counter clock prescaler. The clock of BPWM counter is divided by (CLKPSC+ 1).

BPWM Counter Enable Register (BPWM_CNTEN)

Register	Offset	R/W	Description	Reset Value
BPWM_CNTEN	BPWMx_BA+0x20	R/W	BPWM Counter Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTEN0**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTEN0**	**BPWM Counter 0 Enable Bit** 0 = BPWM Counter and clock prescaler stop running. 1 = BPWM Counter and clock prescaler start running.

BPWM Clear Counter Register (BPWM_CNTCLR)

Register	Offset	R/W	Description	Reset Value
BPWM_CNTCLR	BPWMx_BA+0x24	R/W	BPWM Clear Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTCLR0**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTCLR0**	**Clear BPWM Counter Control Bit 0** It is automatically cleared by hardware. 0 = No effect. 1 = Clear 16-bit BPWM counter to 0000H.

BPWM Period Register (BPWM_PERIOD)

Register	Offset	R/W	Description	Reset Value
BPWM_PERIOD	BPWMx_BA+0x30	R/W	BPWM Period Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PERIOD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PERIOD							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PERIOD**	**BPWM Period Register** Up-Count mode: In this mode, BPWM counter counts from 0 to PERIOD, and restarts from 0. Down-Count mode: In this mode, BPWM counter counts from PERIOD to 0, and restarts from PERIOD. BPWM period time = (PERIOD+1) * BPWM_CLK period. Up-Down-Count mode: In this mode, BPWM counter counts from 0 to PERIOD, then decrements to 0 and repeats again. BPWM period time = 2 * PERIOD * BPWM_CLK period.

BPWM Comparator Register 0~5 (BPWM_CMPDAT0~5)

Register	Offset	R/W	Description	Reset Value
BPWM_CMPDAT0	BPWMx_BA+0x50	R/W	BPWM Comparator Register 0	0x0000_0000
BPWM_CMPDAT1	BPWMx_BA+0x54	R/W	BPWM Comparator Register 1	0x0000_0000
BPWM_CMPDAT2	BPWMx_BA+0x58	R/W	BPWM Comparator Register 2	0x0000_0000
BPWM_CMPDAT3	BPWMx_BA+0x5C	R/W	BPWM Comparator Register 3	0x0000_0000
BPWM_CMPDAT4	BPWMx_BA+0x60	R/W	BPWM Comparator Register 4	0x0000_0000
BPWM_CMPDAT5	BPWMx_BA+0x64	R/W	BPWM Comparator Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPDAT**	**BPWM Comparator Register** CMPDAT use to compare with CNT to generate BPWM waveform, interrupt and trigger EADC0/1/2. In independent mode, CMPDAT0~5 denote as 6 independent BPWM_CH0~5 compared point.

BPWM Counter Register (BPWM_CNT)

Register	Offset	R/W	Description	Reset Value
BPWM_CNT	BPWMx_BA+0x90	R	BPWM Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**DIRF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT							

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DIRF**	**BPWM Direction Indicator Flag (Read Only)** 0 = Counter is Down count. 1 = Counter is UP count.
[15:0]	**CNT**	**BPWM Data Register (Read Only)** User can monitor CNT to know the current value in 16-bit period counter.

BPWM Generation Register 0 (BPWM_WGCTL0)

Register	Offset	R/W	Description	Reset Value
BPWM_WGCTL0	BPWMx_BA+0xB0	R/W	BPWM Generation Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**PRDPCTL5**		**PRDPCTL4**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRDPCTL3		**PRDPCTL2**		**PRDPCTL1**		**PRDPCTL0**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**ZPCTL5**		**ZPCTL4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ZPCTL3		**ZPCTL2**		**ZPCTL1**		**ZPCTL0**	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[16+2n+1:16+2n] n=0,1..5	**PRDPCTLn**	**BPWM Period or Center Point Control** Each bit n controls the corresponding BPWM channel n. 00 = Do nothing. 01 = BPWM period (center) point output Low. 10 = BPWM period (center) point output High. 11 = BPWM period (center) point output Toggle. BPWM can control output level when BPWM counter count to (PERIOD+1). **Note:** This bit is center point control when BPWM counter is operating in up-down counter type.
[15:12]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..5	**ZPCTLn**	**BPWM Zero Point Control** Each bit n controls the corresponding BPWM channel n. 00 = Do nothing. 01 = BPWM zero point output Low. 10 = BPWM zero point output High. 11 = BPWM zero point output Toggle. **Note:** BPWM can control output level when BPWM counter counts to zero.

BPWM Generation Register 1 (BPWM_WGCTL1)

Register	Offset	R/W	Description	Reset Value
BPWM_WGCTL1	BPWMx_BA+0xB4	R/W	BPWM Generation Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**CMPDCTL5**		**CMPDCTL4**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPDCTL3		**CMPDCTL2**		**CMPDCTL1**		**CMPDCTL0**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CMPUCTL5**		**CMPUCTL4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPUCTL3		**CMPUCTL2**		**CMPUCTL1**		**CMPUCTL0**	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[16+2n+1:16+2n] n=0,1..5	**CMPDCTLn**	**BPWM Compare Down Point Control** Each bit n controls the corresponding BPWM channel n. 00 = Do nothing. 01 = BPWM compare down point output Low. 10 = BPWM compare down point output High. 11 = BPWM compare down point output Toggle. **Note:** BPWM can control output level when BPWM counter down counts to CMPDAT.
[15:12]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..5	**CMPUCTLn**	**BPWM Compare Up Point Control** Each bit n controls the corresponding BPWM channel n. 00 = Do nothing. 01 = BPWM compare up point output Low. 10 = BPWM compare up point output High. 11 = BPWM compare up point output Toggle. **Note:** BPWM can control output level when BPWM counter up counts to CMPDAT.

BPWM Mask Enable Register (BPWM_MSKEN)

Register	Offset	R/W	Description	Reset Value
BPWM_MSKEN	BPWMx_BA+0xB8	R/W	BPWM Mask Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MSKEN5**	**MSKEN4**	**MSKEN3**	**MSKEN2**	**MSKEN1**	**MSKEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**MSKENn**	**BPWM Mask Enable Bits** Each bit n controls the corresponding BPWM channel n. The BPWM output signal will be masked when this bit is enabled. The corresponding BPWM channel n will output MSKDATn (BPWM_MSK[5:0]) data. 0 = BPWM output signal is non-masked. 1 = BPWM output signal is masked and output MSKDATn data.

BPWM Mask DATA Register (BPWM_MSK)

Register	Offset	R/W	Description	Reset Value
BPWM_MSK	BPWMx_BA+0xBC	R/W	BPWM Mask Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MSKDAT5**	**MSKDAT4**	**MSKDAT3**	**MSKDAT2**	**MSKDAT1**	**MSKDAT0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**MSKDATn**	**BPWM Mask Data Bit** This data bit control the state of BPWMn output pin, if corresponding mask function is enabled. Each bit n controls the corresponding BPWM channel n. 0 = Output logic low to BPWMn. 1 = Output logic high to BPWMn.

BPWM Pin Polar Inverse Control (BPWM_POLCTL)

Register	Offset	R/W	Description	Reset Value
BPWM_POLCTL	BPWMx_BA+0xD4	R/W	BPWM Pin Polar Inverse Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**PINV5**	**PINV4**	**PINV3**	**PINV2**	**PINV1**	**PINV0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**PINVn**	**BPWM PIN Polar Inverse Control** The register controls polarity state of BPWM output. Each bit n controls the corresponding BPWM channel n. 0 = BPWM output polar inverse Disabled. 1 = BPWM output polar inverse Enabled.

BPWM Output Enable Register (BPWM_POEN)

Register	Offset	R/W	Description	Reset Value
BPWM_POEN	BPWMx_BA+0xD8	R/W	BPWM Output Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**POEN5**	**POEN4**	**POEN3**	**POEN2**	**POEN1**	**POEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**POENn**	**BPWM Pin Output Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = BPWM pin at tri-state. 1 = BPWM pin in output mode.

BPWM Interrupt Enable Register (BPWM_INTEN)

Register	Offset	R/W	Description	Reset Value
BPWM_INTEN	BPWMx_BA+0xE0	R/W	BPWM Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIEN5	CMPDIEN4	CMPDIEN3	CMPDIEN2	CMPDIEN1	CMPDIEN0
23	22	21	20	19	18	17	16
Reserved		CMPUIEN5	CMPUIEN4	CMPUIEN3	CMPUIEN2	CMPUIEN1	CMPUIEN0
15	14	13	12	11	10	9	8
Reserved							PIEN0
7	6	5	4	3	2	1	0
Reserved							ZIEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CMPDIENn**	**BPWM Compare Down Count Interrupt Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Compare down count interrupt Disabled. 1 = Compare down count interrupt Enabled.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CMPUIENn**	**BPWM Compare Up Count Interrupt Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Compare up count interrupt Disabled. 1 = Compare up count interrupt Enabled.
[15:9]	**Reserved**	Reserved.
[8]	**PIEN0**	**BPWM Period Point Interrupt 0 Enable Bit** 0 = Period point interrupt Disabled. 1 = Period point interrupt Enabled. **Note:** Up-down counter type period point means center point.
[7:1]	**Reserved**	Reserved.
[0]	**ZIEN0**	**BPWM Zero Point Interrupt 0 Enable Bit** 0 = Zero point interrupt Disabled. 1 = Zero point interrupt Enabled.

BPWM Interrupt Flag Register (BPWM_INTSTS)

Register	Offset	R/W	Description	Reset Value
BPWM_INTSTS	BPWMx_BA+0xE8	R/W	BPWM Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIF5	CMPDIF4	CMPDIF3	CMPDIF2	CMPDIF1	CMPDIF0
23	22	21	20	19	18	17	16
Reserved		CMPUIF5	CMPUIF4	CMPUIF3	CMPUIF2	CMPUIF1	CMPUIF0
15	14	13	12	11	10	9	8
Reserved							PIF0
7	6	5	4	3	2	1	0
Reserved							ZIF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CMPDIFn**	**BPWM Compare Down Count Interrupt Flag** Each bit n controls the corresponding BPWM channel n. Flag is set by hardware when BPWM counter down count and reaches BPWM_CMPDATn. **Note:** If CMPDAT is equal to PERIOD, this flag does not work in down counter type selection. This bit is cleared by writing 1 to it.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CMPUIFn**	**BPWM Compare Up Count Interrupt Flag** Flag is set by hardware when BPWM counter up count and reaches BPWM_CMPDATn. Each bit n controls the corresponding BPWM channel n. **Note:** If CMPDAT is equal to PERIOD, this flag does not work in up counter type selection. This bit is cleared by writing 1 to it.
[15:9]	**Reserved**	Reserved.
[8]	**PIF0**	**BPWM Period Point Interrupt Flag 0** This bit is set by hardware when BPWM_CH0 counter reaches BPWM_PERIOD0. **Note:** This bit is cleared by writing 1 to it.
[7:1]	**Reserved**	Reserved.
[0]	**ZIF0**	**BPWM Zero Point Interrupt Flag 0** This bit is set by hardware when BPWM_CH0 counter reaches 0. **Note:** This bit is cleared by writing 1 to it.

BPWM Trigger EADC0/1/2 Source Select Register 0 (BPWM_EADCTS0)

Register	Offset	R/W	Description	Reset Value
BPWM_EADCTS0	BPWMx_BA+0xF8	R/W	BPWM Trigger EADC0/1/2 Source Select Register 0	0x0000_0000

31	30	29	28	27	26	25	24
TRGEN3	Reserved			TRGSEL3			
23	22	21	20	19	18	17	16
TRGEN2	Reserved			TRGSEL2			
15	14	13	12	11	10	9	8
TRGEN1	Reserved			TRGSEL1			
7	6	5	4	3	2	1	0
TRGEN0	Reserved			TRGSEL0			

Bits	Description	
[31]	**TRGEN3**	BPWM_CH3 Trigger EADC0/1/2 Enable Bit
[30:28]	**Reserved**	Reserved.
[27:24]	**TRGSEL3**	**BPWM_CH3 Trigger EADC0/1/2 Source Select** 0000 = BPWM_CH2 zero point. 0001 = BPWM_CH2 period point. 0010 = BPWM_CH2 zero or period point. 0011 = BPWM_CH2 up-count CMPDAT point. 0100 = BPWM_CH2 down-count CMPDAT point. 0101 = Reserved. 0110 = Reserved. 0111 = Reserved. 1000 = BPWM_CH3 up-count CMPDAT point. 1001 = BPWM_CH3 down-count CMPDAT point. Others reserved.
[23]	**TRGEN2**	BPWM_CH2 Trigger EADC0/1/2 Enable Bit
[22:20]	**Reserved**	Reserved.
[19:16]	**TRGSEL2**	**BPWM_CH2 Trigger EADC0/1/2 Source Select** 0000 = BPWM_CH2 zero point. 0001 = BPWM_CH2 period point. 0010 = BPWM_CH2 zero or period point. 0011 = BPWM_CH2 up-count CMPDAT point. 0100 = BPWM_CH2 down-count CMPDAT point. 0101 = Reserved. 0110 = Reserved. 0111 = Reserved. 1000 = BPWM_CH3 up-count CMPDAT point. 1001 = BPWM_CH3 down-count CMPDAT point.

		Others reserved
[15]	**TRGEN1**	BPWM_CH1 Trigger EADC0/1/2 Enable Bit
[14:12]	**Reserved**	Reserved.
[11:8]	**TRGSEL1**	**BPWM_CH1 Trigger EADC0/1/2 Source Select** 0000 = BPWM_CH0 zero point. 0001 = BPWM_CH0 period point. 0010 = BPWM_CH0 zero or period point. 0011 = BPWM_CH0 up-count CMPDAT point. 0100 = BPWM_CH0 down-count CMPDAT point. 0101 = Reserved. 0110 = Reserved. 0111 = Reserved. 1000 = BPWM_CH1 up-count CMPDAT point. 1001 = BPWM_CH1 down-count CMPDAT point. Others reserved
[7]	**TRGEN0**	BPWM_CH0 Trigger EADC0/1/2 Enable Bit
[6:4]	**Reserved**	Reserved.
[3:0]	**TRGSEL0**	**BPWM_CH0 Trigger EADC0/1/2 Source Select** 0000 = BPWM_CH0 zero point. 0001 = BPWM_CH0 period point. 0010 = BPWM_CH0 zero or period point. 0011 = BPWM_CH0 up-count CMPDAT point. 0100 = BPWM_CH0 down-count CMPDAT point. 0101 = Reserved. 0110 = Reserved. 0111 = Reserved. 1000 = BPWM_CH1 up-count CMPDAT point. 1001 = BPWM_CH1 down-count CMPDAT point. Others reserved

BPWM Trigger EADC0/1/2 Source Select Register 1 (BPWM_EADCTS1)

Register	Offset	R/W	Description	Reset Value
BPWM_EADCTS1	BPWMx_BA+0xFC	R/W	BPWM Trigger EADC0/1/2 Source Select Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TRGEN5	Reserved			TRGSEL5			
7	6	5	4	3	2	1	0
TRGEN4	Reserved			TRGSEL4			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**TRGEN5**	BPWM_CH5 Trigger EADC0/1/2 Enable Bit
[14:12]	**Reserved**	Reserved.
[11:8]	**TRGSEL5**	**BPWM_CH5 Trigger EADC0/1/2 Source Select** 0000 = BPWM_CH4 zero point. 0001 = BPWM_CH4 period point. 0010 = BPWM_CH4 zero or period point. 0011 = BPWM_CH4 up-count CMPDAT point. 0100 = BPWM_CH4 down-count CMPDAT point. 0101 = Reserved. 0110 = Reserved. 0111 = Reserved. 1000 = BPWM_CH5 up-count CMPDAT point. 1001 = BPWM_CH5 down-count CMPDAT point. Others reserved
[7]	**TRGEN4**	BPWM_CH4 Trigger EADC0/1/2 Enable Bit
[6:4]	**Reserved**	Reserved.
[3:0]	**TRGSEL4**	**BPWM_CH4 Trigger EADC0/1/2 Source Select** 0000 = BPWM_CH4 zero point. 0001 = BPWM_CH4 period point. 0010 = BPWM_CH4 zero or period point. 0011 = BPWM_CH4 up-count CMPDAT point. 0100 = BPWM_CH4 down-count CMPDAT point. 0101 = Reserved. 0110 = Reserved. 0111 = Reserved.

		1000 = BPWM_CH5 up-count CMPDAT point. 1001 = BPWM_CH5 down-count CMPDAT point. Others reserved

BPWM Synchronous Start Control Register (BPWM_SSCTL)

Register	Offset	R/W	Description	Reset Value
BPWM_SSCTL	BPWMx_BA+0x110	R/W	BPWM Synchronous Start Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**SSRC**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**SSEN0**

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:8]	**SSRC**	**BPWM Synchronous Start Source Select** 00 = Synchronous start source come from EPWM0. 01 = Synchronous start source come from EPWM1. 10 = Synchronous start source come from BPWM0. 11 = Synchronous start source come from BPWM1.
[7:1]	**Reserved**	Reserved.
[0]	**SSEN0**	**BPWM Synchronous Start Function 0 Enable Bit** When synchronous start function is enabled, the BPWM_CH0 counter enable bit (CNTEN0) can be enabled by writing BPWM synchronous start trigger bit (CNTSEN). 0 = BPWM synchronous start function Disabled. 1 = BPWM synchronous start function Enabled.

BPWM Synchronous Start Trigger Register (BPWM_SSTRG)

Register	Offset	R/W	Description	Reset Value
BPWM_SSTRG	BPWMx_BA+0x114	W	BPWM Synchronous Start Trigger Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTSEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTSEN**	**BPWM Counter Synchronous Start Enable Bit (Write Only)** BPMW counter synchronous enable function is used to make EPWM or BPWM channels start counting at the same time. Writing this bit to 1 will also set the counter enable bit if correlated BPWM channel counter synchronous start function is enabled.

BPWM Status Register (BPWM_STATUS)

Register	Offset	R/W	Description	Reset Value
BPWM_STATUS	BPWMx_BA+0x120	R/W	BPWM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		EADCTRG5	EADCTRG4	EADCTRG3	EADCTRG2	EADCTRG1	EADCTRG0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							CNTMAXF0

Bits	Description	
[31:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**EADCTRGn**	**EADC0 & EADC1 & EADC2 Start of Conversion Status** Each bit n controls the corresponding BPWM channel n. 0 = No EADC0/1/2 start of conversion trigger event has occurred. 1 = An EADC0/1/2 start of conversion trigger event has occurred. **Note:** This bit is cleared by writing 1 to it.
[15:1]	**Reserved**	Reserved.
[0]	**CNTMAXF0**	**Time-base Counter 0 Equal to 0xFFFF Latched Flag** 0 = The time-base counter never reached its maximum value 0xFFFF. 1 = The time-base counter reached its maximum value. **Note:** This bit is cleared by writing 1 to it.

BPWM Capture Input Enable Register (BPWM_CAPINEN)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPINEN	BPWMx_BA+0x200	R/W	BPWM Capture Input Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**Reserved**	**CAPINEN5**	**CAPINEN4**	**CAPINEN3**	**CAPINEN2**	**CAPINEN1**	**CAPINEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPINENn**	**Capture Input Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = BPWM Channel capture input path Disabled. The input of BPWM channel capture function is always regarded as 0. 1 = BPWM Channel capture input path Enabled. The input of BPWM channel capture function comes from correlative multifunction pin.

BPWM Capture Control Register (BPWM_CAPCTL)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPCTL	BPWMx_BA+0x204	R/W	BPWM Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		FCRLDEN5	FCRLDEN4	FCRLDEN3	FCRLDEN2	FCRLDEN1	FCRLDEN0
23	22	21	20	19	18	17	16
Reserved		RCRLDEN5	RCRLDEN4	RCRLDEN3	RCRLDEN2	RCRLDEN1	RCRLDEN0
15	14	13	12	11	10	9	8
Reserved		CAPINV5	CAPINV4	CAPINV3	CAPINV2	CAPINV1	CAPINV0
7	6	5	4	3	2	1	0
Reserved		CAPEN5	CAPEN4	CAPEN3	CAPEN2	CAPEN1	CAPEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**FCRLDENn**	**Falling Capture Reload Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Falling capture reload counter Disabled. 1 = Falling capture reload counter Enabled.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**RCRLDENn**	**Rising Capture Reload Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Rising capture reload counter Disabled. 1 = Rising capture reload counter Enabled.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CAPINVn**	**Capture Inverter Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Capture source inverter Disabled. 1 = Capture source inverter Enabled. Reverse the input signal from GPIO.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPENn**	**Capture Function Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Capture function Disabled. RCAPDAT/FCAPDAT register will not be updated. 1 = Capture function Enabled. Capture latched the BPWM counter value when detected rising or falling edge of input signal and saved to RCAPDAT (Rising latch) and FCAPDAT (Falling latch).

BPWM Capture Status Register (BPWM_CAPSTS)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPSTS	BPWMx_BA+0x208	R	BPWM Capture Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CFIFOV5**	**CFIFOV4**	**CFIFOV3**	**CFIFOV2**	**CFIFOV1**	**CFIFOV0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CRIFOV5**	**CRIFOV4**	**CRIFOV3**	**CRIFOV2**	**CRIFOV1**	**CRIFOV0**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CFIFOVn**	**Capture Falling Interrupt Flag Overrun Status (Read Only)** This flag indicates if falling latch happened when the corresponding CAPFIF is 1. Each bit n controls the corresponding BPWM channel n. **Note:** This bit will be cleared automatically when user clears corresponding CAPFIF.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CRIFOVn**	**Capture Rising Interrupt Flag Overrun Status (Read Only)** This flag indicates if rising latch happened when the corresponding CAPRIF is 1. Each bit n controls the corresponding BPWM channel n. **Note:** This bit will be cleared automatically when user clears corresponding CAPRIF.

BPWM Rising Capture Data Register 0~5 (BPWM_RCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
BPWM_RCAPDAT0	BPWMx_BA+0x20C	R	BPWM Rising Capture Data Register 0	0x0000_0000
BPWM_RCAPDAT1	BPWMx_BA+0x214	R	BPWM Rising Capture Data Register 1	0x0000_0000
BPWM_RCAPDAT2	BPWMx_BA+0x21C	R	BPWM Rising Capture Data Register 2	0x0000_0000
BPWM_RCAPDAT3	BPWMx_BA+0x224	R	BPWM Rising Capture Data Register 3	0x0000_0000
BPWM_RCAPDAT4	BPWMx_BA+0x22C	R	BPWM Rising Capture Data Register 4	0x0000_0000
BPWM_RCAPDAT5	BPWMx_BA+0x234	R	BPWM Rising Capture Data Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RCAPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RCAPDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RCAPDAT**	**BPWM Rising Capture Data (Read Only)** When rising capture condition happened, the BPWM counter value will be saved in this register.

BPWM Falling Capture Data Register 0~5 (BPWM_FCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
BPWM_FCAPDAT0	BPWMx_BA+0x210	R	BPWM Falling Capture Data Register 0	0x0000_0000
BPWM_FCAPDAT1	BPWMx_BA+0x218	R	BPWM Falling Capture Data Register 1	0x0000_0000
BPWM_FCAPDAT2	BPWMx_BA+0x220	R	BPWM Falling Capture Data Register 2	0x0000_0000
BPWM_FCAPDAT3	BPWMx_BA+0x228	R	BPWM Falling Capture Data Register 3	0x0000_0000
BPWM_FCAPDAT4	BPWMx_BA+0x230	R	BPWM Falling Capture Data Register 4	0x0000_0000
BPWM_FCAPDAT5	BPWMx_BA+0x238	R	BPWM Falling Capture Data Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FCAPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCAPDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FCAPDAT**	**BPWM Falling Capture Data (Read Only)** When falling capture condition happened, the BPWM counter value will be saved in this register.

BPWM Capture Interrupt Enable Register (BPWM_CAPIEN)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPIEN	BPWMx_BA+0x250	R/W	BPWM Capture Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CAPFIEN5	CAPFIEN4	CAPFIEN3	CAPFIEN2	CAPFIEN1	CAPFIEN0
7	6	5	4	3	2	1	0
Reserved		CAPRIEN5	CAPRIEN4	CAPRIEN3	CAPRIEN2	CAPRIEN1	CAPRIEN0

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:8]	**CAPFIENn**	**BPWM Capture Falling Latch Interrupt Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Capture falling edge latch interrupt Disabled. 1 = Capture falling edge latch interrupt Enabled.
[7:6]	**Reserved**	Reserved.
[5:0]	**CAPRIENn**	**BPWM Capture Rising Latch Interrupt Enable Bits** Each bit n controls the corresponding BPWM channel n. 0 = Capture rising edge latch interrupt Disabled. 1 = Capture rising edge latch interrupt Enabled.

BPWM Capture Interrupt Flag Register (BPWM_CAPIF)

Register	Offset	R/W	Description	Reset Value
BPWM_CAPIF	BPWMx_BA+0x254	R/W	BPWM Capture Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CAPFIF5	CAPFIF4	CAPFIF3	CAPFIF2	CAPFIF1	CAPFIF0
7	6	5	4	3	2	1	0
Reserved		CAPRIF5	CAPRIF4	CAPRIF3	CAPRIF2	CAPRIF1	CAPRIF0

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CAPFIFn**	**BPWM Capture Falling Latch Interrupt Flag** Each bit n controls the corresponding BPWM channel n. 0 = No capture falling latch condition happened. 1 = Capture falling latch condition happened, this flag will be set to high. **Note:** This bit is cleared by writing 1 to it.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPRIFn**	**BPWM Capture Rising Latch Interrupt Flag** Each bit n controls the corresponding BPWM channel n. 0 = No capture rising latch condition happened. 1 = Capture rising latch condition happened, this flag will be set to high. **Note:** This bit is cleared by writing 1 to it.

BPWM Period Register Buffer (BPWM_PBUF)

Register	Offset	R/W	Description	Reset Value
BPWM_PBUF	BPWMx_BA+0x304	R	BPWM PERIOD Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PBUF**	**BPWM Period Buffer (Read Only)** Used as PERIOD active register.

BPWM Comparator Register Buffer 0~5 (BPWM_CMPBUF0~5)

Register	Offset	R/W	Description	Reset Value
BPWM_CMPBUF0	BPWMx_BA+0x31C	R	BPWM CMPDAT 0 Buffer	0x0000_0000
BPWM_CMPBUF1	BPWMx_BA+0x320	R	BPWM CMPDAT 1 Buffer	0x0000_0000
BPWM_CMPBUF2	BPWMx_BA+0x324	R	BPWM CMPDAT 2 Buffer	0x0000_0000
BPWM_CMPBUF3	BPWMx_BA+0x328	R	BPWM CMPDAT 3 Buffer	0x0000_0000
BPWM_CMPBUF4	BPWMx_BA+0x32C	R	BPWM CMPDAT 4 Buffer	0x0000_0000
BPWM_CMPBUF5	BPWMx_BA+0x330	R	BPWM CMPDAT 5 Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPBUF**	**BPWM Comparator Buffer (Read Only)** Used as CMP active register.

6.12 EPWM Generator and Capture Timer (EPWM)

6.12.1 Overview

The chip provides two EPWM generators — EPWM0 and EPWM1. Each EPWM supports 6 channels of EPWM output or input capture. There is a 12-bit prescaler to support flexible clock to the 16-bit EPWM counter with 16-bit comparator. The EPWM counter supports up, down and up-down counter types. EPWM uses comparator compared with counter to generate events. These events use to generate EPWM pulse, interrupt and trigger signal for EADC/DAC to start conversion.

The EPWM generator supports two standard EPWM output modes: Independent mode and Complementary mode, they have difference architecture. There are two output functions based on standard output modes: Group function and Synchronous function. Group function can be enabled under Independent mode or complementary mode. Synchronous function only enabled under complementary mode. Complementary mode has two comparators to generate various EPWM pulse with 12-bit dead-time generator and another free trigger comparator to generate trigger signal for EADC. For EPWM output control unit, it supports polarity output, independent pin mask and brake functions.

The EPWM generator also supports input capture function. It supports latch EPWM counter value to corresponding register when input channel has a rising transition, falling transition or both transition is happened. Capture function also support PDMA to transfer captured data to memory.

6.12.2 Features

6.12.2.1 EPWM Function Features

- Supports maximum clock frequency up to maximum PLL frequency
- Supports up to two EPWM modules, each module provides 6 output channels
- Supports independent mode for EPWM output/Capture input channel
- Supports complementary mode for 3 complementary paired EPWM output channel
  - Dead-time insertion independent control with 12-bit resolution
  - Synchronous function for phase control
  - Two compared values during one period
- Supports 12-bit prescaler from 1 to 4096
- Supports 16-bit resolution EPWM counter
  - Up, down and up/down counter operation type
- Supports one-shot or auto-reload counter operation mode
- Supports group function
- Supports synchronous function
- Supports mask function and tri-state enable for each EPWM pin
- Supports brake function
  - Brake source from pin, analog comparator and system safety events (clock failed, SRAM parity error, Brown-out detection and CPU lockup).
  - Noise filter for brake source from pin
  - Leading edge blanking (LEB) function for brake source from analog comparator
  - Edge detect brake source to control brake state until brake interrupt cleared
  - Level detect brake source to auto recover function after brake condition removed
- Supports interrupt on the following events:

- EPWM counter matches 0, period value or compared value
- Brake condition happened

- Supports trigger EADC/DAC on the following events:
  - EPWM counter matches 0, period value or compared value
  - EPWM counter matches free trigger comparator compared value (only for EADC)
  - Supports EPWM trigger EADC event prescaler feature
- Supports PDMA transfer for Interrupt Flag Accumulator Function
- Supports EPWM output accumulator stop counter mode
- Supports Fault Detect function
- Supports External Pin Trigger Function

6.12.2.2 Capture Function Features

- Supports 3-bit capture input noise filter
- Supports up to 12 capture input channels with 16-bit resolution
- Supports rising or falling capture condition
- Supports input rising/falling capture interrupt
- Supports rising/falling capture with counter reload option
- Supports PDMA transfer function for EPWM all channels

6.12.3 Block Diagram

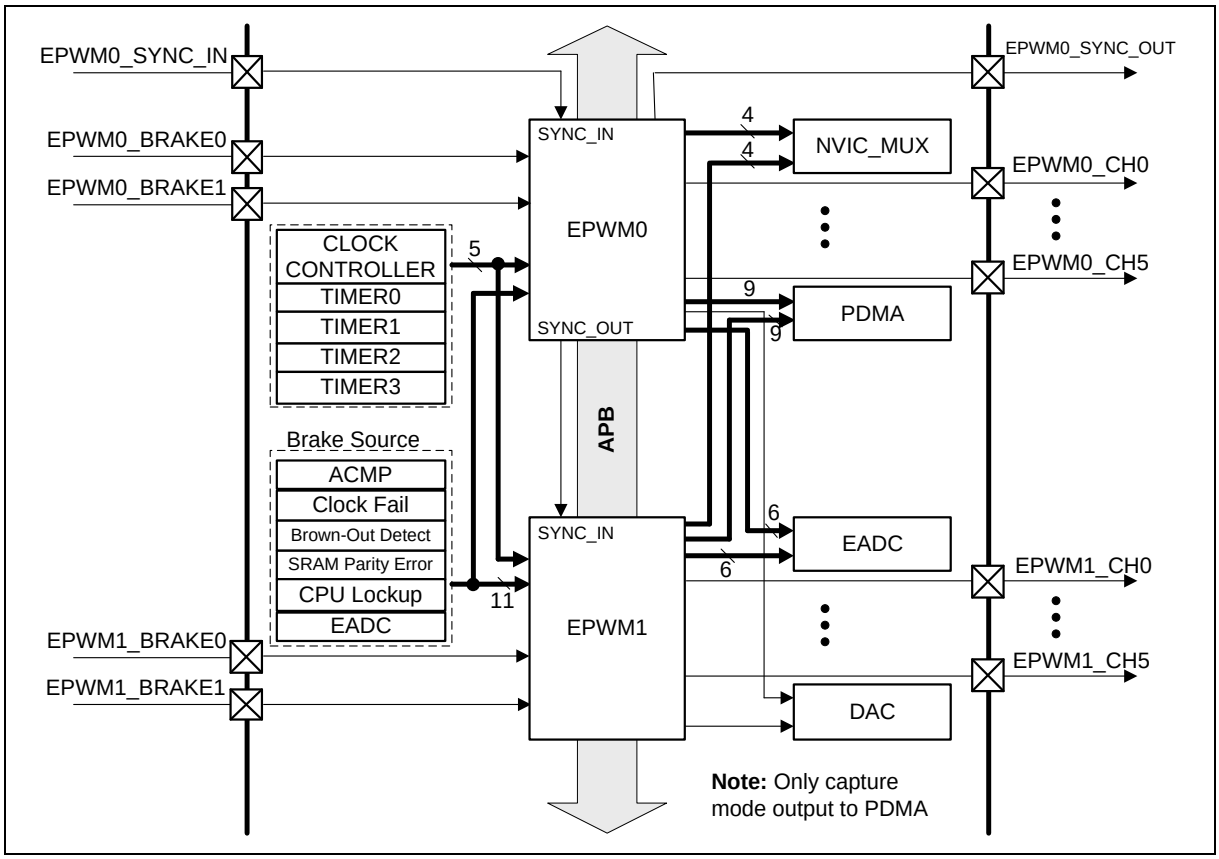


Figure 6.12-1 EPWM Generator Overview Block Diagram

EPWM Clock frequency can be set equal or double to PCLK frequency as Figure 6.12-2. For the detailed register setting, please refer to Table 6.12-1. Each EPWM generator has three clock source inputs, each clock source can be selected from EPWM Clock or four TIMER trigger EPWM outputs as Figure 6.12-3 by ECLKSRC0 (EPWM_CLKSRC[2:0]) for EPWM_CLK0, ECLKSRC2 (EPWM_CLKSRC[10:8]) for EPWM_CLK2 and ECLKSRC4 (EPWM_CLKSRC[18:16]) for EPWM_CLK4.

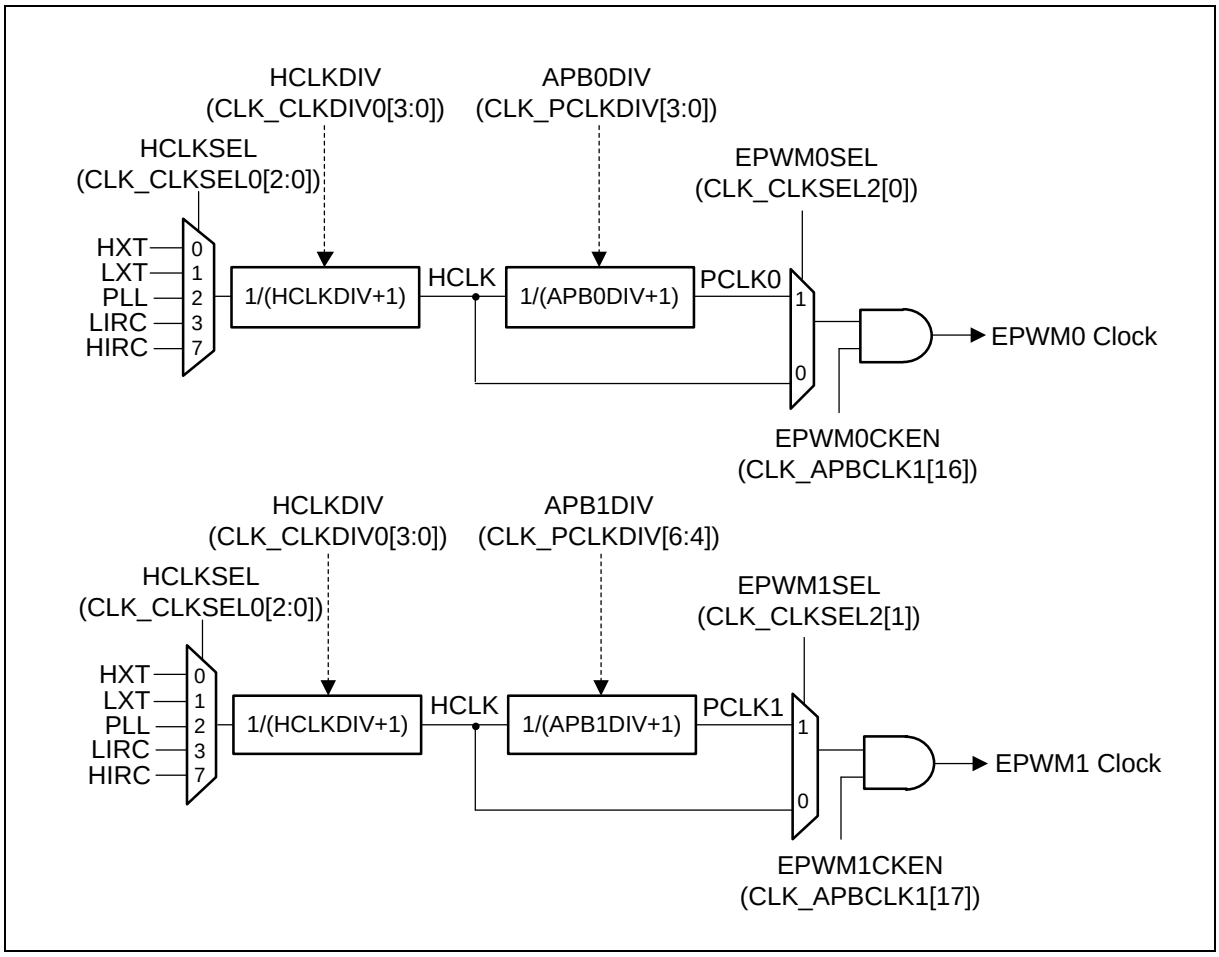


Figure 6.12-2 EPWM Clock Source Control

Frequency Ratio PCLK:EPWM Clock	HCLK	PCLK	EPWM Clock	HCLKSEL CLK_CLKSEL0[2:0]	HCLKDIV CLK_CLKDIV0[3:0]	APBnDIV (CLK_PCLKDIVn [2+4n:4n]), N Denotes 0 Or 1	EPWMnSEL (CLK_CLKSEL2[N]), N Denotes 0 Or 1
1:1	HCLK	PCLK	PCLK	Don't care	Don't care	Don't care	1
1:2	PLL	HCLK/ 2	HCLK	2	0	1	0

Table 6.12-1 EPWM Clock Source Control Registers Setting Table

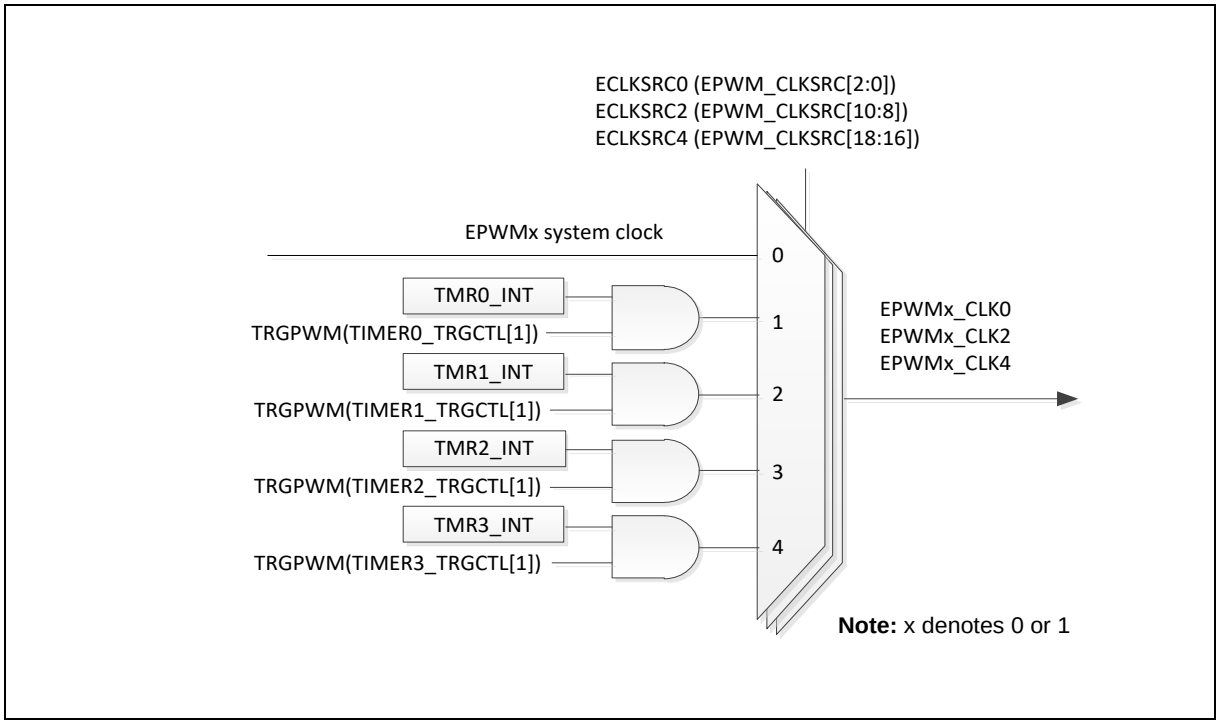


Figure 6.12-3 EPWM Clock Source Control

Figure 6.12-4 and Figure 6.12-5 illustrate the architecture of EPWM independent mode and complementary mode. No matter independent mode or complementary mode, paired channels' (EPWM_CH0 and EPWM_CH1, EPWM_CH2 and EPWM_CH3, EPWM_CH4 and EPWM_CH5) counters both come from the same clock source and prescaler. When counter count to 0, PERIOD (EPWM_PERIODn[15:0]) or equal to comparator, events will be generated. These events are passed to corresponding generators to generate EPWM pulse, interrupt signal and trigger signal for EADC/DAC to start conversion. Output control is used to change EPWM pulse output state; brake function in output control also generates interrupt events. In complementary mode, synchronize function is available and even channel use odd channel comparator to generate events, free trigger comparator events only use to generate trigger EADC signals.

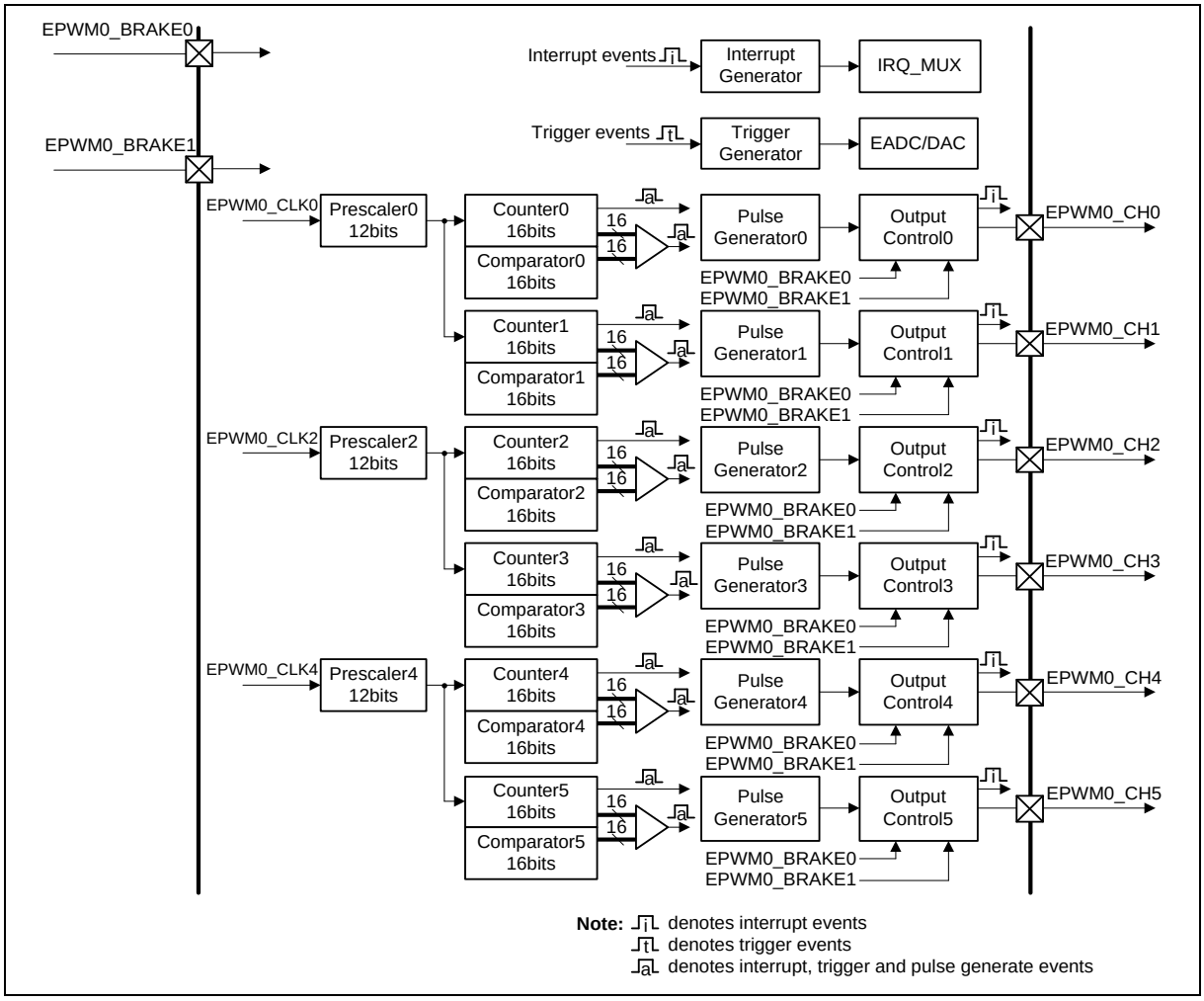


Figure 6.12-4 EPWM Independent Mode Architecture Diagram

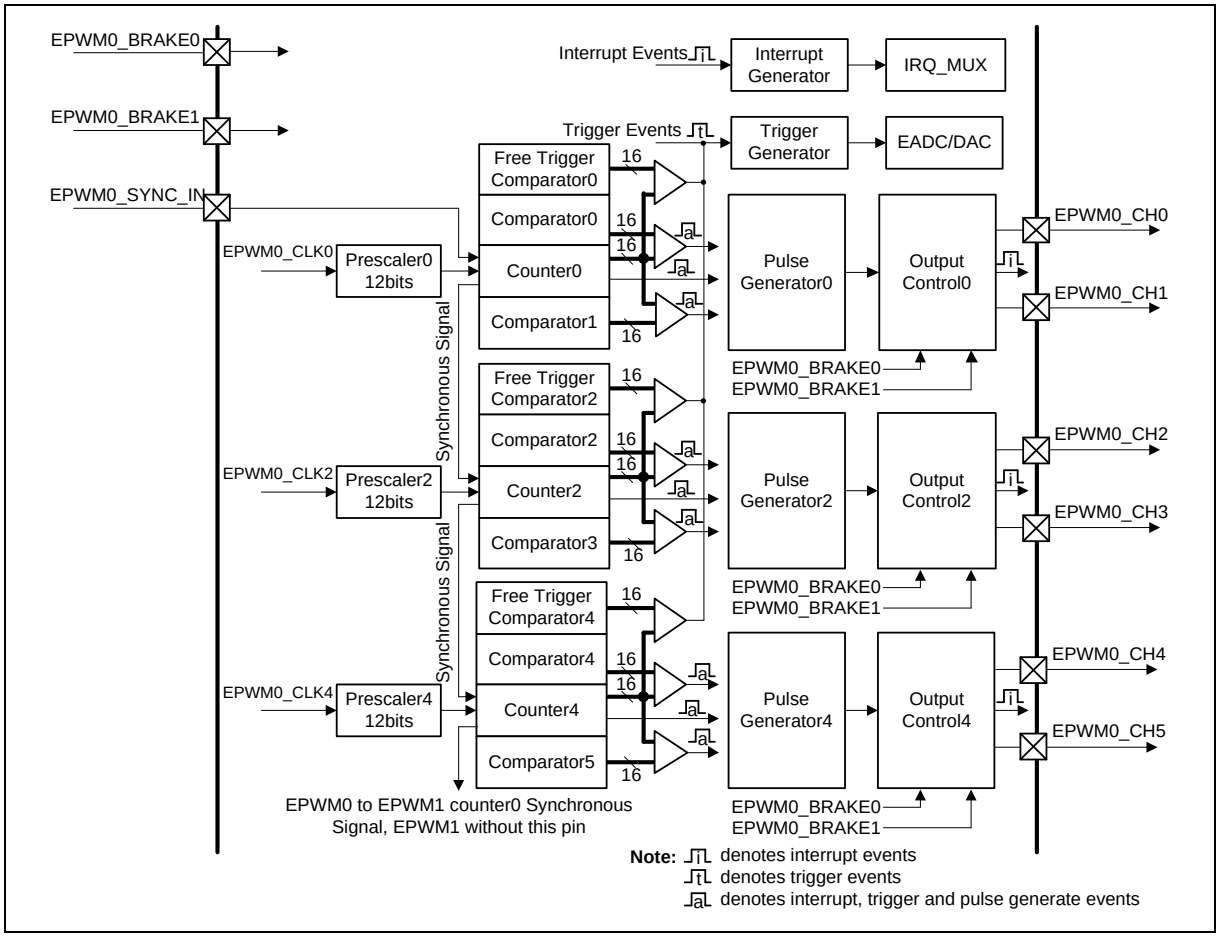


Figure 6.12-5 EPWM Complementary Mode Architecture Diagram

6.12.4 Basic Configuration

6.12.4.1 EPWM0 Basic Configuration

- Clock Source Configuration
  - Select the source of EPWM0 peripheral clock on EPWM0SEL (CLK_CLKSEL2[0])
  - Enable EPWM0 peripheral clock in EPWM0CKEN (CLK_APBCLK1[16]).
- Reset Configuration
  - Reset EPWM0 in EPWM0RST SYS_IPRST2[16]

6.12.4.2 EPWM1 Basic Configuration

- Clock Source Configuration
  - Select the source of EPWM1 peripheral clock on EPWM1SEL (CLK_CLKSEL2[1])
  - Enable EPWM1 peripheral clock in EPWM1CKEN (CLK_APBCLK1[17]).
- Reset Configuration
  - Reset EPWM1 in EPWM1RST (SYS_IPRST2[17])

6.12.5 Functional Description

6.12.5.1 EPWM Prescaler

The EPWM prescaler is used to divide clock source, prescaler counting CLKPSC+1 times, EPWM counter only count once. The prescale double buffer is setting by CLKPSC (EPWM_CLKPSCn[11:0], n = 0, 1, …, 5). Figure 6.12-6 is an example of EPWM channel 0 prescale waveform. The prescale counter will reload CLKPSC at the begin of the next prescale counter down-count.

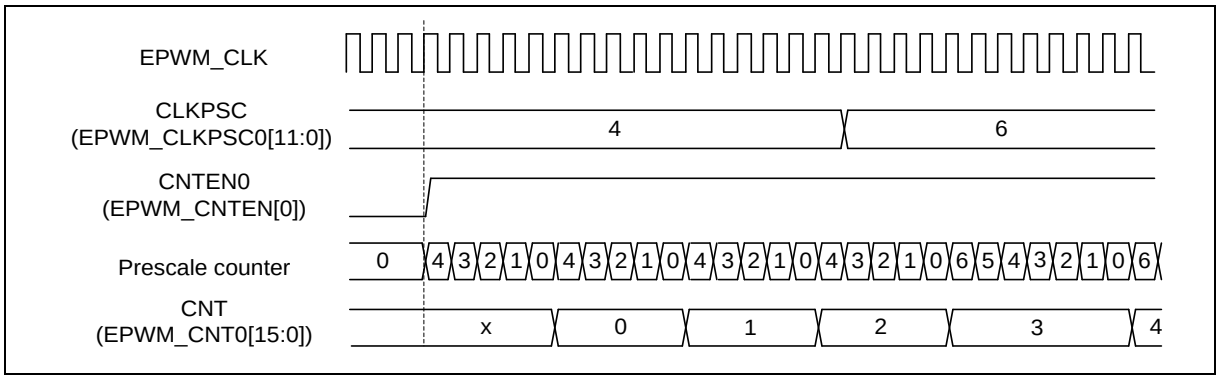


Figure 6.12-6 EPWM_CH0 Prescaler Waveform in Up Counter Type

6.12.5.2 EPWM Counter

The EPWM supports 3 counter types operation: Up Counter, Down Counter and Up-Down Counter types.

For EPWM channel0, CNT(EPWM_CNT0[15:0]) can clear to 0x00 by CNTCLR0 (EPWM_CNTCLR[0]). CNT will be cleared when prescale counter count to 0, and CNTCLR0 will be set 0 by hardware automatically.

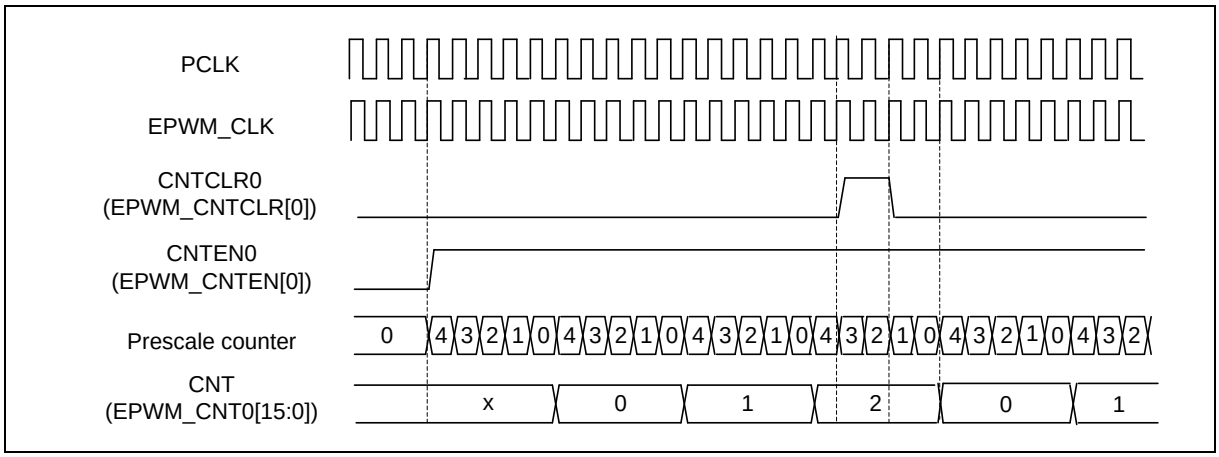


Figure 6.12-7 EPWM Counter Waveform when Setting Clear Counter

6.12.5.3 Up Counter Type

When EPWM counter is set to up counter type, CNTTYPEn (EPWM_CTL1[2n+1:2n], n = 0,1..5) is 0x0, it starts up-counting from 0 to PERIOD (EPWM_PERIODn[15:0], where n denotes channel number) to complete a EPWM period. The current counter value can be read from CNT (EPWM_CNTn[15:0]) bits. EPWM generates zero point event when the counter counts to 0 and prescale counts to 0. EPWM generates period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.12-8 shows an example of up counter, wherein

EPWM period time = (PERIOD+1) *(CLKPSC+1)* EPWMx_CLK.

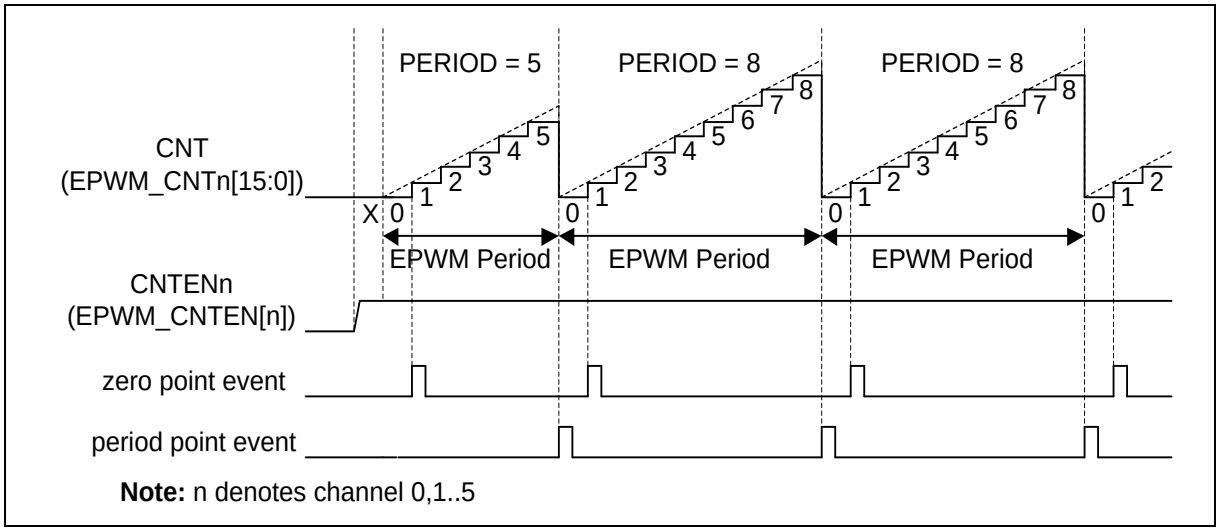


Figure 6.12-8 EPWM Up Counter Type

6.12.5.4 Down Counter Type

When EPWM counter is set to down counter type, CNTTYPEn (EPWM_CTL1[2n+1:2n], n = 0,1..5) is 0x1, it starts down-counting from PERIOD to 0 to complete a EPWM period. The current counter value can be read from CNT (EPWM_CNTn[15:0]) bits. EPWM generates zero point event when the counter counts to 0 and prescale counts to 0. EPWM generates period point event when the counter counts to PERIOD and prescale counts to 0. Figure 6.12-9 shows an example of down counter, wherein

EPWM period time = (PERIOD+1) * (CLKPSC+1) * EPWMx_CLK.

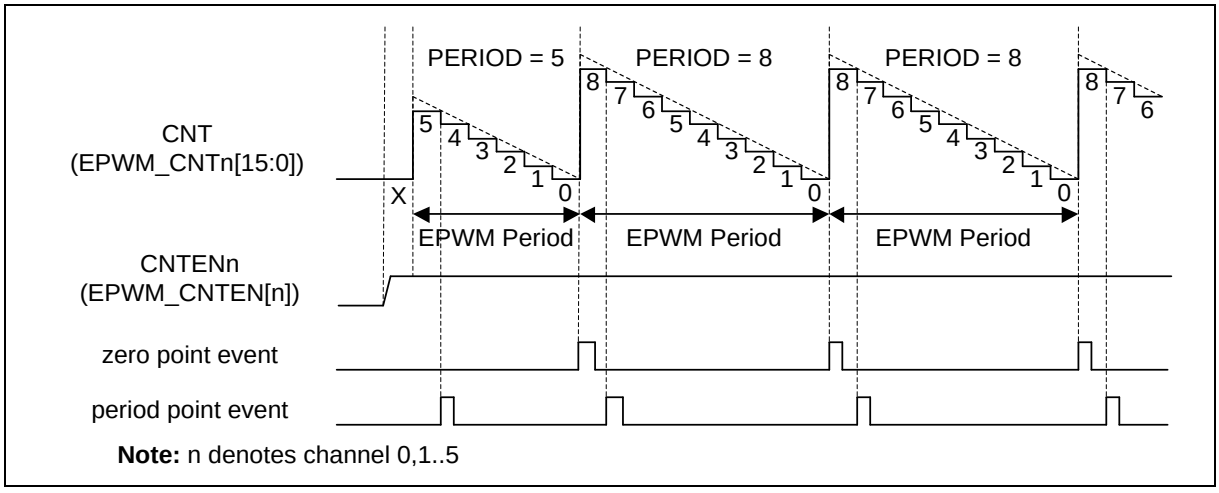


Figure 6.12-9 EPWM Down Counter Type

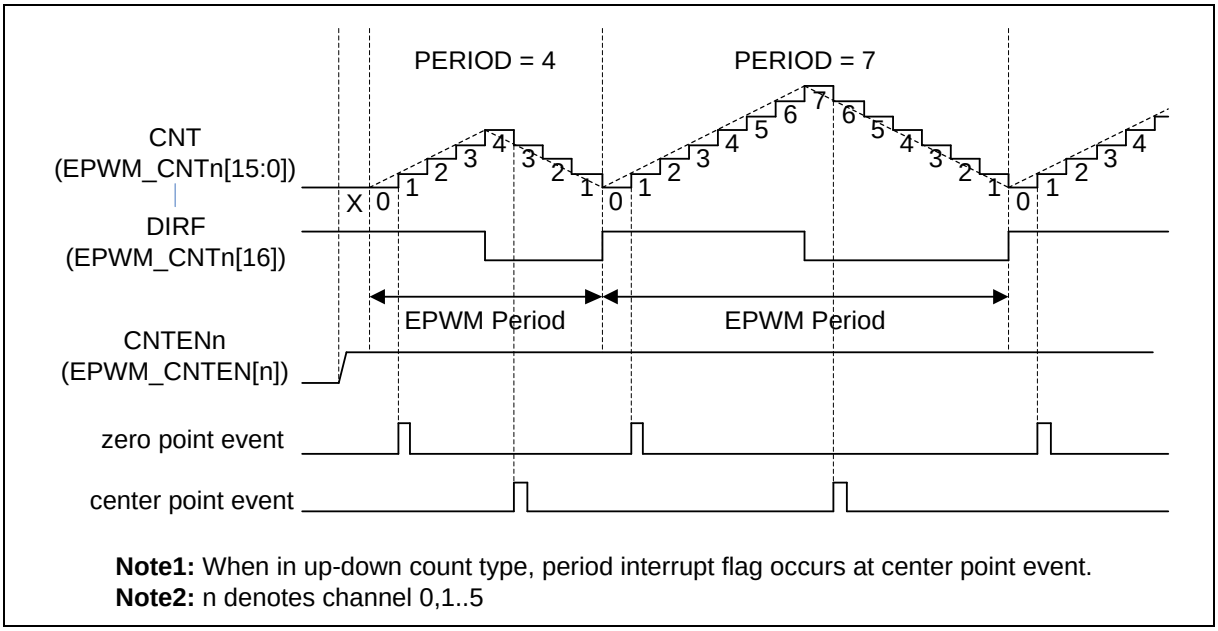
6.12.5.5 Up-Down Counter Type

When EPWM counter is set to up-down count type, CNTTYPEn (EPWM_CTL1[2n+1:2n], n = 0,1..5) is 0x2, it starts counting-up from 0 to PERIOD and then starts counting down to 0 to complete a EPWM

period. The current counter value can be read from CNT (EPWM_CNTn[15:0]) bits. EPWM generates zero point event when the counter counts to 0 and prescale counts to 0. EPWM generates center point event which is equal to period point event when the counter counts to PERIOD. Figure 6.12-10 shows an example of up-down counter, wherein

EPWM period time =(2*PERIOD) * (CLKPSC+1) * EPWMx_CLK.

The DIRF (EPWM_CNTn[16]) bit is counter direction indicator flag, where high is up counting, and low is down counting.



Note1: When in up-down count type, period interrupt flag occurs at center point event.
Note2: n denotes channel 0,1..5

Figure 6.12-10 EPWM Up-Down Counter Type

6.12.5.6 EPWM Comparator

There are two kinds of comparator registers : one is EPWM_CMPDATn(n = 0,1..5), and the other is EPWM_FTCMPDATn_m(n = 0,2,4, m = 1,3,5) register. EPWM_CMPDATn is a basic comparator register of EPWM channel n; In Independent mode each channel only has one comparator, the value of EPWM_CMPDATn register is continuously compared to the corresponding channel's counter value. In Complementary mode each paired channels has two comparators, and the value of EPWM_CMPDATn and EPWM_CMPDATm (n = 0,2,4, m = 1,3,5) registers are continuously compared to the complementary even channel's counter value, because of odd channel's counter is useless. For example, channel 0 and channel 1 are complementary channels, in Complementary mode, channel 1's comparator is continuously compared to channel 0's counter, but not channel 1's. When the counter is equal to value of EPWM_CMPDAT0 register, EPWM generates a compared point event and uses the event to generate EPWM pulse, interrupt or use to trigger EADC/DAC. In up-down counter type, two events will be generated in a EPWM period as shown in Figure 6.12-11. The CMPU is up count compared point event and CMPD is down count compared point event.

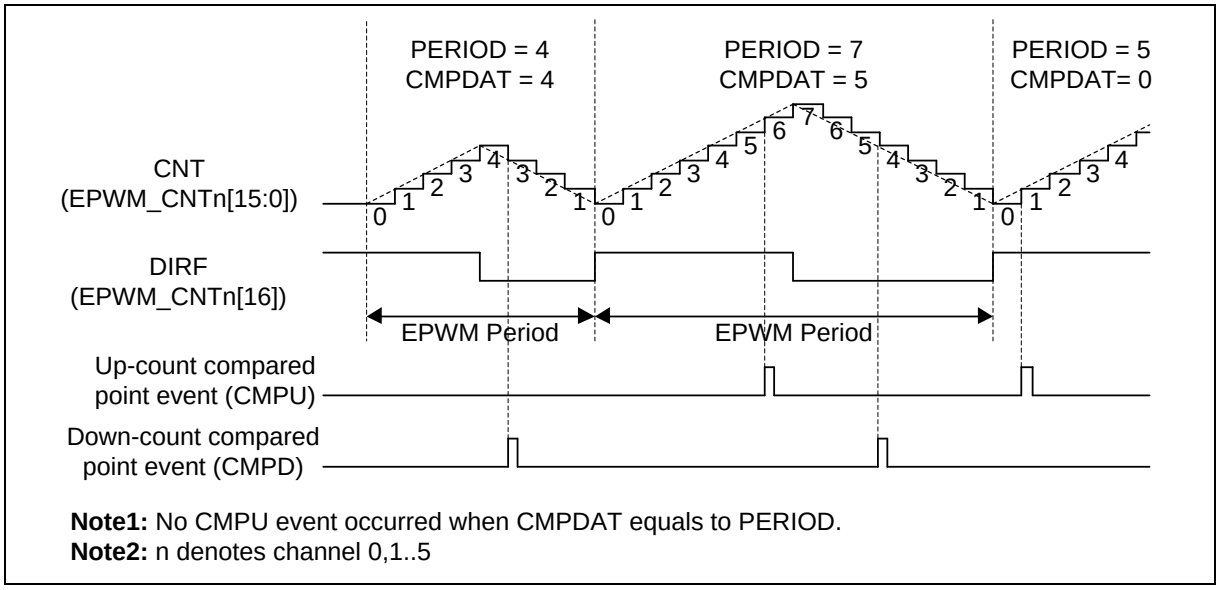


Figure 6.12-11 EPWM Compared point Events in Up-Down Counter Type

EPWM_FTCMPDATn_m is a free trigger comparator register. Each complementary paired channel only supports one free trigger comparator. The value of EPWM_FTCMPDATn_m(n = 0,2,4, m = 1,3,5) register is continuously compared to even channel's counter value. When counter is equal to the value of EPWM_FTCMPDATn_m register, FTCMDn (EPWM_FTCI[10:8], n=0,2,4) indicator is set in down count type and FTCMUn (EPWM_FTCI[2:0], n=0,2,4) indicator is set in up count type. In addition, EPWM generates an event and only uses to trigger EADC.

6.12.5.7 EPWM Double Buffering

The double buffering uses double buffers to separate software writing and hardware action operation timing. There are four loading modes for loading values to buffer: period loading mode, immediately loading mode, window loading mode and center loading mode. After registers are modified through software, hardware will load register value to the buffer register according to the loading mode timing. The hardware action is based on the buffer value. This can prevent asynchronously operation problem due to software and hardware asynchronism.

The EPWM provides PBUF (EPWM_PBUFn[15:0]) as the active EPWM_PERIODn buffer register, CMPBUF (EPWM_CMPBUFn[15:0]) as the active EPWM_CMPDATn buffer register, FTCMPBUF (EPWM_FTCBUFn_m[15:0]) as the active EPWM_FTCMPDATn_m buffer register and CPSCBUF (EPWM_CPSCBUFn[15:0]) as the active EPWM_CLKPSCn buffer register. The concept of double buffering is used in loading modes, which are described in the following sections. For example, as shown in Figure 6.12-12, in period loading mode, writing PERIOD(EPWM_PERIODn[15:0]), CMPDAT(EPWM_CMPDATn[15:0]) and FTCMP(EPWM_FTCMPDATn_m[15:0]) buffer through software, EPWM will load new values to their buffer PBUF(EPWM_PBUFn[15:0]), CMPBUF (EPWM_CMPBUFn[15:0]) and FTCMPBUF (EPWM_FTCBUFn_m[15:0]) at start of the next period without affecting the current period counter operation. FTCMPU denotes up-count free trigger compared point event and FTCMPD denotes down-count free trigger compared event.

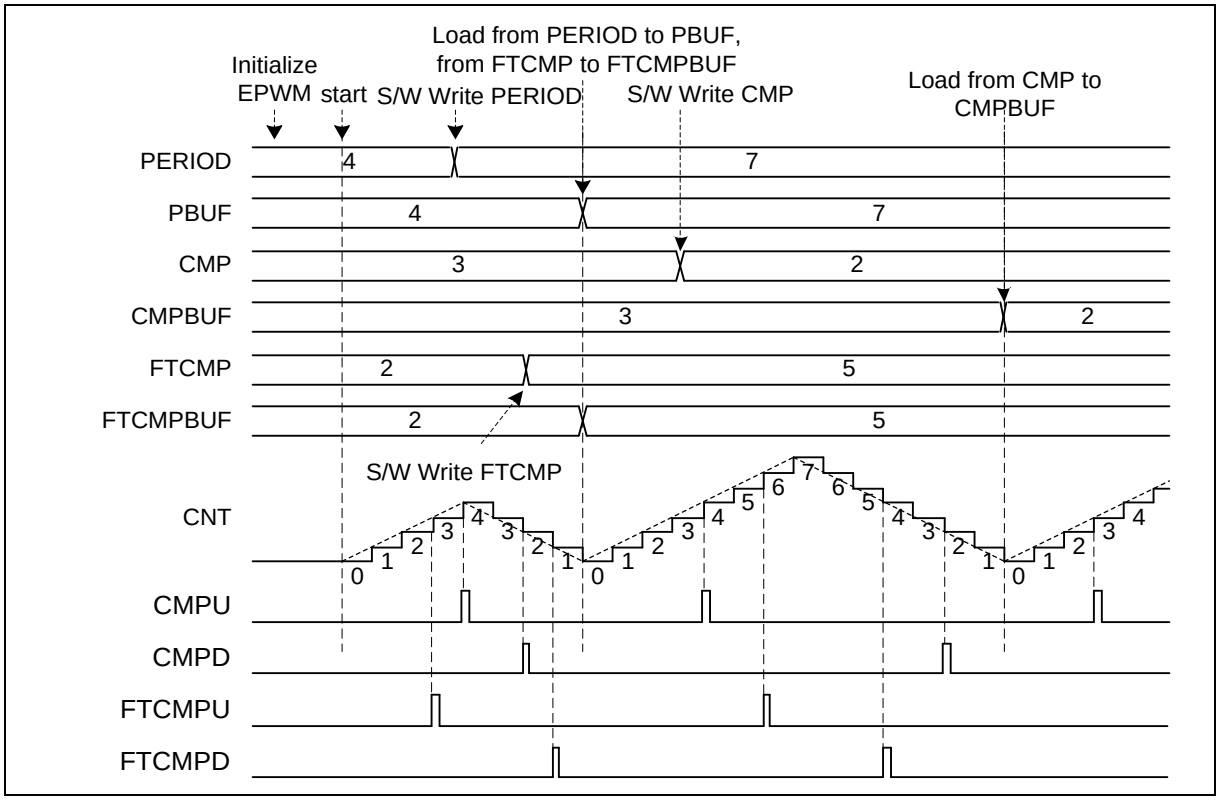


Figure 6.12-12 EPWM Double Buffering Illustration

6.12.5.8 Period Loading Mode

When immediately loading mode, window loading mode and center loading mode are disabled that IMMLDENn bits, WINLDENn bits and CTRLDn bits of EPWM_CTL0 register are set to 0, EPWM operates at period Loading mode. In period Loading mode, CLKPSC(EPWM_CLKPSCn[11:0]), PERIOD(EPWM_PERIODn[15:0]), CMP(EPWM_CMPDATn[15:0]) and FTCMPDAT (EPWM_FTCMPDATn_m[15:0]) will all load to their active CPSCBUF(EPWM_CPSCBUFn[11:0]), PBUF(EPWM_PBUFn[15:0]), CMPBUF(EPWM_CMPBUFn[15:0]) and FTCMPBUF(EPWM_FTCBUFn_m[15:0]) buffer while each period is completed. For example, after EPWM counter up counts from 0 to PERIOD in the up-counter operation or down counts from PERIOD to 0 in the down-counter operation or counts up from 0 to PERIOD and then counts down to 0 in the up-down counter operation.

Figure 6.12-13 shows period loading timing of up-count operation, where PERIOD DATA0 denotes the initial data of PERIOD, PERIOD DATA1 denotes the first updated PERIOD data by software and so on. CMP also follows this rule. The following describes steps sequence of Figure 6.12-13. User can know the PERIOD and CMP update condition, by watching EPWM period and CMPU event.

1. Software writes CMP DATA1 to CMP at point 1.
2. Hardware loads CMP DATA1 to CMPBUF at the end of EPWM period at point 2.
3. Software writes PERIOD DATA1 to PERIOD at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of EPWM period at point 4.
5. Software writes PERIOD DATA2 to PERIOD at point 5.
6. Hardware loads PERIOD DATA2 to PBUF at the end of EPWM period at point 6.

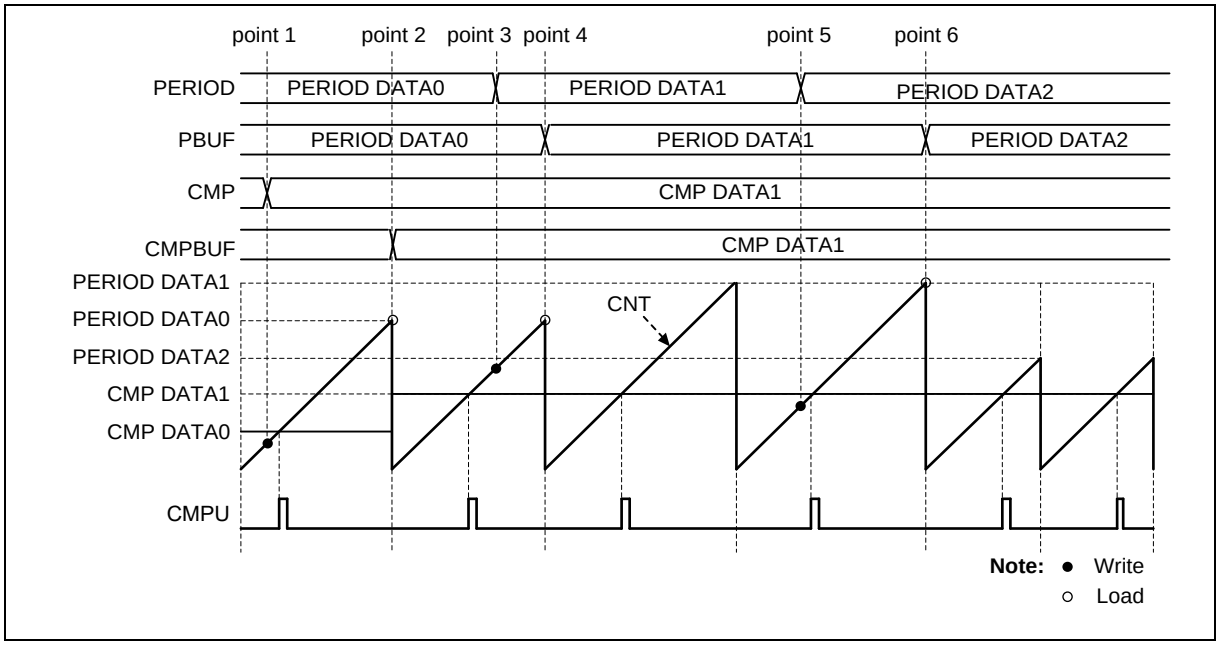


Figure 6.12-13 Period Loading in Up-Count Mode

6.12.5.9 Immediately Loading Mode

If the IMMLDENn (EPWM_CTL0[21:16]) bit is set to 1, EPWM operates at immediately loading mode. In immediately loading mode, when user update CLKPSC(EPWM_CLKPSCn[11:0]), PERIOD(EPWM_PERIODn[15:0]), CMP(EPWM_CMPDATn[15:0]) or FTCMPDAT (EPWM_FTCMPDATn_m[15:0]), CLKPSC, PERIOD, CMP or FTCMPDAT will be load to active CPSCBUF (EPWM_CPSCBUFn[15:0]), PBUF (EPWM_PBUFn[15:0]), CMPBUF (EPWM_CMPBUFn[15:0]) or FTCMPBUF (EPWM_FTCBUF[15:0]) after current counter count is completed. If the updated PERIOD value is less than current counter value, counter will count to 0xFFFF, when counter count to 0xFFFF and prescale count to 0, the flag CNTMAXF(EPWMx_STATUS[5:0]) will raise, and then counter will count wraparound. Immediately loading mode has the highest priority. If IMMLDENn has been set, other loading mode for channel n will become invalid.Figure 6.12-14 shows an example and its steps sequence is described below.

1. Software writes CMP DATA1 and hardware immediately loading CMP DATA1 to CMPBUF at point 1.
2. Software writes PERIOD DATA1 which is greater than current counter value at point 2; counter will continue counting until equal to PERIOD DATA1 to finish a period loading.
3. Software writes PERIOD DATA2 which is less than the current counter value at point 3; counter will continue counting to its maximum value 0xFFFF and count wraparound from 0 to PERIOD DATA2 to finish this period loading.

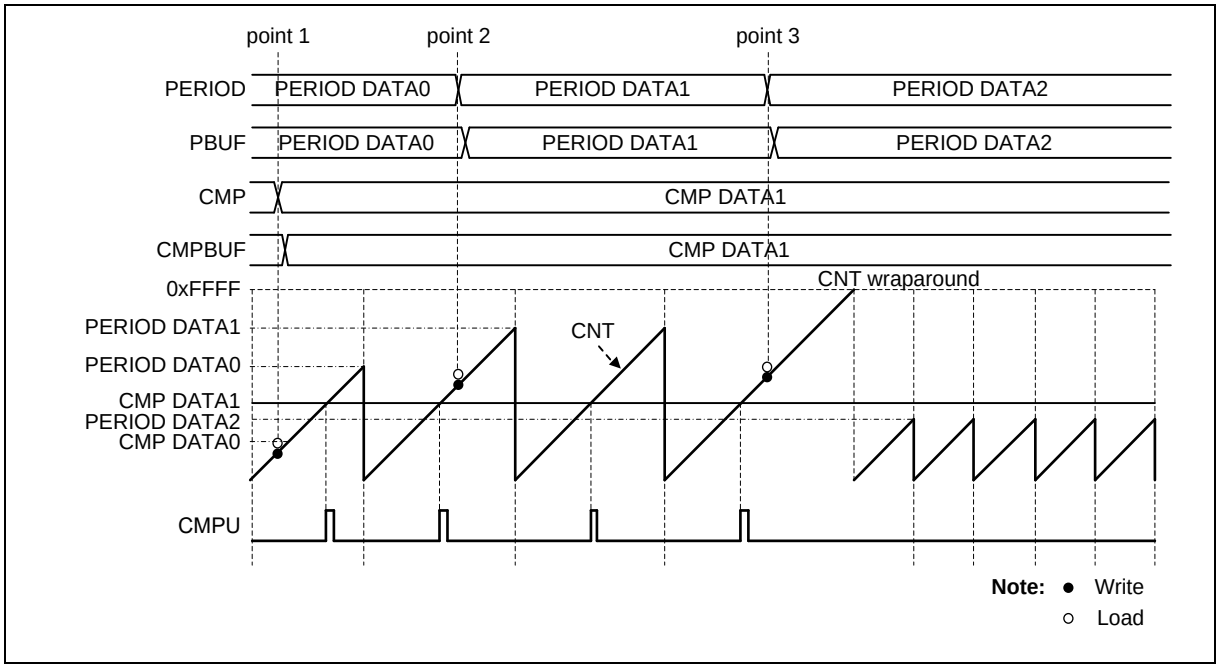


Figure 6.12-14 Immediately Loading in Up-Count Mode

6.12.5.10 Window Loading Mode

When the WINLDENn (EPWM_CTL0[13:8]) bit is set to 1, EPWM operates at window loading mode. In Window loading mode, CLKPSC(EPWM_CLKPSCn[11:0]), PERIOD(EPWM_PERIODn[15:0]), CMP(EPWM_CMPDATn[15:0]) and FTCMP(EPWM_FTCMPDATn_m[15:0]) will all load to their active CPSCBUF(EPWM_CPSCBUFn[11:0]), PBUF(EPWM_PBUFn[15:0]), CMPBUF(EPWM_CMPBUFn[15:0]) and FTCMPBUF(EPWM_FTCBUFn_m[15:0]) buffers while each period is completed, but CMPBUF loading are valid only when load window is opened. Every channel n's load window is opened by setting the corresponding LOADn (EPWM_LOAD[5:0]) to 1, and hardware will close the window at the end of EPWM period. Figure 6.12-15 shows an example and its steps sequence is described below.

1. Software writes CMP DATA1 at point 1, and the load window is not opened at this period so CMP will not load to CMPBUF.
2. Software writes LOAD to open the load window at point2.
3. Software writes CLKPSC DATA1 and PERIOD DATA1 at point 3.
4. At point 4, load window has been opened, hardware loads CLKPSC DATA1, PERIOD DATA1 and CMP DATA1 to their buffer and closes the load window at the end of EPWM period.
5. Software writes CLKPSC DATA2 and PERIOD DATA2 at point 5.
6. Hardware loads CLKPSC DATA2 and PERIOD DATA2 to their buffer at the end of EPWM period at point 6.
7. Software writes CLKPSC DATA3 and PERIOD DATA3 at point 7.
8. Software writes LOAD to open the load window at point8.
9. Hardware loads CLKPSC DATA3 and PERIOD DATA3 to their buffer and closes the load window at the end of EPWM period at point 9.

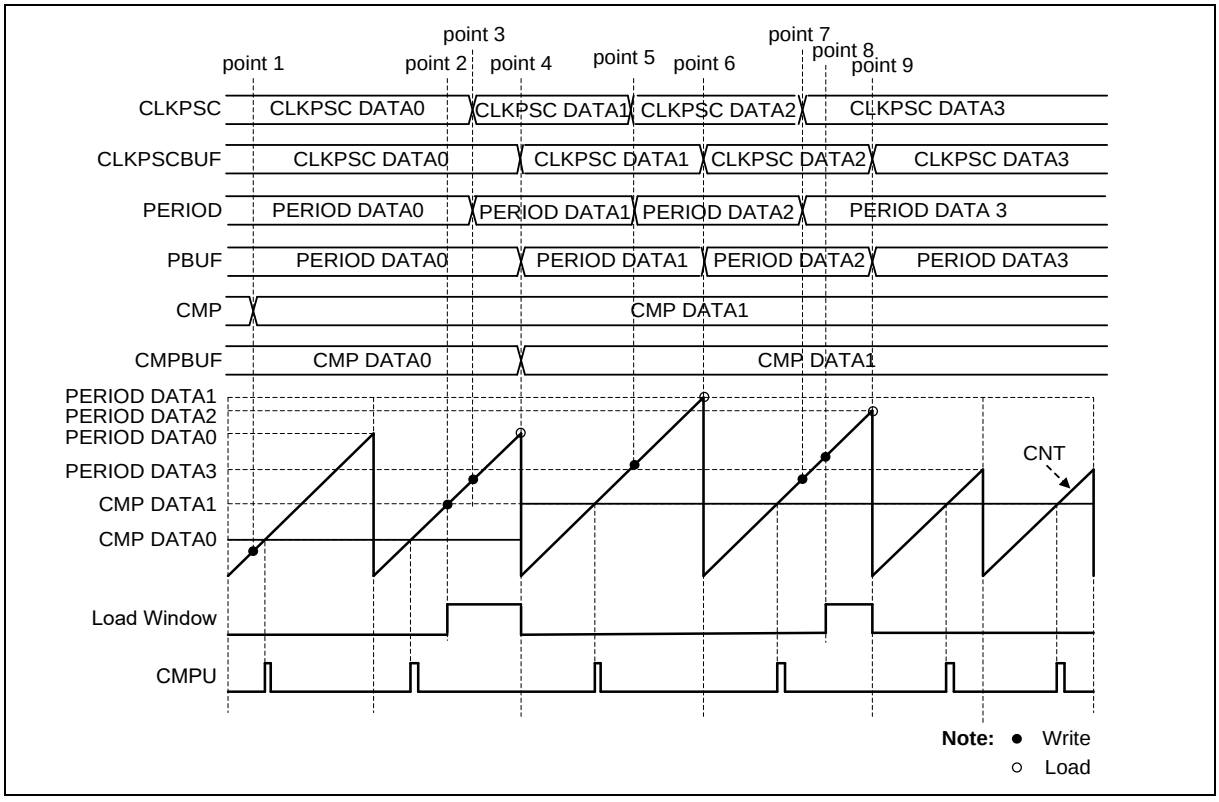


Figure 6.12-15 Window Loading in Up-Count Mode

6.12.5.11 Center Loading Mode

When the CTRLDn (EPWM_CTL0[5:0]) bit is set to 1 and EPWM counter is set to up-down count type, CNTTYPEn (EPWM_CTL1[2n+1:2n], n = 0,1..5) is 0x2, EPWM operates at center loading mode. In center loading mode, CMP(EPWM_CMPDATn[15:0]) and FTCMPDAT (EPWM_FTCMPDATn_m[15:0]) will load to active CMPBUF(EPWM_CMPBUFn[15:0]) and FTCMPBUF(EPWM_FTCBUFn_m[15:0]) buffers in center of each period, that is, counter counts to PERIOD. CLKPSC(EPWM_CLKPSCn[11:0]) and PERIOD(EPWM_PERIODn[15:0]) will all load to their active CPSCBUF(EPWM_CPSCBUFn[11:0]) and PBUF(EPWM_PBUFn[15:0]) buffers while each period is completed. Center loading mode can work with window loading mode, the CMP(EPWM_CMPDATn[15:0]) will load to active CMPBUF(EPWM_CMPBUFn[15:0]) buffer in center of each period, but it is valid only at the interval of load window. Figure 6.12-16 shows an example and its steps sequence is described below.

1. Software writes CMP DATA1 at point 1.
2. Hardware loads CMP DATA1 to CMPBUF at center of EPWM period at point 2.
3. Software writes PERIOD DATA1 at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of EPWM period at point 4.
5. Software writes CMP DATA2 at point 5.
6. Hardware loads CMP DATA2 to CMPBUF at center of EPWM period at point 6.
7. Software writes PERIOD DATA2 at point 7.
8. Hardware loads PERIOD DATA2 to PBUF at the end of EPWM period at point 8.

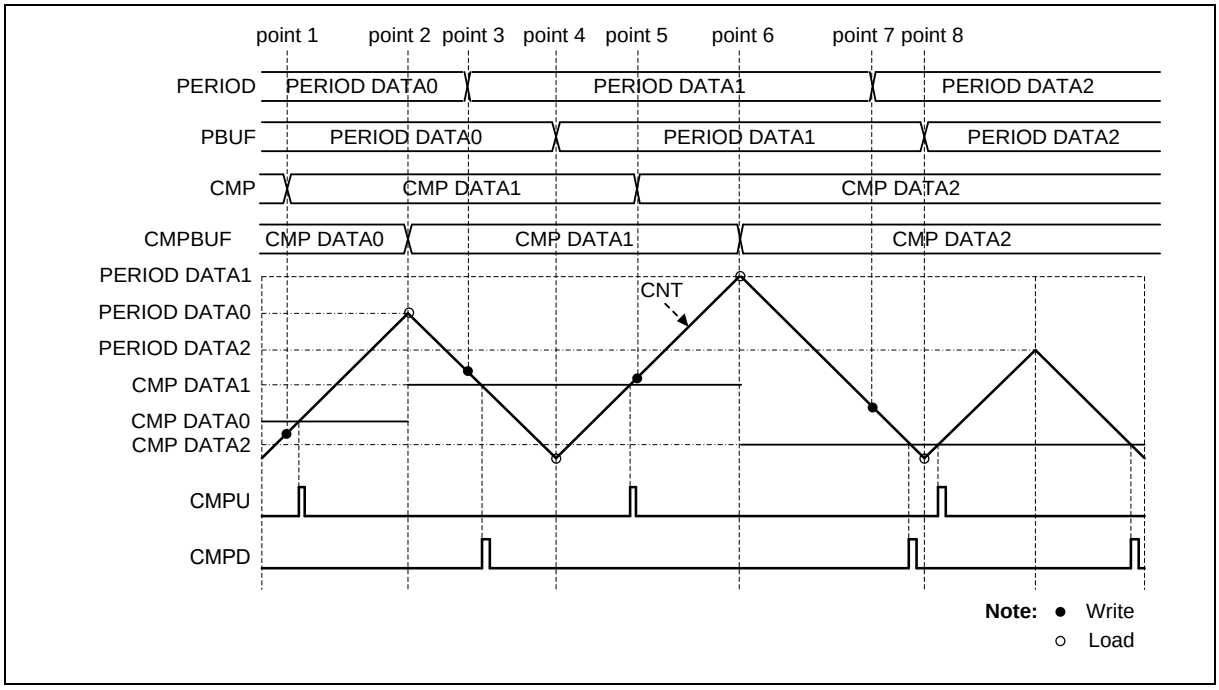


Figure 6.12-16 Center Loading in Up-Down-Count Mode

6.12.5.12 EPWM Counter Operation Mode

The EPWM counter supports two operation modes: One-shot mode and Auto-reload mode. EPWM counter will operate in One-shot mode if CNTMODEn (EPWM_CTL1[21:16]) bit is set to 1, and operate in Auto-reload mode if set to 0.

In One-shot mode, EPWM_CMPDATn and EPWM_PERIODn registers should be written first and then set CNTENn (EPWM_CNTEN[5:0]) bit as 1 to enable EPWM prescaler and counter start running. After EPWM counter counted a period, counter value will keep 0.

User can re-start next one-shot by writing new value to CMP(EPWM_CMPDATn[15:0]) bits. If one-shot counter still running, to update EPWM_CMPDATn register will cause next one-shot as continuous one-shot. Besides, to write EPWM_CMPDATn register twice under continuous one-shot operation, latest value in EPWM_CMPDATn register is valid at next one-shot period and only generate one-shot pulse once. Moreover, if user wants to clear counter within one-shot operation and starts next one-shot, user should monitor counter value to check counter has cleared and then writes EPWM_CMPDATn register. Figure 6.12-17 is an example and following is steps sequence.

1. Software writes PERIOD DATA1 and hardware immediately loading PERIOD DATA1 to PBUF at point 1.
2. Software writes CMP DATA1 which is equal to CMP DATA0 at point 2 and hardware immediately loading CMPDAT DATA1 to CMPBUF, this event also trigger one-shot.
3. Software writes CMP DATA2 and re-trigger next one-shot (continuous one-shot) at point 3.
4. Software writes CMP DATA3 to cover CMP DATA2 and re-trigger next one-shot at point 4.
5. Period loading CMP DATA3 to CMPBUF at point 5.
6. There are no new CMP write in the previous period, and the counter value is kept as 0 at point 6.

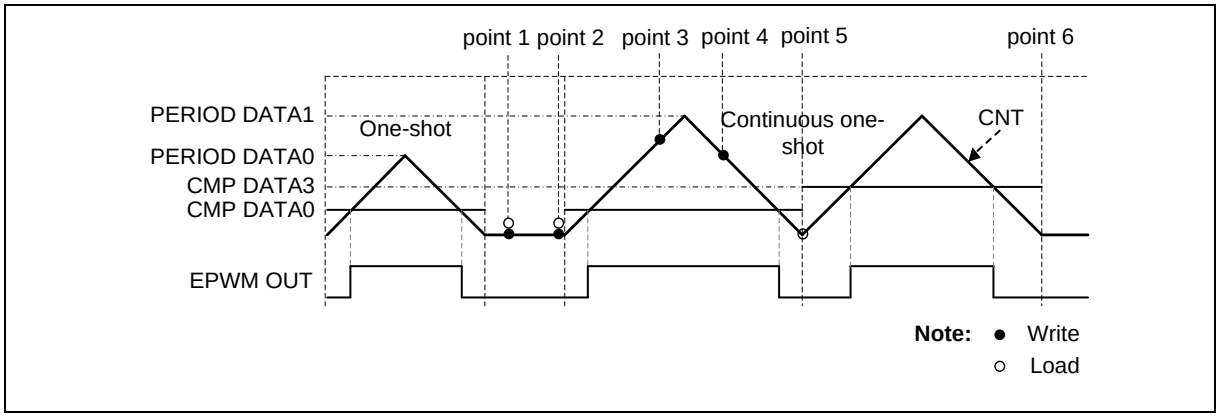


Figure 6.12-17 EPWM One-shot Mode Output Waveform

In Auto-reload mode, EPWM_CMPDATn and EPWM_PERIODn registers should be written first and then the CNTENn(EPWM_CNTEN[n]) bit is set to 1 to enable EPWM prescaler and start to run counter. The value of CLKPSC(EPWM_CLKPSCn[11:0]), PERIOD(EPWM_PERIODn[15:0]) and CMP(EPWM_CMPDATn[15:0]) will auto reload to their active buffer according different loading mode. If PERIOD(EPWM_PERIODn[15:0]) is set to 0, EPWM counter will be set to 0.

6.12.5.13 EPWM Pulse Generator

The EPWM pulse generator uses counter and comparator events to generate EPWM pulse. The events are: zero point, period point in up counter type and down counter type, center point in up-down counter type and counter equal to comparator point in three types. As to up-down counter type, there are two counter equal comparator points, one at up count and the other at down count. Besides, Complementary mode has two comparators compared with counter, and thus comparing equal points will become four in up-down counter type and two for up or down counter type.

Each event point can decide EPWM waveform to do nothing (X), set Low (L), set High (H) or toggle (T) by setting the EPWM_WGCTL0 and EPWM_WGCTL1 registers. Using these points can easily generate asymmetric EPWM pulse or variant waveform as shown in Figure 6.12-18. In the figure, EPWM is in complementary mode, there are two comparators n and m to generate EPWM pulse. n denotes even channel number 0, 2, or 4, and m denotes odd channel number 1, 3, or 5. n channel and m channel are complementary paired. Complementary mode uses two channels (CH0 and CH1, CH2 and CH3, or CH4 and CH5) as a pair of EPWM outputs to generate complement paired waveforms. CMPU denotes CNT(EPWM_CNTn[15:0]) is equal to CMP(EPWM_CMPDATn[15:0]) when counting up. CMPD denotes CNT bits is equal to CMP bits when counting down.

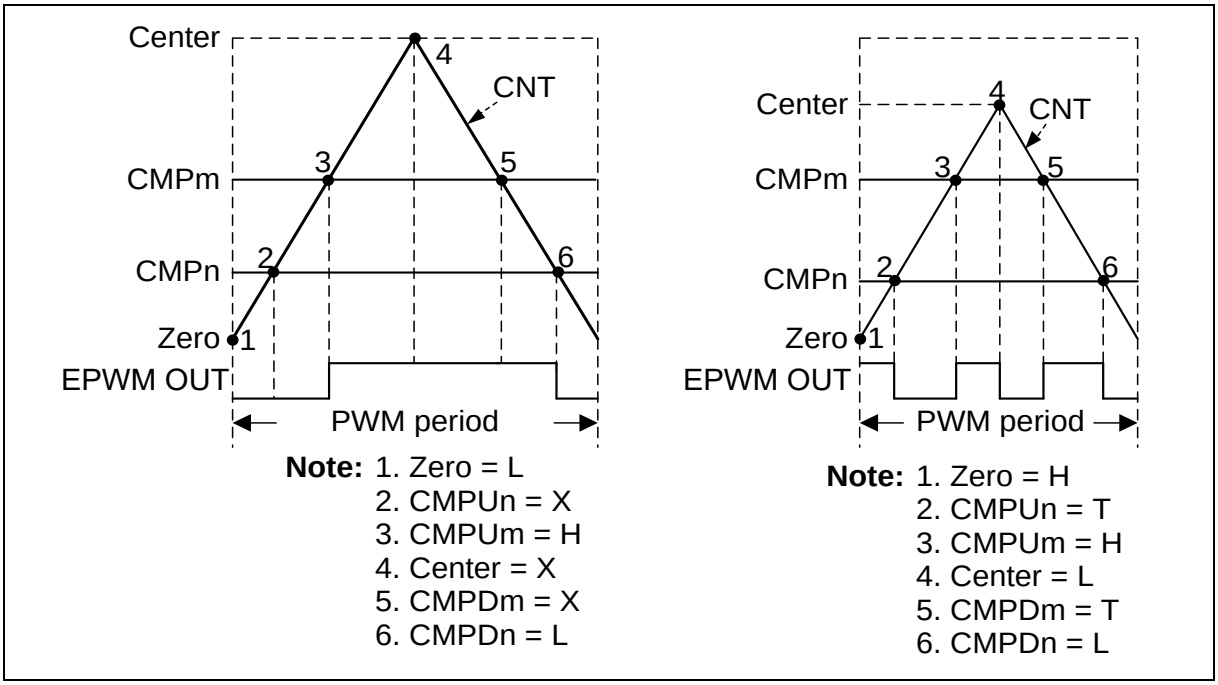


Figure 6.12-18 EPWM Pulse Generation

User can also use software event to force EPWM output. Software trigger SWETRGn (EPWM_SWEOFTRG[n], n=0~5) event can decide EPWM waveform to do nothing (X), set Low (L), set High (H) or toggle (T) by setting OUTACTSn (EPWM_SWEOFCTL[2n+1:2n]).

Figure 6.12-19 shows software event effect EPWM output according to OUTACTS0 setting. Besides, software event has the highest priority. If other events and software event happen at the same time, software event will be serviced first.

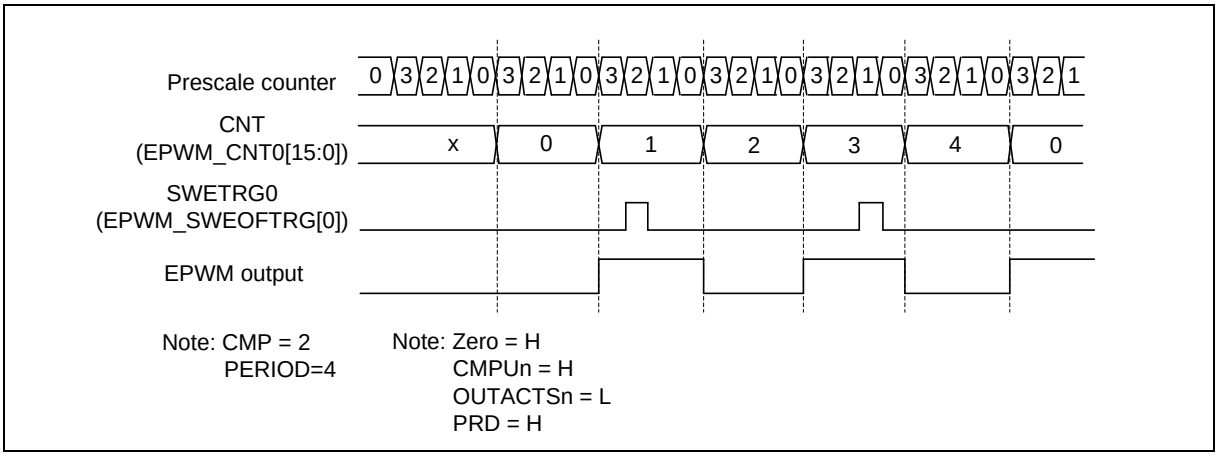


Figure 6.12-19 Software Event Output Force

The generation events may sometimes set to the same value, as the reason, events priority between different counter types are list below, up counter type(Table 6.12-2), down counter type (Table 6.12-3) and up-down counter type (Table 6.12-4). By using event priority, user can easily generate 0% to 100% duty pulse as shown in Figure 6.12-20.

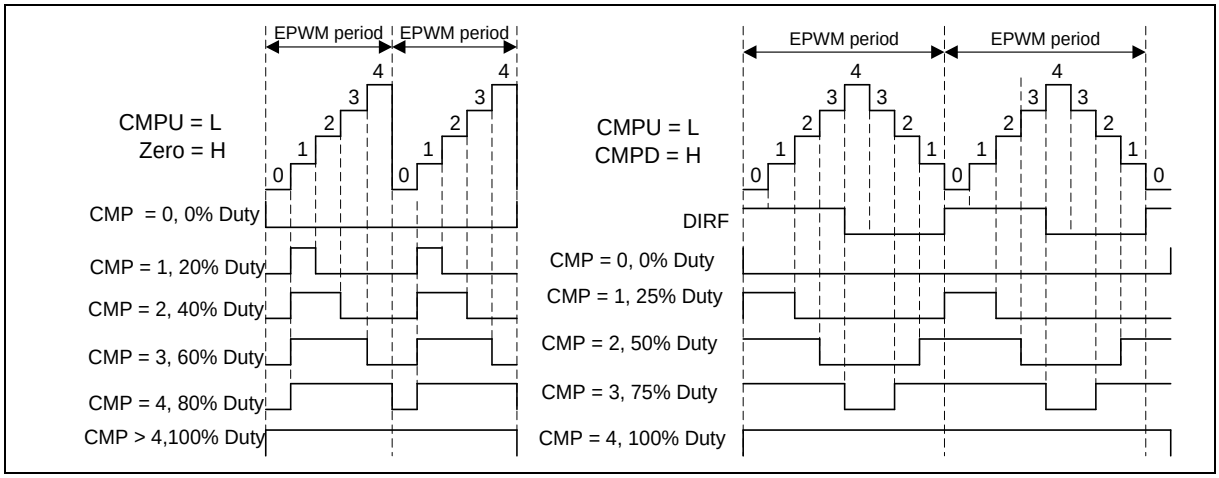


Figure 6.12-20 EPWM 0% to 100% Pulse Generation

Priority	Up Event
1 (Highest)	Period event (CNT = PERIOD)
2	Compare up event of odd channel (CNT = CMPUm)
3	Compare up event of even channel (CNT = CMPUn)
4 (Lowest)	Zero event (CNT = 0)

Table 6.12-2 EPWM Pulse Generation Event Priority for Up-Counter

Priority	Down Event
1 (Highest)	Zero event (CNT = 0)
2	Compare down event of odd channel (CNT = CMPDm)
3	Compare down event of even channel (CNT = CMPDn)
4 (Lowest)	Period event (CNT = PERIOD)

Table 6.12-3 EPWM Pulse Generation Event Priority for Down-Counter

Priority	Up Event	Down Event
1 (Highest)	Compare up event of odd channel (CNT = CMPUm	Compare down event of odd channel (CNT = CMPDm)
2	Compare up event of even channel (CNT = CMPUn)	Compare down event of even channel (CNT = CMPDn)
3 (Lowest)	Zero event (CNT = 0)	Period (center) event (CNT = PERIOD)

Table 6.12-4 EPWM Pulse Generation Event Priority for Up-Down-Counter

6.12.5.14 EPWM Output Mode

The EPWM supports two output modes: Independent mode that may be applied to DC motor system, and Complementary mode with dead-time insertion that may be used in the application of AC induction motor and permanent magnet synchronous motor.

6.12.5.15 Independent mode

By default, the EPWM is operating in independent mode, independent mode is enabled when channel n corresponding OUTMODEn (EPWM_CTL1[26:24]) bit is set to 0. In this mode six EPWM channels: EPWM_CH0, EPWM_CH1, EPWM_CH2, EPWM_CH3, EPWM_CH4 and EPWM_CH5 are running off its own period and duty as shown in Figure 6.12-21.

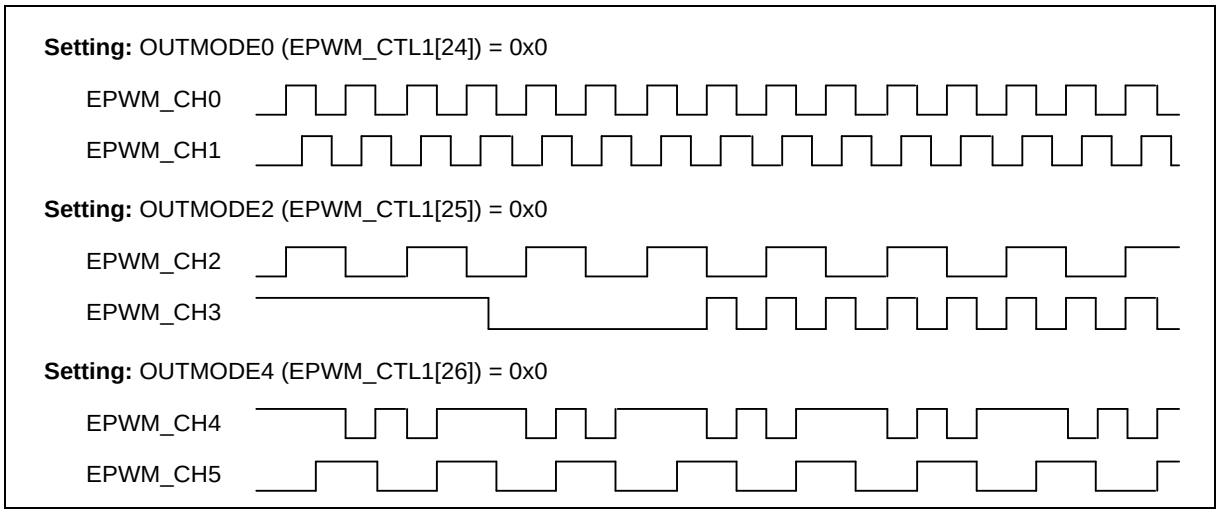


Figure 6.12-21 EPWM Independent Mode Waveform

6.12.5.16 Complementary Mode

Complementary mode is enabled when the pair channel corresponding OUTMODEn (EPWM_CTL1[26:24]) bit set to 1. In this mode there are 3 EPWM generators utilized for complementary mode, with total of 3 EPWM output paired pins in this module. It is noted that only need to set even channel counter enable of EPWM_CNTEN register, even channel period setting of EPWM_PERIODn registers, even channel prescale setting of EPWM_CLKPSCn registers (n=0,2,4) in complementary mode. In this mode, the internal odd EPWM signal must always be the complement of the corresponding even EPWM signal. EPWM_CH1 will be the complement of EPWM_CH0. EPWM_CH3 will be the complement of EPWM_CH2 and EPWM_CH5 will be the complement of EPWM_CH4 as shown in Figure 6.12-22.

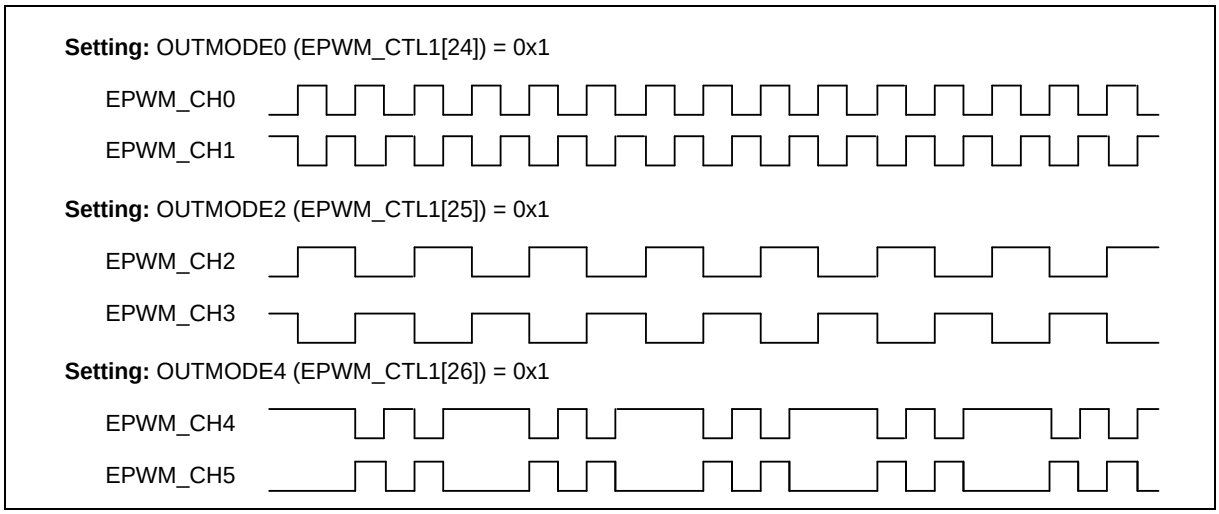


Figure 6.12-22 EPWM Complementary Mode Waveform

6.12.5.17 EPWM Output Function

Based on the output mode, there are two output functions: group and synchronous functions for advanced output control. Group function, forces the EPWM_CH2 and EPWM_CH4 synchronous with EPWM_CH0 generator and forces the EPWM_CH3 and EPWM_CH5 synchronous with EPWM_CH1, may simplify updating duty control in DC and BLDC motor applications. Besides, Synchronous function makes any channel of EPWM0 and EPWM1 in phase, user can control phase value and direction.

6.12.5.18 Group Function

Group function is enabled when GROUPEN (EPWM_CTL0[24]) is set to 1, no matter in independent or complementary mode. This control allows all even EPWM channels output to be controllable by EPWM_PERIOD0 and EPWM_CMPDAT0 registers and all odd EPWM channels output to be controllable by EPWM_PERIOD1 and EPWM_CMPDAT1 registers. That is, user only needs to set EPWM_CH0 to get EPWM_CH0, EPWM_CH2 and EPWM_CH4 output the same pulse, and set EPWM_CH1 to get EPWM_CH1, EPWM_CH3 and EPWM_CH5 output the same pulse, as shown in Figure 6.12-23. When operating group function, OUTMODE0, OUTMODE2 and OUTMODE4 bits of EPWM_CTL1 register must all set to 0 for independent mode or all set to 1 for complementary mode.

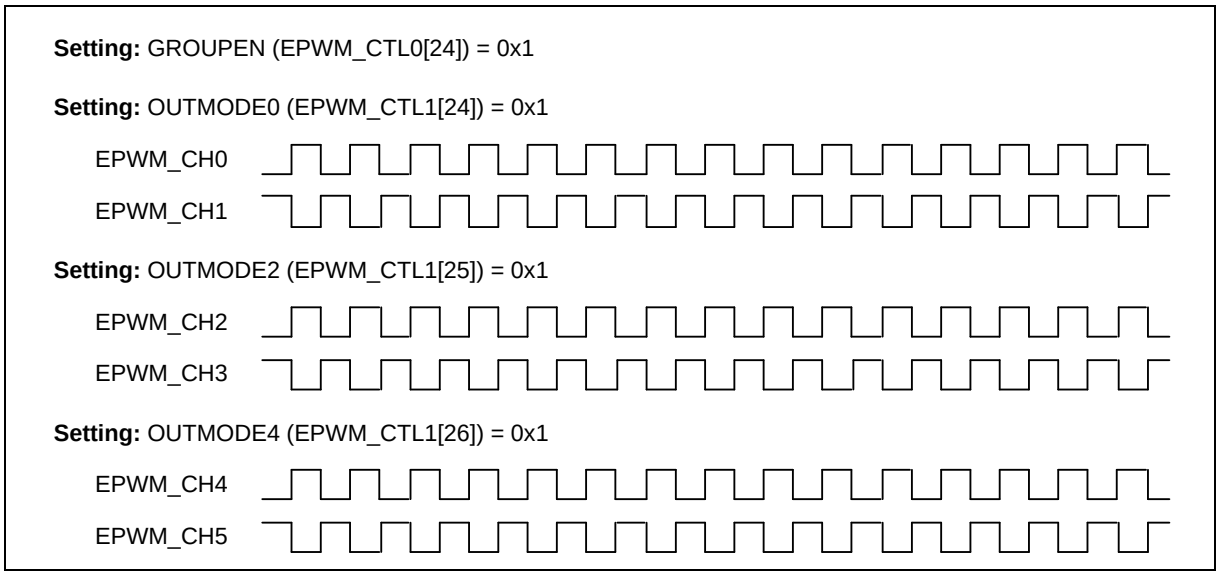


Figure 6.12-23 EPWM Group Function Waveform

6.12.5.19 Synchronous Function

Synchronous function can only be enabled when complementary mode is enabled. Figure 6.12-25 is counter synchronous function block diagram. Every counter of EPWM pairs has a SYNC_IN and a SYNC_OUT signals. The SYNC_IN signal for the first EPWM0 pair counter comes from EPWM0_SYNC_IN pin, and the others come from the SYNC_OUT signal of the previous EPWM pair counter. The input signal from EPWM0_SYNC_IN pin will be filtered by a 3-bit noise filter as Figure 6.12-24. In addition, it can be inversed by setting the bit SINPINV (EPWM_SYNC[23]) to realize the polarity setup for the input signal. The noise filter sampling clock can be selected by setting bits SFLTCSEL (EPWM_SYNC[19:17]) to fit different noise properties. Moreover, by setting the bits SFLTCNT (EPWM_SYNC[22:20]), user can define by how many sampling clock cycles a filter will recognize the effective edge of the SYNC_IN signal. Configuring the SNFLTEN (EPWM_SYNC[16]) will enable the noise filter function. By default, it is disabled.

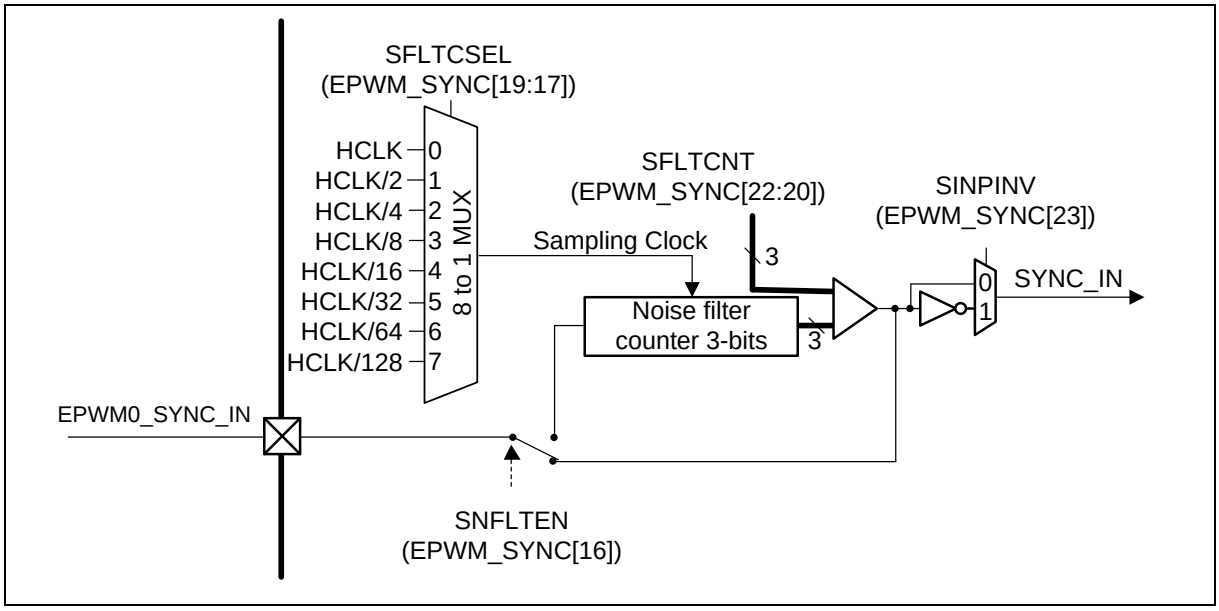


Figure 6.12-24 EPWM SYNC_IN Noise Filter Block Diagram

User can use SINSRCn (EPWM_SYNC[13:8]) bits to select the synchronize source. When SINSRCn bits is set to 0, user can generate SYNC_IN signal for the next counter's synchronization when EPWM0_SYNC_IN pin is high or setting SWSYNCn (EPWM_SWSYNC[2:0]) to 1. Synchronizing source can also be selected as CNT = 0 or CNT = EPWM_CMPDATm register (if being the up-down counter type, it will synchronize twice in a EPWM period) to trigger a sync event or to disable SYNC_OUT signal.

When the PHSENn (EPWM_SYNC[2:0]) is enabled and the synchronous source has a happening event, the counter will load a value from the PHS (EPWM_PHSn_m[15:0]) register. This method synchronizes counters to different phase in the same time. In the up-down counter type, user can set the value in PHSDIRn (EPWM_SYNC[26:24]) to control the counter direction after synchronization. Although the Synchronous function can synchronize channels in phase, it can't work from the beginning of EPWM enable. To start EPWM and BPWM counters in the same time, user have to set the EPWM Synchronous Start Control Register (EPWM_SSCTL[5:0]) to enable the channel counters which are planned to start counting together, and select the SSRC(EPWM_SSCTL[9:8]) to choose the Synchronous Start source, followed by setting the EPWM Synchronous Start Trigger Register CNTSEN (EPWM_SSTRG[0]).

For applications, please do not use Group and Synchronous function simultaneously because the Synchronous function will be inactive.

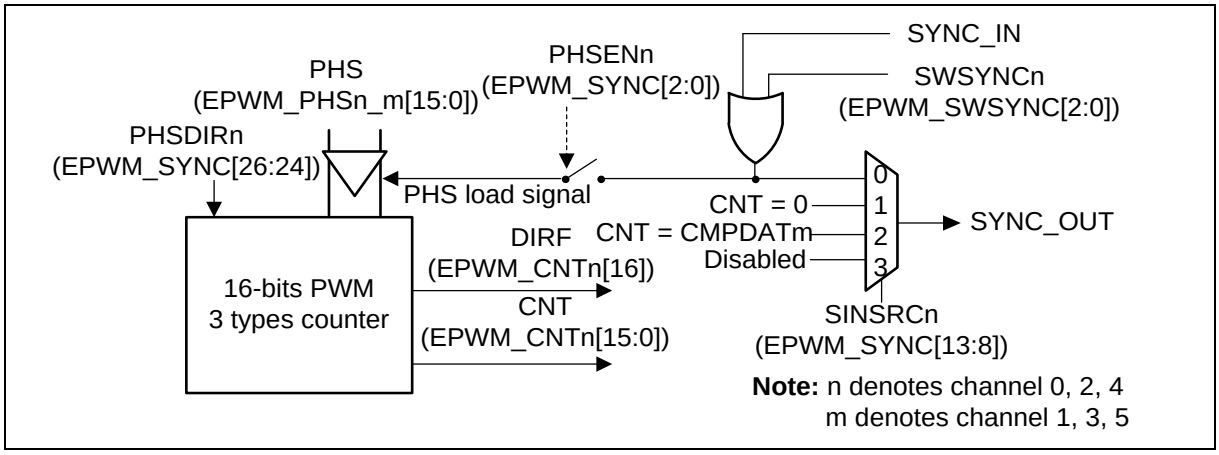


Figure 6.12-25 EPWM Counter Synchronous Function Block Diagram

Figure 6.12-26 is an example of the synchronous function in the up-down counter type. In the example, synchronizing source comes from the external EPWM SYNC_IN signal. At the beginning, the output waveform of EPWM_CH0, EPWM_CH2 and EPWM_CH4 are in the same phase. Then at Point A, the EPWM SYNC input signal comes as a sync event, resulting in phase shifts and counting direction changes for all of the counters. To realize the altered counter behaviors before the sync event coming, user has to setup the corresponding phase value in the PHS of(EPWM_PHSn_m[15:0]) as well as the counting direction in the PHSDIRn (EPWM_SYNC[26:24]). In this case, one third of phase shifts are made. by setting the corresponding channel n's counter counting direction after synchronizing, as illustrated around the left side of Figure 6.12-26.

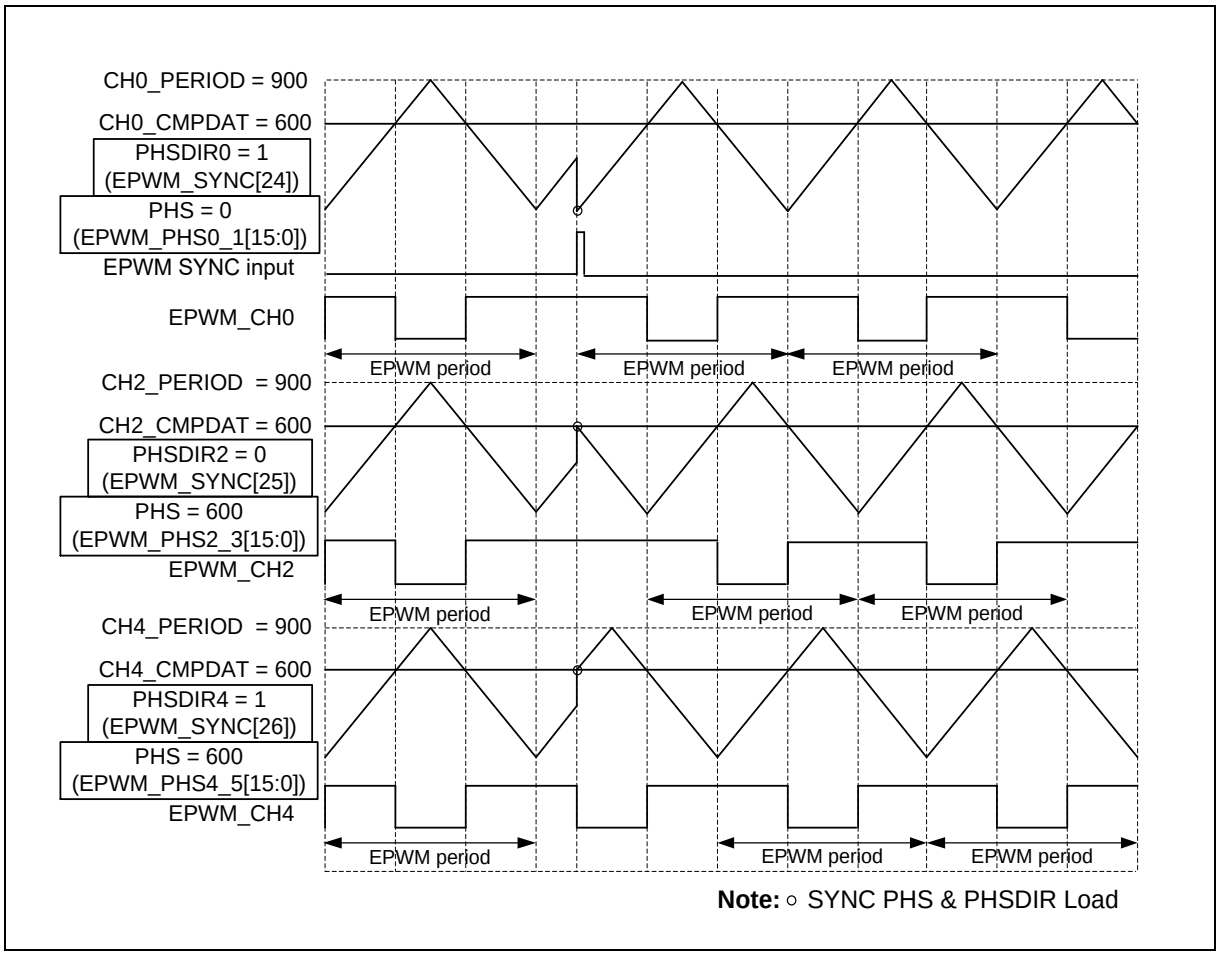


Figure 6.12-26 EPWM Synchronous Function with Synchronize source from SYNC_IN Signal

6.12.5.20 EPWM Output Control

After EPWM pulse generation, there are four to six steps to control the output of EPWM channels. In independent mode, there are Mask, Brake, Pin Polarity and Output Enable four steps as shown in Figure 6.12-27. In complementary mode, it needs two more steps to precede these four steps, Complementary channels and Dead-Time Insertion as shown in Figure 6.12-28.

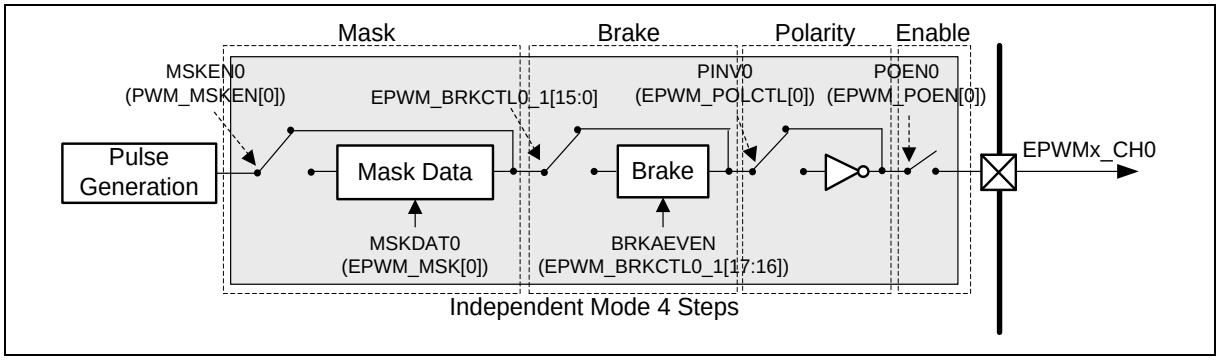


Figure 6.12-27 EPWMx_CH0 Output Control in Independent Mode

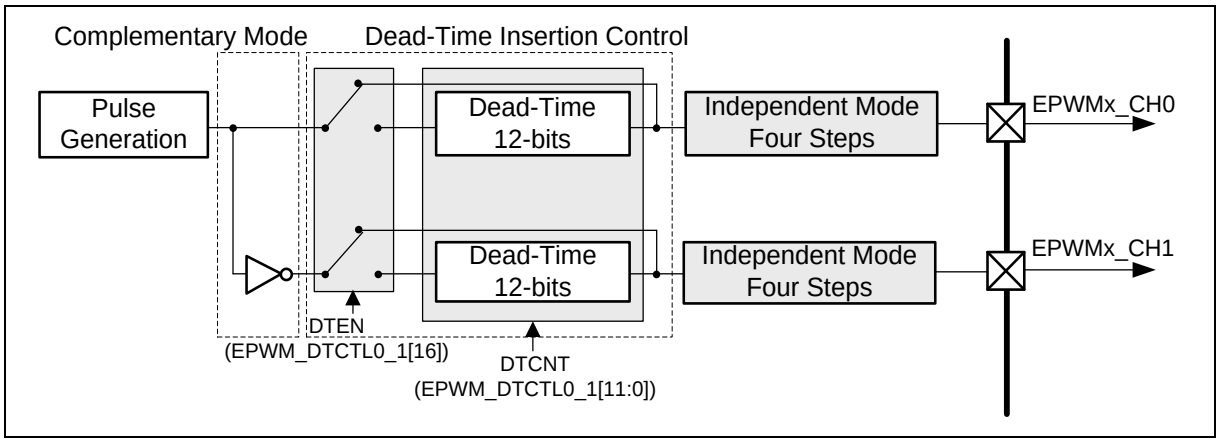


Figure 6.12-28 EPWMx_CH0 and EPWMx_CH1 Output Control in Complementary Mode

6.12.5.21 Dead-Time Insertion

In the complementary application, the complement channels may drive the external devices like power switches. The dead-time generator inserts a low level period called “dead-time” between complementary outputs to drive these devices safely and to prevent system or devices from the burn-out damage. Hence the dead-time control is a crucial mechanism to the proper operation of the complementary system. Set corresponding channel n RDTEN (EPWM_DTCTL[n/2]) bit to enable rising dead-time function and FDTEN (EPWM_DTCTL[n/2+8]) bit to enable falling dead-time function. Set RDTCNT (EPWM_RDTCNTn_m[11:0], n=0,2,4, m=n+1) and FDTCNT (EPWM_FDTCNTn_m[11:0], n=0,2,4, m=n+1) to control dead-time rising and falling period. Dead-time can be calculated from the following formula:

Rising dead-time = (RDTCNT (EPWM_RDTCNTn_m[11:0])+1) * EPWMx_CLK period

Falling dead-time = (FDTCNT (EPWM_FDTCNTn_m[11:0])+1) * EPWMx_CLK period

Dead-time insertion clock source can be selected from prescaler output by setting DTCKSEL n (EPWM_DTCTL[16+n], n=0,2,4) to 1. By default, clock source comes from EPWM_CLK, which is prescaler input. Then the dead-time can be calculated from the following formula:

Rising Dead-time = (RDTCNT (EPWM_RDTCNTn_m[11:0])+1) *

(CLKPSC (EPWM_CLKPSCn[11:0])+1)*EPWMx_CLK period

Falling Dead-time = (FDTCNT (EPWM_FDTCNTn_m[11:0])+1) *

(CLKPSC (EPWM_CLKPSCn[11:0])+1)*EPWMx_CLK period

Please note that the EPWM_RDTCNTn_m, EPWM_FDTCNTn_m and EPWM_DTCTL are write-protected registers.

Figure 6.12-29 indicates the dead-time insertion for one pair of EPWM signals.

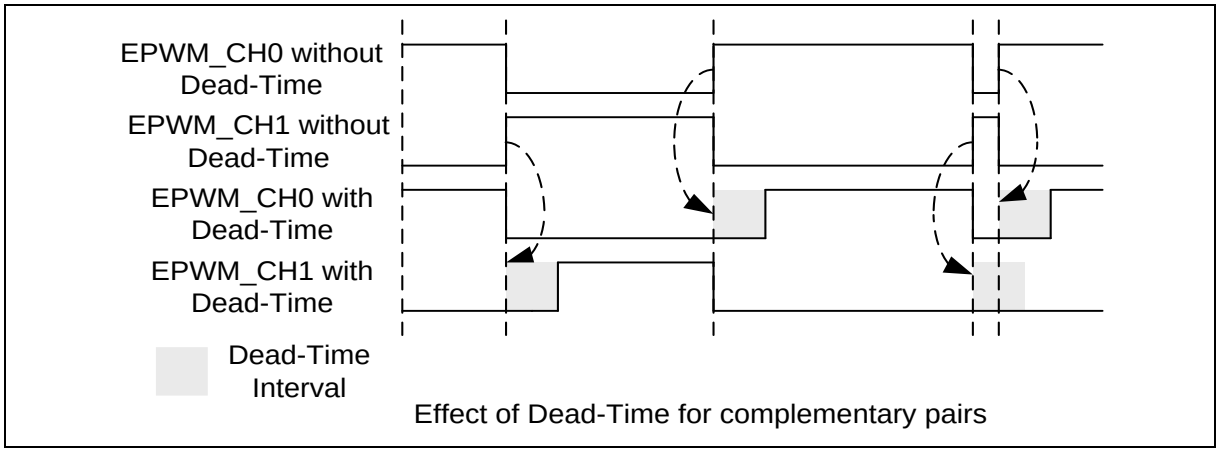


Figure 6.12-29 Dead-Time Insertion

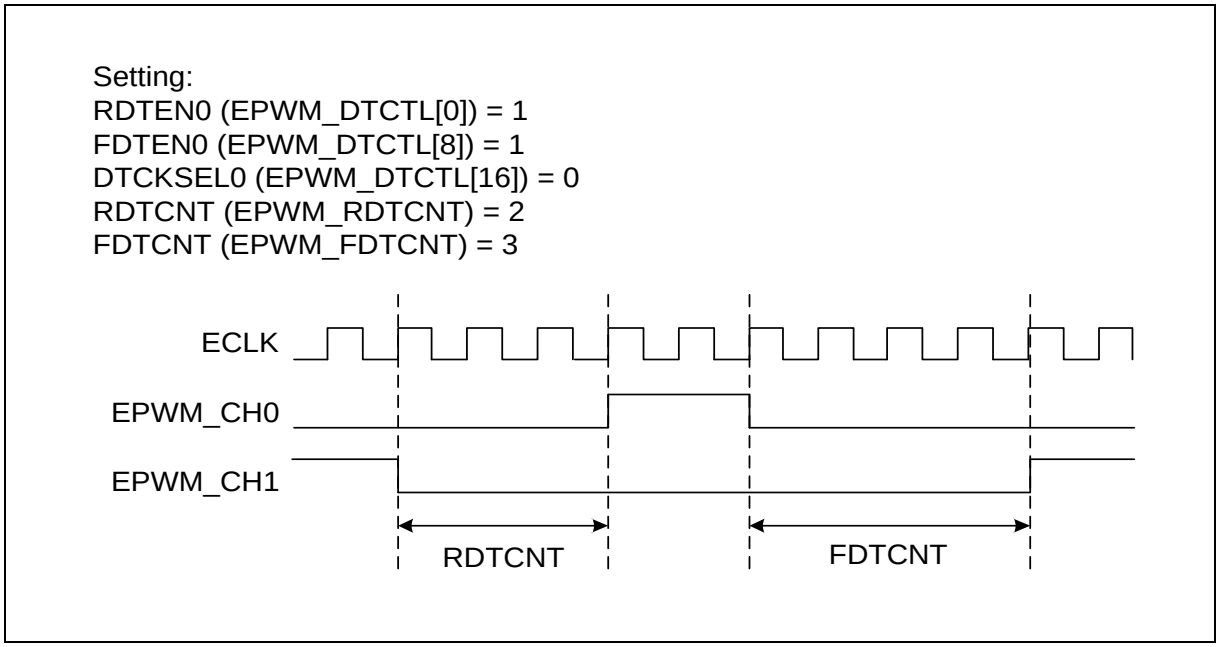


Figure 6.12-30 Dead-Time Independent Control

6.12.5.22 EPWM Mask Output Function

Each of the EPWM channel output value can be manually overridden with the settings in the EPWM

Mask Enable Control Register (EPWM_MSKEN) and the EPWM Masked Data Register (EPWM_MSK) With these settings, the EPWM channel outputs can be assigned to specified logic states independent of the duty cycle comparison units. The EPWM mask bits are useful when controlling various types of Electrically Commutated Motor (ECM) like a BLDC motor. The EPWM_MSKEN register contains six bits, MSKENn(EPWM_MSKEN[5:0]). If the MSKENn is set to active-high, the EPWM channel n output will be overridden. The EPWM_MSK register contains six bits, MSKDATn(EPWM_MSK[5:0]). The bit value of the MSKDATn determines the state value of the EPWM channel n output when the channel is overridden. Figure 6.12-31 shows an example of how EPWM mask control can be used for the override feature.

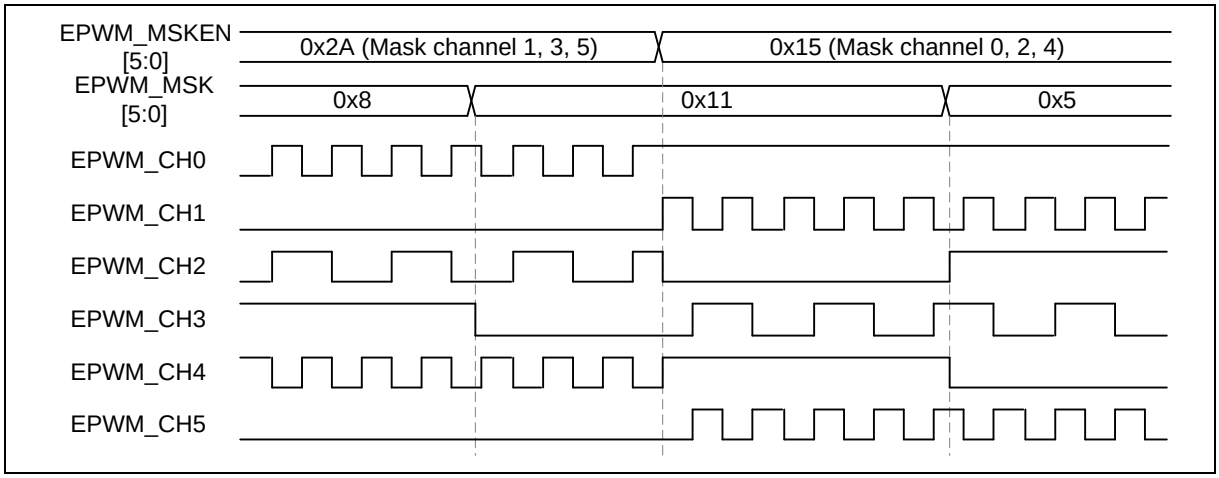


Figure 6.12-31 Illustration of Mask Control Waveform

6.12.5.23 EPWM Brake

Each EPWM module has two external input brake control signals. User can select active brake pin source is from EPWMx_BRAKEy pin by BKxSRC bits of EPWM_BNF register(x=0,1, y=0,1). The external signals will be filtered by a 3-bit noise filter. User can enable the noise filter function by BRKxNFEN bits of EPWM_BNF register, and noise filter sampling clock can be selected by setting BRKxNFSEL bits of EPWM_BNF register to fit different noise properties. Moreover, by setting the BRKxFCNT bits, user can define by how many sampling clock cycles a filter will recognize the effective edge of the brake signal.

In addition, it can be inversed by setting the BRKxPINV (x denotes input external pin 0 or 1) bits of EPWM_BNF register to realize the polarity setup for the brake control signals. Set BRKxPINV bit to 0, brake event will occurr when EPWMx_BRAKEy(x=0,1, y=0,1) pin status is from low to high; set BRKxPINV to 1, brake event will occur when EPWMx_BRAKEy pin status is from high to low.

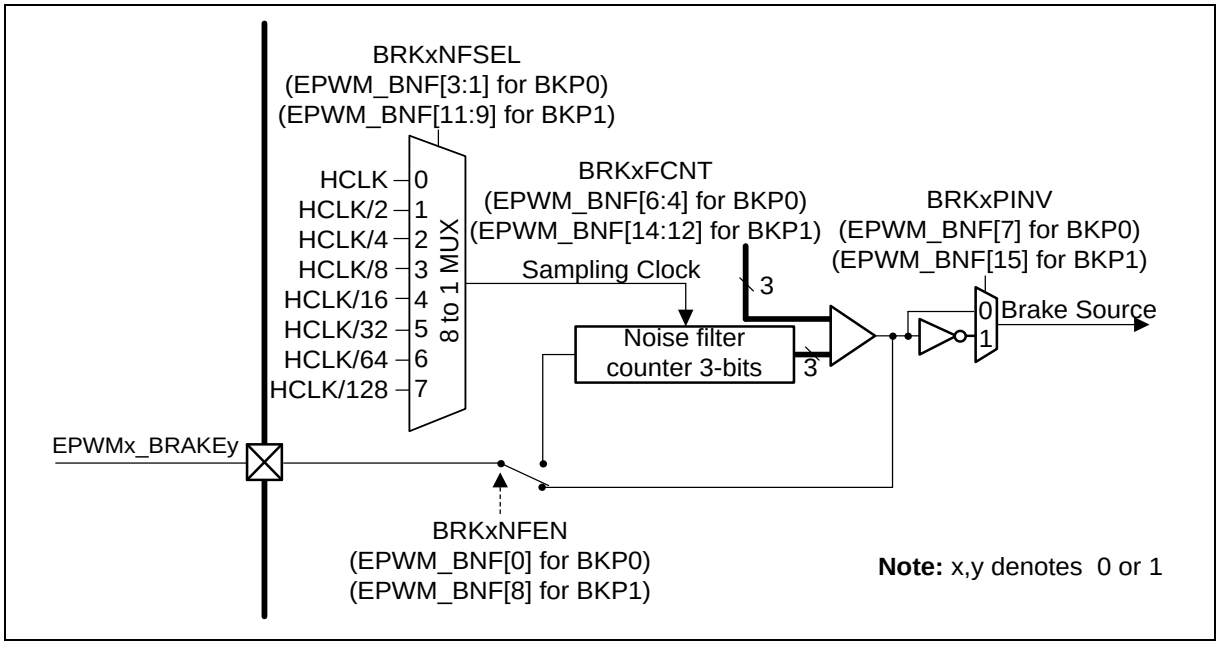


Figure 6.12-32 Brake Noise Filter Block Diagram

For Complementary mode, it is often necessary to set a safe output state to the complement output pairs once the brake event occurs.

Each complementary channel pair shares a EPWM brake function, as shown in Figure 6.12-33. To control paired channels to output safety state, user can setup BRKAEVEN (EPWM_BRKCTL0_1[17:16]) for even channels and BRKAODD (EPWM_BRKCTL0_1[19:18]) for odd channels when the fault brake event happens. There are two brake detectors: Edge detector and Level detector. When the edge detector detects the brake signal and BRKEIENn_m (EPWM_INTEN1[2:0]) is enabled, the brake function generates BRK_INT. This interrupt needs software to clear, and the BRKESTSn (EPWM_INTSTS1[21:16]) brake state will keep until the next EPWM period starts after the interrupt cleared. The brake function can also operate in another way through the level detector. Once the level detector detects the brake signal and the BRKLIENn_m (EPWM_INTEN1[10:8]) is also enabled, the brake function will generate BRK_INT, but BRKLSTSn (EPWM_INTSTS1[29:24]) brake state will auto recovery to normal output while level brake source recovery to high level and pass through “Low Level Detection” at the EPWM waveform period when brake condition removed without clear interrupt.

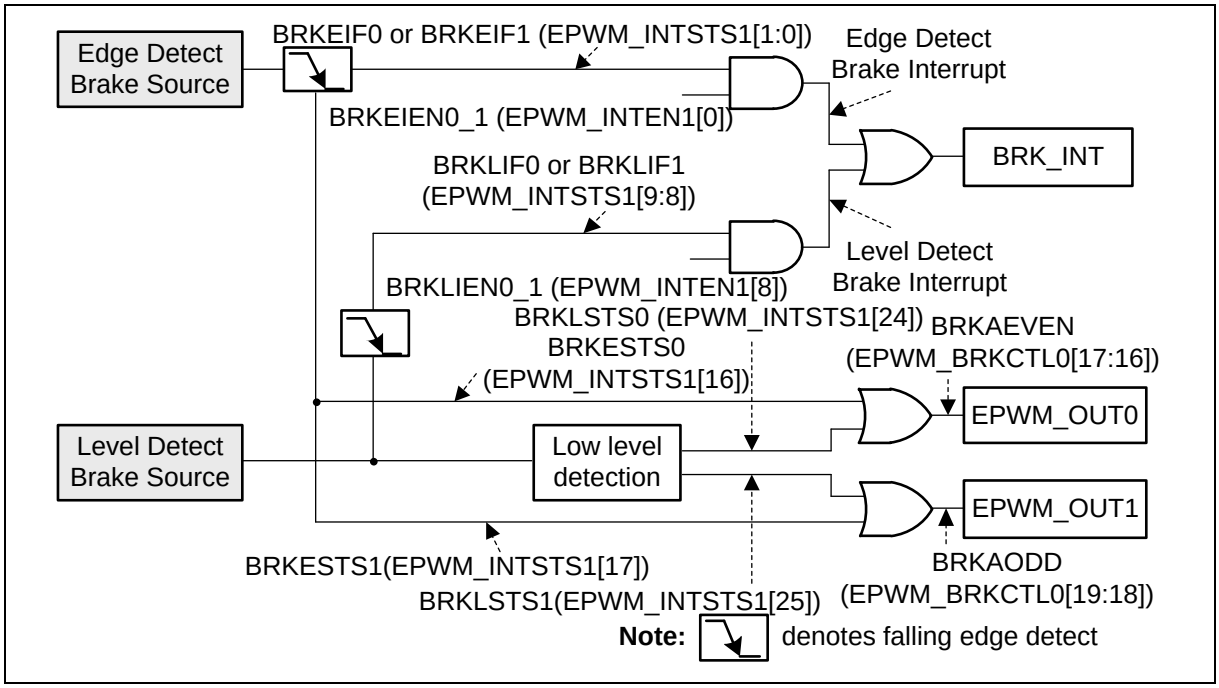


Figure 6.12-33 Brake Block Diagram for EPWMx_CH0 and EPWMx_CH1 Pair

Figure 6.12-34 illustrates the edge detector waveform for EPWMx_CH0 and EPWMx_CH1 pair. In this case, the edge detect brake source has occurred twice for the brake events. When the event occurs, both of the BRKEIF0 and BRKEIF1 flags are set and BRKESTS0 and BRKESTS1 bits are also set to indicate brake state of EPWMx_CH0 and EPWMx_CH1. For the first occurring event, software writes 1 to clear the BRKEIF0 flag. After that, the BRKESTS0 bit is cleared by hardware at the next start of the EPWM period. At the same moment, the EPWMx_CH0 outputs the normal waveform even though the brake event is still occurring. The second event also triggers the same flags, but at this time, software writes 1 to clear the BRKEIF1 flag. Afterward, EPWMx_CH1 outputs normally at the next start of the EPWM period.

As a contrast to the edge detector example, Figure 6.12-35 illustrates the level detector waveform for EPWMx_CH0 and EPWMx_CH1 pair. In this case, the BRKLIF0 and BRKLIF1 flags can only indicate the brake event having occurred. The BRKLSTS0 and BRKLSTS1 brake states will automatically recover at the start of the next EPWM period no matter at what states the BRKLIF0 and BRKLIF1 flags are at that moment.

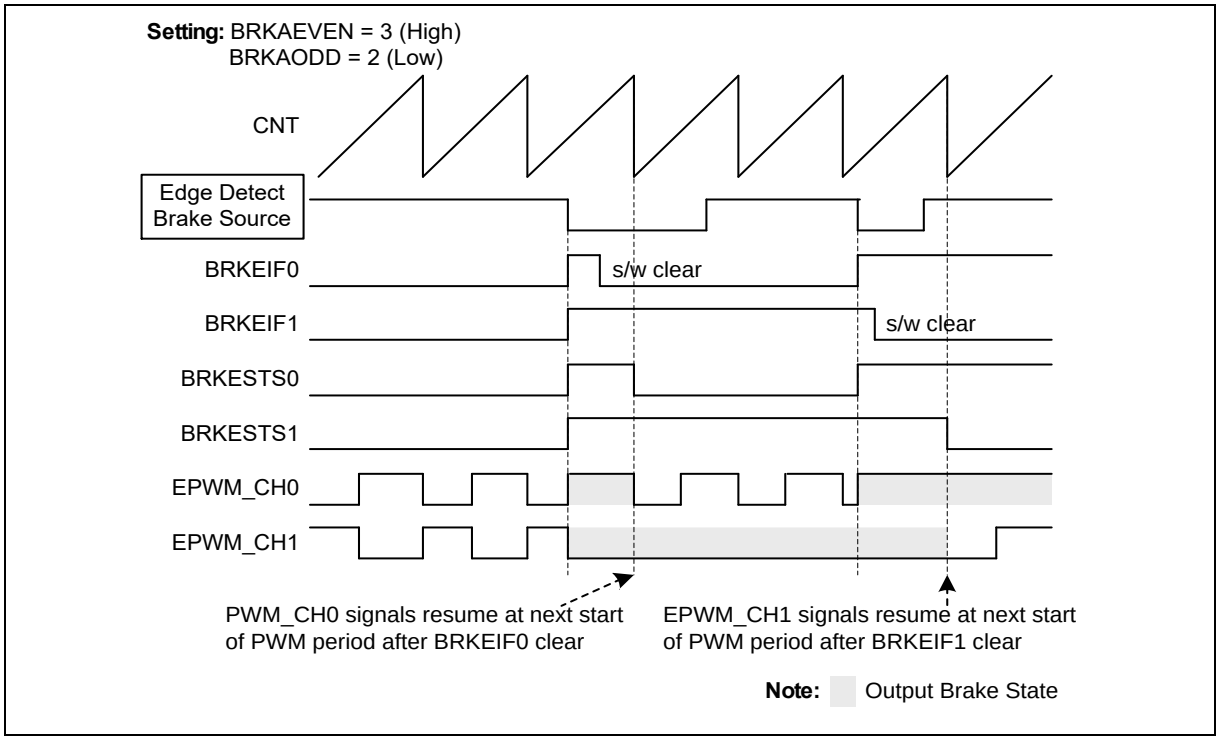


Figure 6.12-34 Edge Detector Waveform for EPWMx_CH0 and EPWMx_CH1 Pair

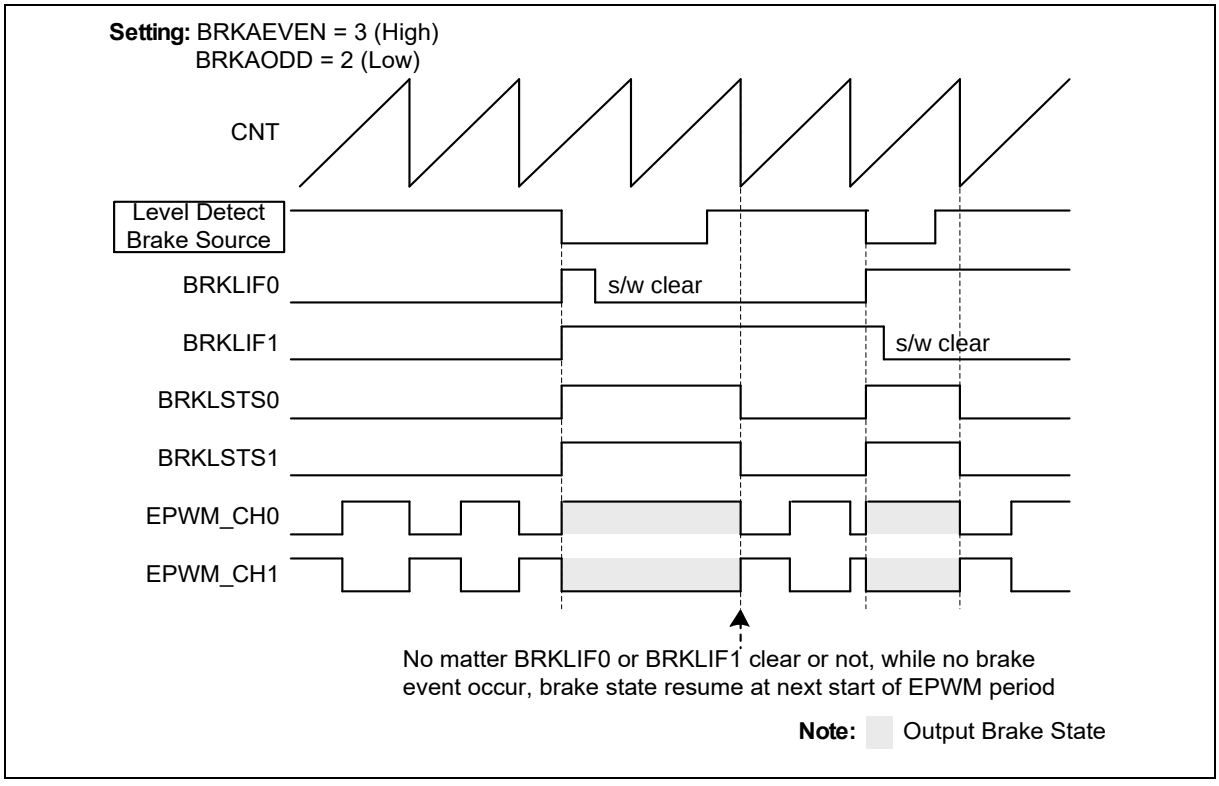


Figure 6.12-35 Level Detector Waveform for EPWMx_CH0 and EPWMx_CH1 Pair

The two kinds of detectors detect the same eleven brake sources: two from external input signals, four from analog comparators(ACMP0~ACMP3), three from EADC result monitor (ADC0RM~ADC2RM),

one from system fail and one from software triggered, that are shown in Figure 6.12-36. ACMP brake sources will be detected only when internal ACMP0_O ~ ACMP3_O signal from low to high.

Among the above described brake sources, the brake source coming from system fail can still be specified to several different system fail conditions. These conditions include clock fail, Brown-out detect, SRAM parity check error and Core lockup. Figure 6.12-37 shows that by setting corresponding enable bits, the enabled system fail condition can be one of the sources to issue the Brake system fail to the EPWM brake.

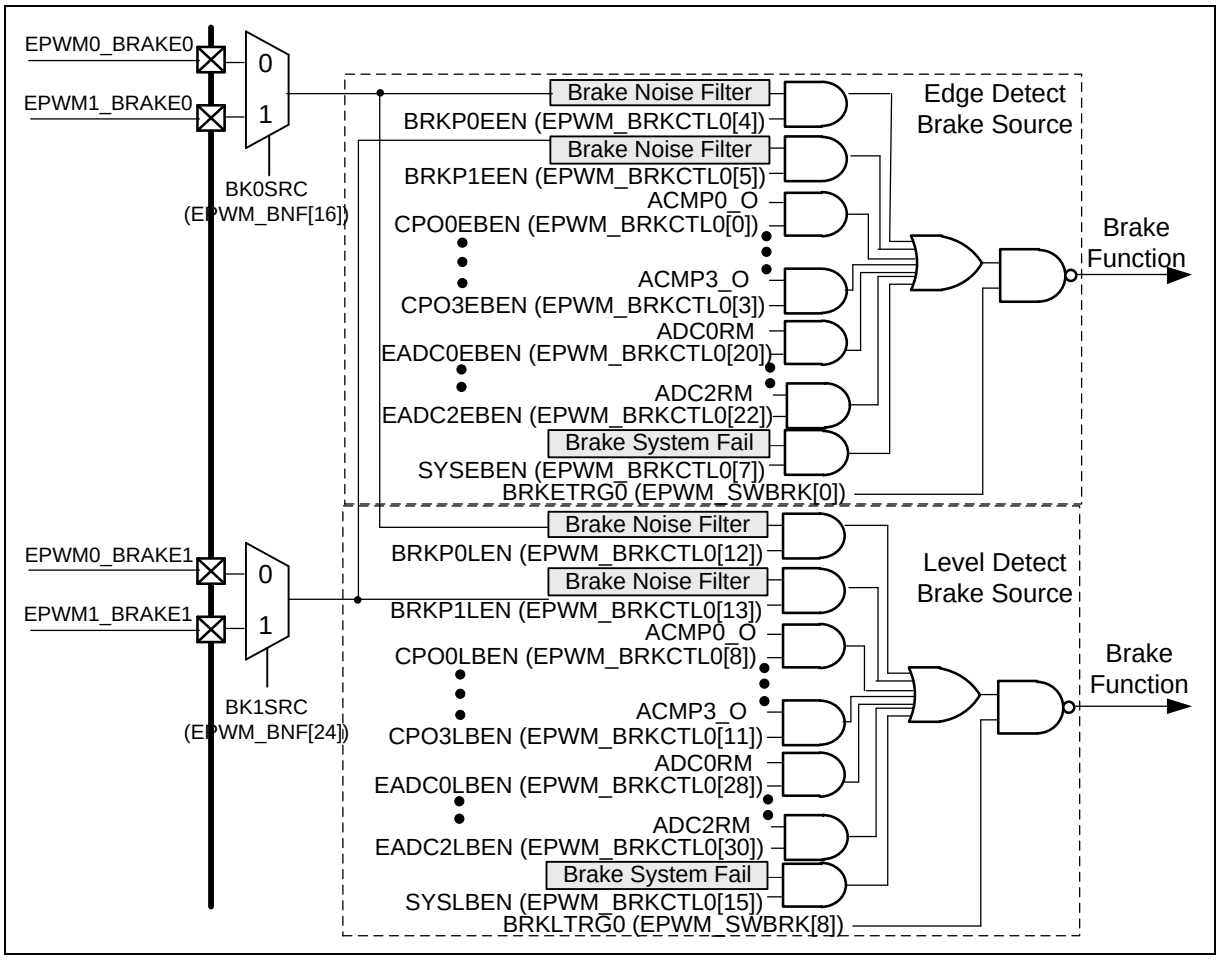


Figure 6.12-36 Brake Source Block Diagram

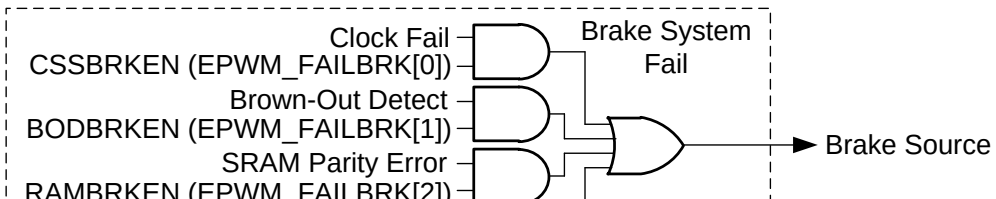


Figure 6.12-37 Brake System Fail Block Diagram

6.12.5.24 LEB Function

Leading edge blanking (LEB) function is use to blank the false trigger from brake source ACMP which may caused by EPWM output transition. Set LEBEN (EPWM_LEBCTL[0]) to enable this function. LEB source comes from EPWM_CH0, EPWM_CH2 and EPWM_CH4, use SRCENn (EPWM_LEBCTL[10:8]) as input source enable. LEB function blanking time is decided by LEBCNT (EPWM_LEBCNT[8:0]), when LEB detected trigger edge, then blanking time will count from LEBCNT+1 to 0, the counter clock base is ECLK. If a new trigger event occurs, blanking counter will reset to LEBCNT and down count again. LEB trigger edge can be rising, falling or both rising and falling edge by setting TRGTYPE (EPWM_LEBCTL[17:16]). Figure 6.12-38 shows that LEB will blanking leading edge caused by EPWM_CH0 and EPWM_CH4.

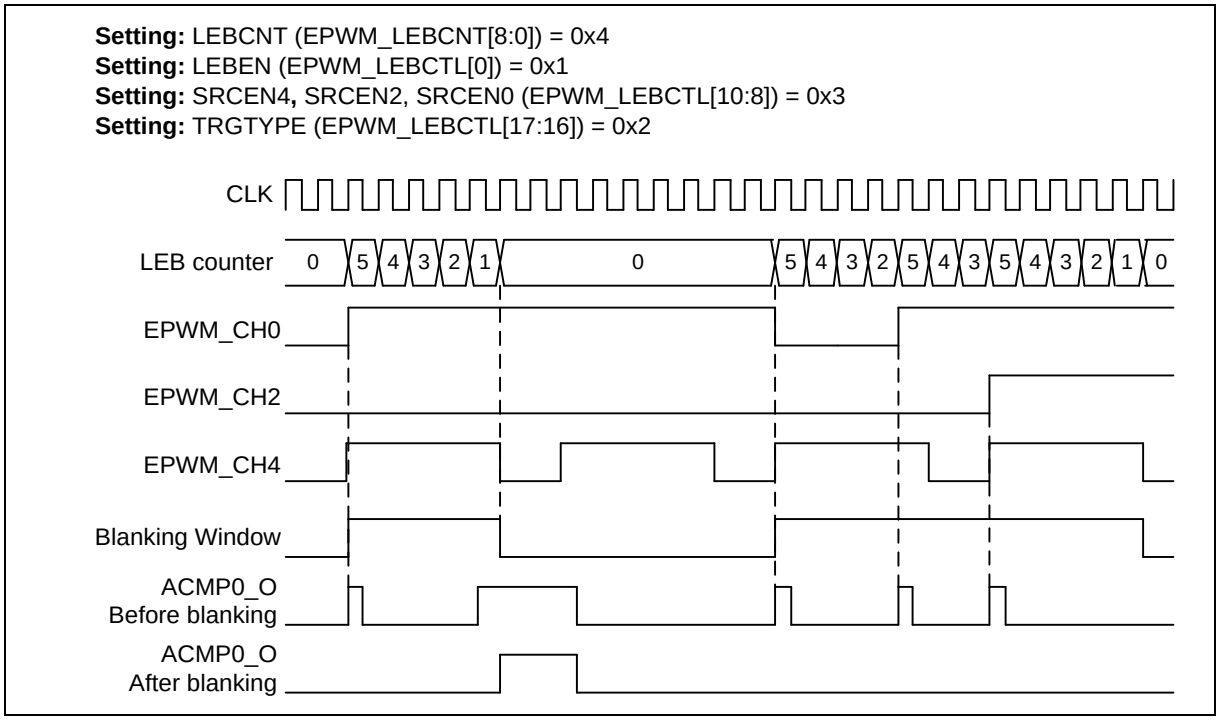
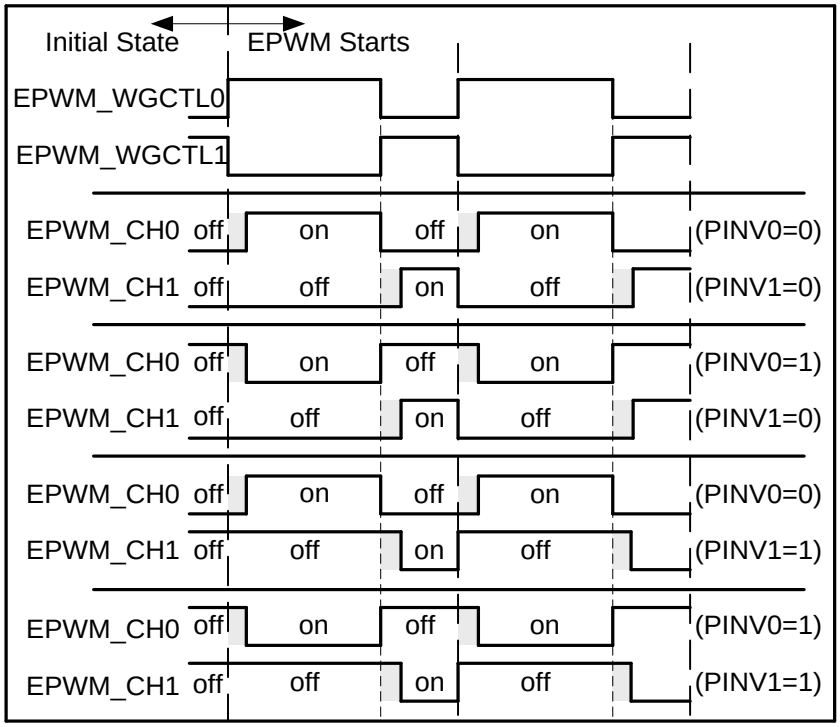


Figure 6.12-38 EPWM LEB Function Waveform

6.12.5.25 Polarity Control

Each EPWM port, from EPWM_CH0 to EPWM_CH5, has an independent polarity control module to configure the polarity of the active state of the EPWM output. By default, the EPWM output is active high. This implies the EPWM OFF state is low and ON state is high. This definition is variable through setting the EPWM Negative Polarity Control Register (EPWM_POLCTL), for each individual EPWM channel. Figure 6.12-39 shows the initial state before EPWM starting with different polarity settings.



Dead-time insertion; It is only effective in complementary mode

Note: PINVx: Negative Polarity control bits; It controls the EPWM output initial state and polarity, x denotes 0 or 1.

Figure 6.12-39 Initial State and Polarity Control with Rising Edge Dead-Time Insertion

6.12.5.26 EPWM Interrupt Generator

Interrupts for each EPWM show in Figure 6.12-41 and Figure 6.12-42.

The 1st EPWM interrupt (EPWM_INT) comes from EPWM complementary pair events. The counter can generate the Zero point Interrupt Flag ZIFn (EPWM_INTSTS0[5:0], n=0,1..5) and the Period point Interrupt Flag PIFn (EPWM_INTSTS0[13:8], n=0,1..5). When EPWM channel n's counter equals to the comparator value stored in EPWM_CMPDATn register, the different interrupt flags will be triggered depending on the counting direction. If the matching occurs at up–count direction, the Up Interrupt Flag CMPUIFn (EPWM_INTSTS0[21:16]) is set and if matching at the opposite direction, the Down Interrupt Flag CMPDIFn (EPWM_INTSTS0[29:24]) is set. If the corresponding interrupt enable bits are set, the trigger events will generates interrupt signals.

EPWM_INT can use the EPWM_IFAn (n=0~5) register to accumulate the number of times that the interrupt flags have been triggered for each channel. By setting one of IFAEN (EPWM_IFAn[31], n=0~5) bit to 1 to enable accumulator, EPWM_INT will switch interrupt source from every event trigger interrupt to trigger interrupt once every accumulate times.

By setting the IFASEL (EPWM_IFAn[29:28], n=0~5) bits, user can select one of the 4 interrupt sources to accumulate interrupt flag times for each channel, and the number of times interrupt flags will compare with IFACNT (EPWM_IFAn[15:0], n=0~5) bits. When interrupt accumulator equals IFACNT then set IFAIFn (EPWM_AINTSTS[n], n=0~5) bits as EPWM_INT signal if user enable IFAIENn (EPWM_AINTEN[n], n=0~5) bits. Accumulator interrupt of each channel can also be as request source of PDMA. Figure 6.10-37 is an example of channel 0 using EPWM_IFA0 register to output EPWM_INT

once every IFCNT0+1 times interrupt events occurred.

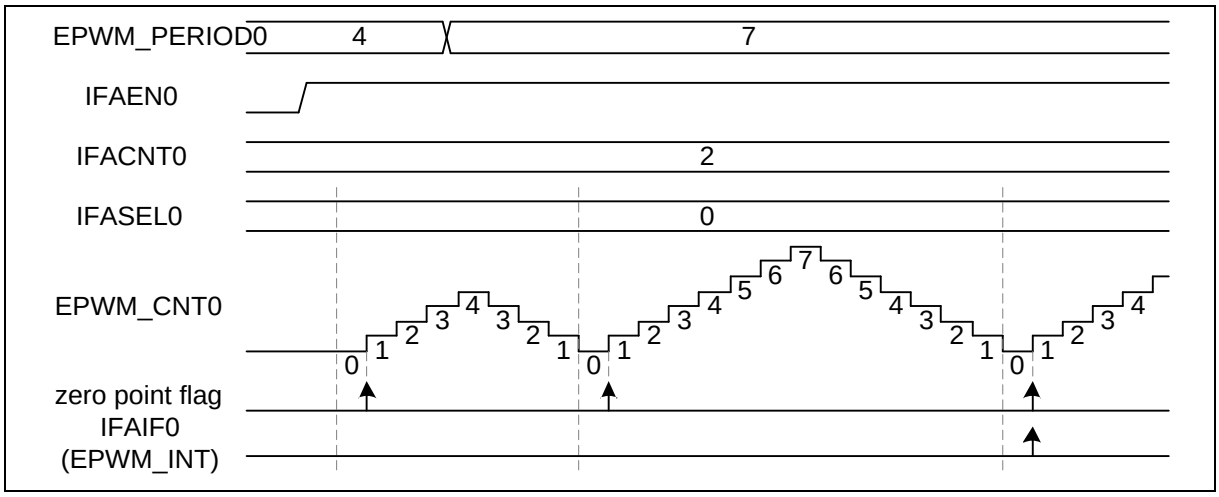


Figure 6.12-40 EPWMx_CH0 Accumulate Interrupt Waveform

The 2nd interrupt is the capture interrupt (CAP_INT). It shares the EPWM_INT vector in NVIC. The CAP_INT can be generated when the CRLIFn (EPWM_CAPIF[5:0]) flag is triggered and the Capture Rising Interrupt Enable bit CAPRIENn (EPWM_CAPIEN[5:0]) is set to 1. Or in the falling edge condition, the CFLIFn (EPWM_CAPIF[13:8]) flag can be triggered when the Capture Falling Interrupt Enable bit CAPFIENn (EPWM_CAPIEN[13:8]) is set to 1.

The 3nd interrupt is the brake interrupt (BRK_INT). The details of the BRK_INT is described in the EPWM Brake section. Figure 6.12-41 demonstrates the architecture of the EPWM interrupts.

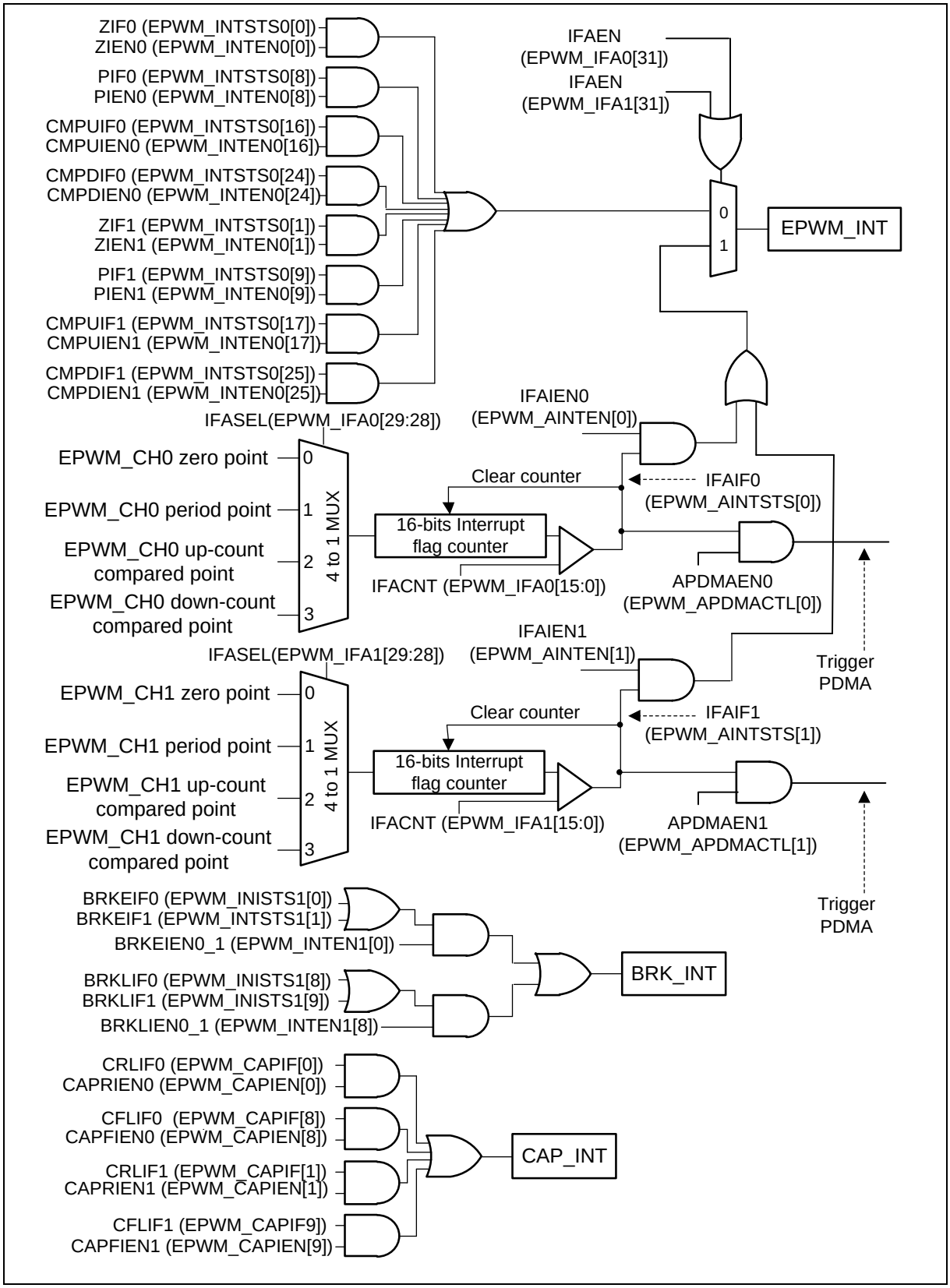


Figure 6.12-41 EPWMx_CH0 and EPWMx_CH1 Pair Interrupt Architecture Diagram

The 4nd interrupt is Fault Detect Function interrupt (FLT_INT). It shares the EPWM_INT vector in NVIC. Figure 6.12-42 shows the FLT_INT can be generate when FDIENn (EPWM_FDIEN[n], n=0~5) is set to 1 and FDIFn (EPWM_FDSTS[n], n=0~5) is set to 1 by detecting output open short.

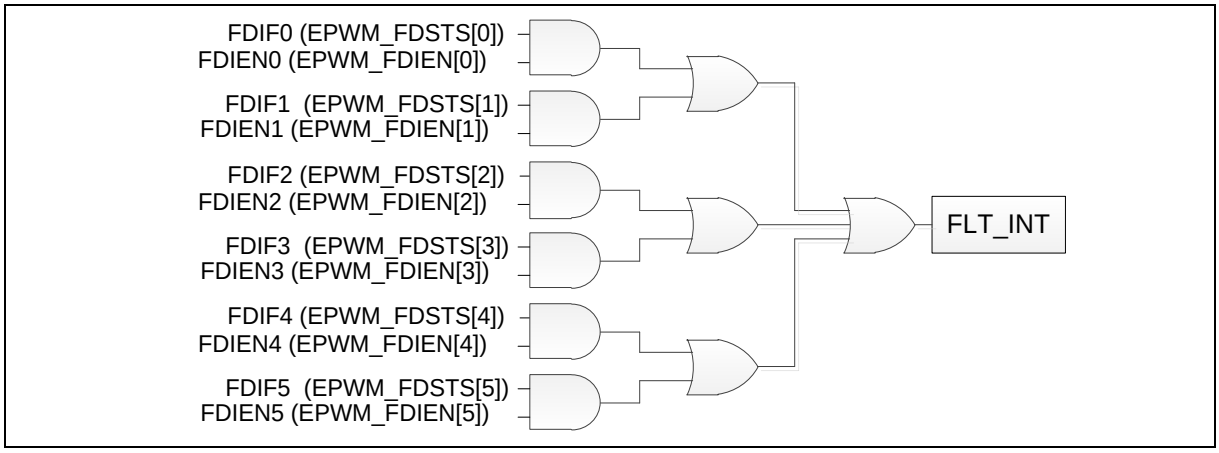


Figure 6.12-42 Fault Detect Function Interrupt Architecture Diagram

6.12.5.27 EPWM Trigger EADC/DAC Generator

EPWM can be one of the EADC conversion trigger source. Each EPWM pair channels share the same trigger source. Setting TRGSELn bit of EPWM_EADCTS0 and EPWM_EADCTS1 registers is to select the trigger sources, where TRGSELn bit is TRGSEL0, TRGSEL1, …, and TRGSEL5, which are located in EPWM_EADCTS0[4:0], EPWM_EADCTS0[12:8], EPWM_EADCTS0[20:16], EPWM_EADCTS0[28:24], EPWM_EADCTS1[4:0] and EPWM_EADCTS1[12:8], respectively. Setting TRGENn bit of EPWM_EADCTS0 and EPWM_EADCTS1 registers is to enable the trigger output to EADC, where TRGENn bit is TRGEN0, TRGEN1, …, TRGEN5, which are located in EPWM_EADCTS0[7], EPWM_EADCTS0[15], EPWM_EADCTS0[23], EPWM_EADCTS0[31], EPWM_EADCTS1[7] and EPWM_EADCTS1[15], respectively. The number n (n = 0,1, ..,5) denotes EPWM channel number.

There are 16 EPWM events can be selected as the trigger source for one pair of channels which shown in Figure 6.12-43. Figure 6.12-44 shows trigger EADC block diagram with prescaler. By setting PSCENn (EPWM_EADCPSCCTR[n], n=0~5) bit to 1, EPWM will trigger EADC when the number of trigger events is equal to EADCPSCn (n=0~5). Reading PSCNTn (n=0~5) can know how many events happened. If PSCENn (n=0~5) is 0, user also can write initial data to PSCNTn (n=0~5) and pre-scale counter will start from this data. If PSCENn (n=0~5) is set from 1 to 0, PSCNTn (n=0~5) will reset to 0. Figure 6.12-45 is the trigger EADC timing waveform in the up-down counter type.

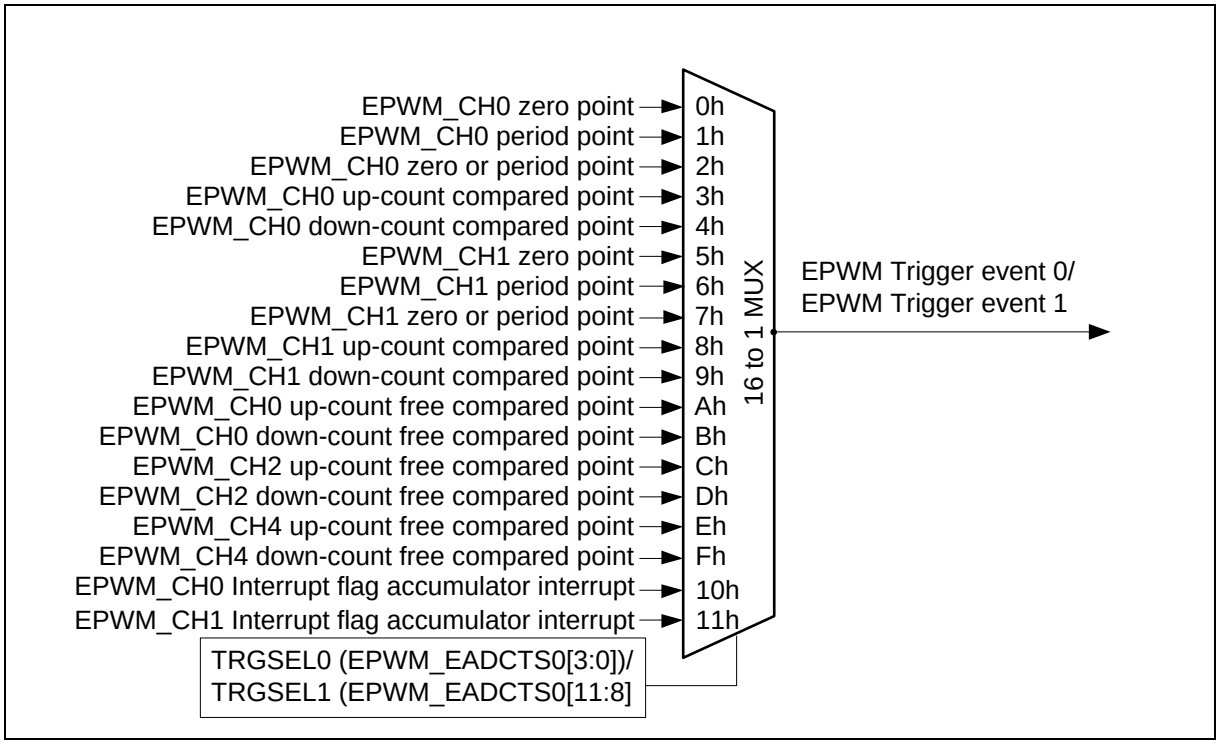


Figure 6.12-43 EPWMx_CH0 and EPWMx_CH1 Pair Trigger EADC Events

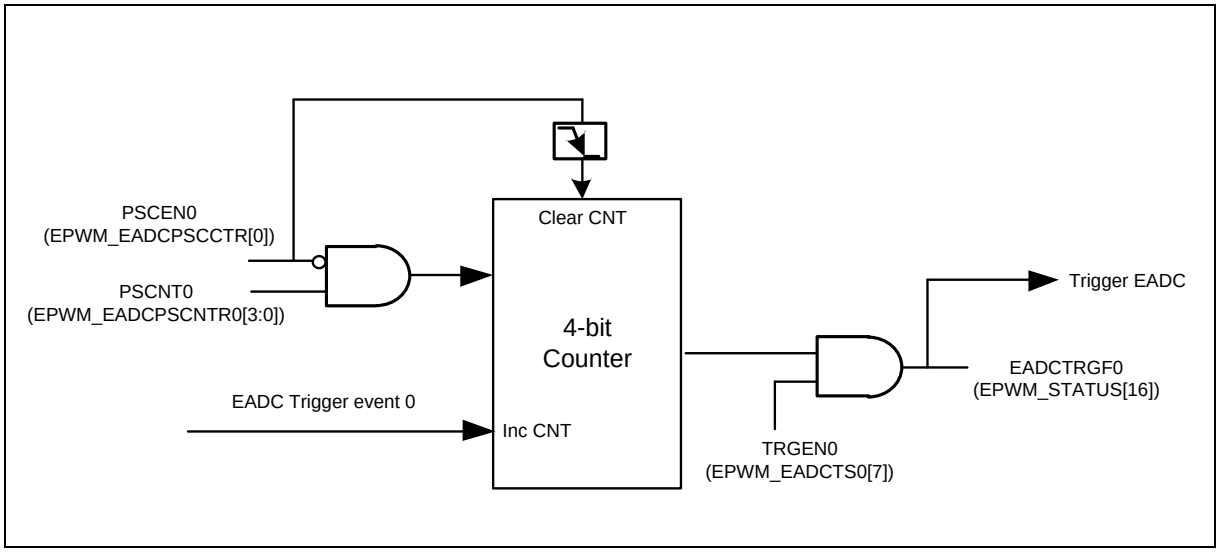


Figure 6.12-44 EPWMx_CH0 Trigger EADC Block Diagram

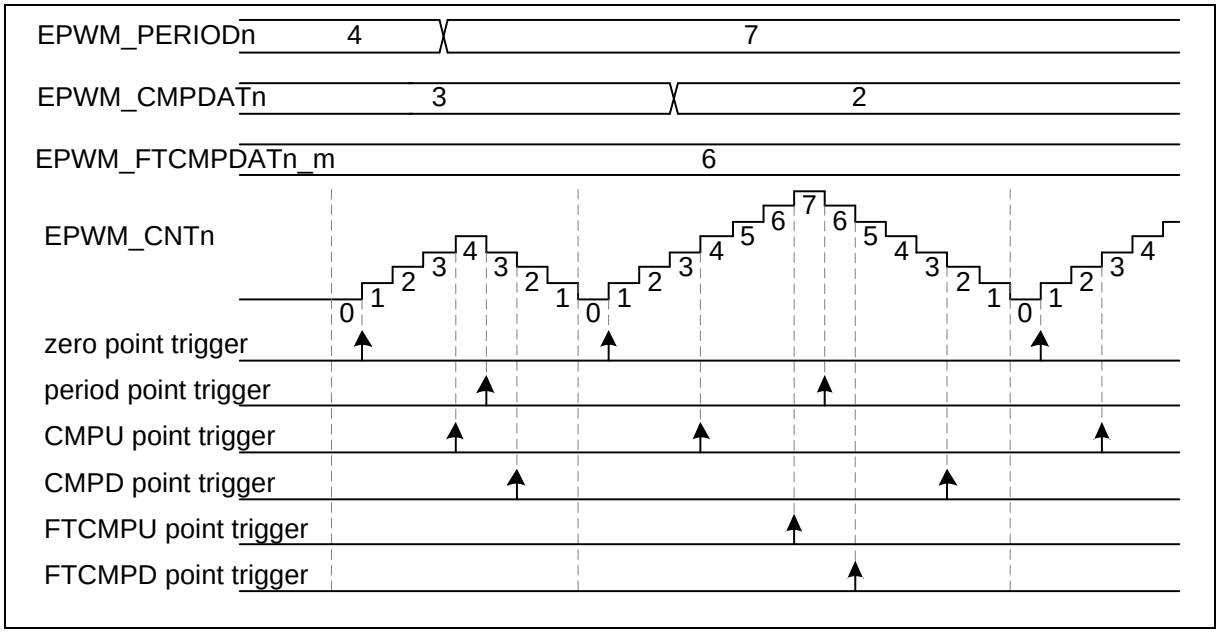


Figure 6.12-45 EPWM Trigger EADC in Up-Down Counter Type Timing Waveform

EPWM can also be used to trigger DAC conversion. Each EPWM pair channel (CH0 and CH1, CH2 and CH3, CH4 and CH5) generates a trigger signal. Using the EPWM Trigger DAC Enable Register (EPWM_DACTRGEN) can decide at which points to trigger DAC. The timing of the EPWM triggerring DAC is similar to those for triggerring EADC. However, DAC triggering function does not include the triggering events from comparison with FTCMPDAT, that is, there are no trigger points the same as FTCMPU and FTCMPD which are shown in EADC triggering.

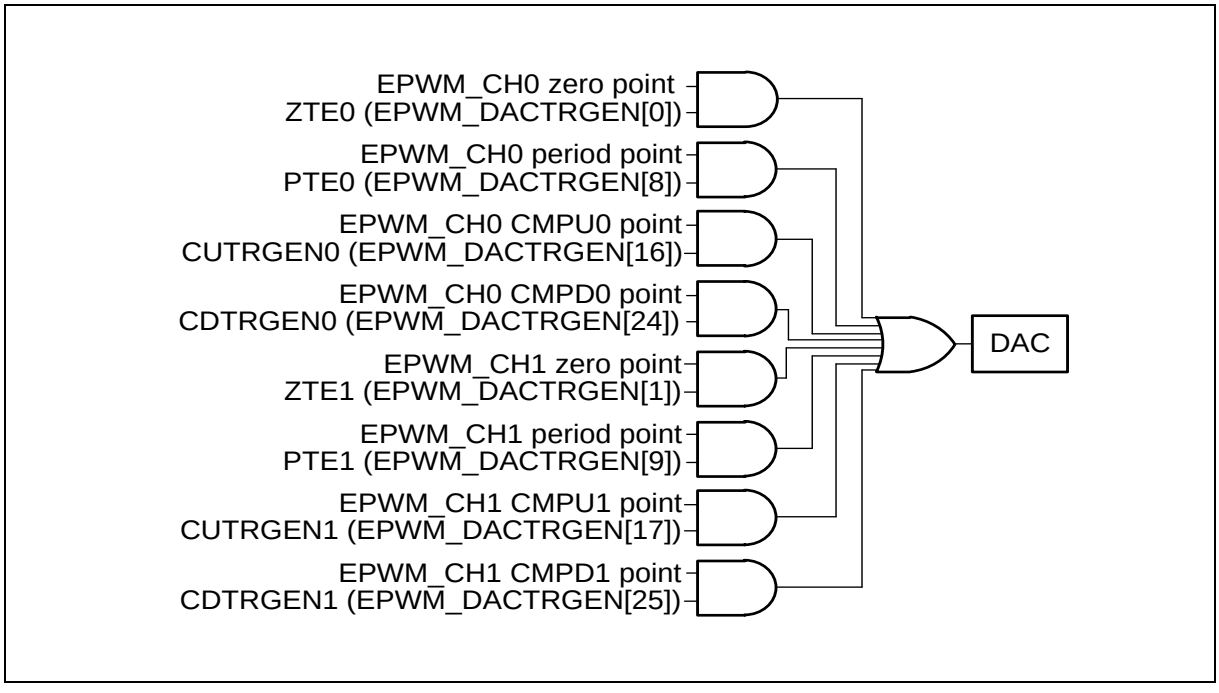


Figure 6.12-46 EPWM_CH0 and EPWM_CH1 Pair Trigger DAC Block Diagram

6.12.5.28 Capture Operation

The channels of the capture input and the EPWM output share the same pin and counter. The counter

can operating in up or down counter type. The capture input will be filtered by a 3-bit noise filter. User can enable the noise filter function by CAPNFEN (EPWM_CAPNFn[0], n=0~5) reigster, and noise filter sampling clock can be selected by setting CAPNFSEL bits of (EPWM_CAPNFn[6:4], n=0~5) register to fit different noise properties. Moreover, by setting the CAPNFCNT (EPWM_CAPNFn[10:8], n=0~5) bits, user can define by how many sampling clock cycles a filter will recognize the effective edge of the capture input signal.The capture function will always latch the EPWM counter to the RCAPDATn (EPWM_RCAPDATn[15:0]) bits or the FCAPDATn (EPWM_FCAPDATn[15:0]) bits, if the input channel has a rising transition or a falling transition, respectively. The capture function will also generate an interrupt CAP_INT (using EPWM_INT vector) if the rising or falling latch occurs and the corresponding channel n's rising or falling interrupt enable bits are set, where the CAPRIENn (EPWM_CAPIEN[5:0]) bit is for the rising edge and the CAPFIENn (EPWM_CAPIEN[13:8]) bit is for the falling edge. When rising or falling latch occurs, the corresponding EPWM counter may be reloaded with the value of EPWM_PERIODn register, depending on the setting of RCRLDENn or FCRLDENn bits (where RCRLDENn and FCRLDENn are located at EPWM_CAPCTL[21:16] and EPWM_CAPCTL[29:24], respectively). Note that the corresponding GPIO pins must be configured as the capture function by enable the CAPINENn (EPWM_CAPINEN[5:0]) bits for the corresponding capture channel n. Figure 6.12-47 is the capture block diagram of channel 0.

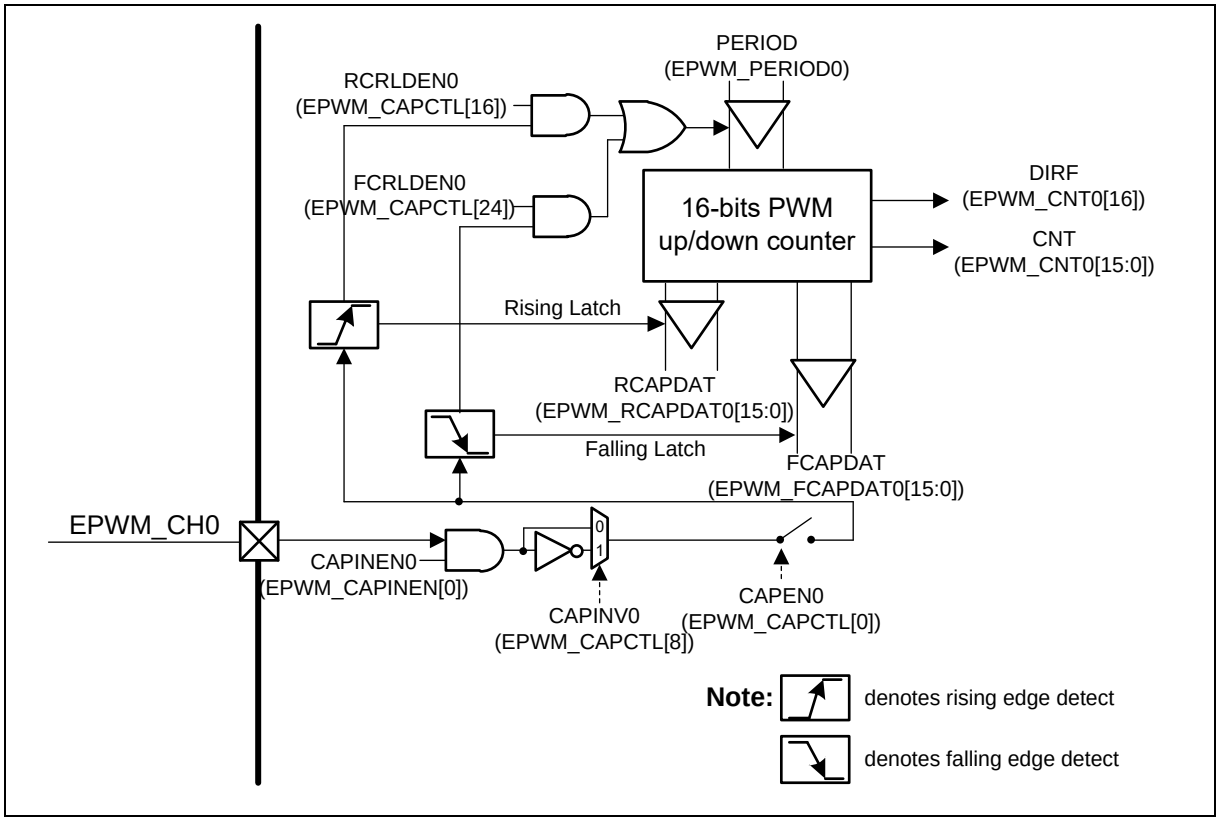


Figure 6.12-47 EPWM_CH0 Capture Block Diagram

Figure 6.12-48 illustrates the capture function timing. In this case, the capture counter is set as EPWM down counter type and the PERIOD is set to 8 so that the counter counts in the down direction, from 8 to 0. When detecting a falling edge at the capture input pin, the capture function latches the counter value to the EPWM_FCAPDATn register. When detecting the rising edge, it latches the counter value to the EPWM_RCAPDATn register. In this timing diagram, when the falling edge is detected at the first time, the capture function will reload the counter value from the PERIOD setting because the FCRLDENn bit is enabled. But at the second time, the falling edge does not result in a reload because of the disabled FCRLDENn bit. In this example, the counter also reloads at the rising edge of the capture input because the RCRLDENn bit is enabled, too.

Moreover, if the case is setup as the up counter type, the counter will reload the value zero and count up to the value PERIOD.

Figure 6.12-48 also illustrates the timing example for the interrupt and interrupt flag generation. When the rising edge at channel n is detected, the corresponding CRLIFn (EPWM_CAPIF[5:0]) bit is set by hardware. Similarly, a falling edge detection at chnnel n causes the corresponding CFLIFn (EPWM_CAPIF[13:8]) bit is set by hardware. CRLIFn and CFLIFn bits can be cleared by software by writing '1'. If the CRLIFn bit is set and the CAPRIENn bit is enabled, the capture function generates an interrupt. If the CFLIFn bit is set and the CAPFIENn bit is enabled, the interrupt also happens.

A condition which is not shown in this figure is: if the rising latch happens again when the CRLIFn bit is already set, the Over run status CRLIFOVn (EPWM_CAPSTS[5:0]) bit will be set to 1 by hardware to indicate the CRLIF flag overrunning. Also, if the falling latch happens again, the same hardware operation occurs for the CFLIF interrupt flag and the Over run status CFLIFOVn (EPWM_CAPSTS[13:8]).

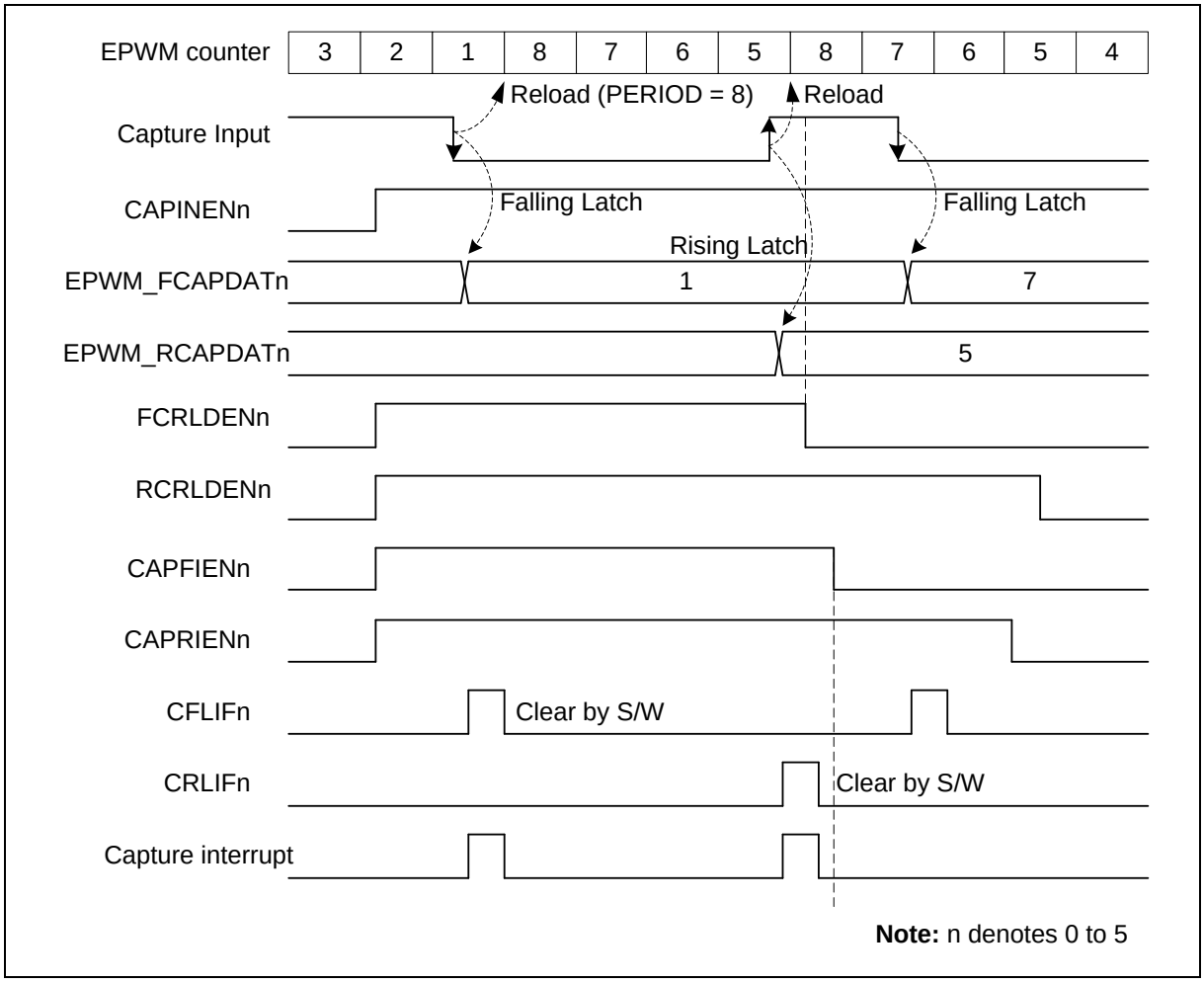


Figure 6.12-48 Capture Operation Waveform

The capture pulse width meeting the following conditions can be calculated according to the formula.

4. The capture positive or negative pulse width is shorter than a counter period.
5. The counter operates in down counter type.
6. The counter can be reloaded by both falling and rising capture events through setting FCRLDENn and RCRLDENn bits of PWM_CAPCTL register to 1.

For the negative pulse case, the channel low pulse width is calculated as (EPWM_PERIODn + 1 -

EPWM_RCAPDATn) EPWM counter time, where one EPWM counter time is (CLKPSC+1) * EPWMx_CLK clock time. In Figure 6.12-48, the low pulse width is 8+1-5 = 4 EPWM counter time.

For the positive pulse case, the channel high pulse width is calculated as (EPWM_PERIODn + 1 - EPWM_FCAPDATn) EPWM counter time, where one EPWM counter time is (CLKPSC+1) * EPWMx_CLK clock time. In Figure 6.12-48, the high pulse width is 8+1-7 = 2 EPWM counter time.

6.12.5.29 Capture PDMA Function

The EPWM module supports the PDMA transfer function when operating in the capture mode. When the corresponding PDMA enable bit CHENn_m (CHEN0_1 at EPWM_PDMACTL[0], CHEN2_3 at EPWM_PDMACTL[8] and CHEN4_5 at EPWM_PDMACTL[16], where n and m denote complement pair channels) is set, the capture module will issue a request to PDMA controller when the preceding capture event has happened. The PDMA controller will issue an acknowledgement to the capture module after it has read back the CAPBUF (EPWM_PDMACAPn_m[15:0], n, m denotes complement pair channels) register in the capture module and has sent the register value to the memory. By setting CAPMODn_m (CAPMOD0_1 at EPWM_PDMACTL[2:1], CAPMOD2_3 at EPWM_PDMACTL[10:9] and CAPMOD4_5 at EPWM_PDMACTL[18:17]) bits, the PDMA can transfer the rising edge captured data or falling edge captured data or both of them to the memory. When using the PDMA to transfer both of the falling and rising edge data, remember to set CAPORDn_m (CAPORD0_1 at EPWM_PDMACTL[3], CAPORD2_3 at EPWM_PDMACTL[11] and CAPORD4_5 at EPWM_PDMACTL[19]) bit to decide the order of the transferred data (falling edge captured is first or rising edge captured first). The complement pair channels share a PDMA channel. Therefore, a selection bit CHSELn_m (CHSEL0_1 (EPWM_PDMACTL[4]), CHSEL2_3 (EPWM_PDMACTL[12]) and CHSEL4_5 (EPWM_PDMACTL[20])) bit is used to decide either channel n or channel m can be serviced by the PDMA channel.

Figure 6.12-49 is capture PDMA waveform. In this case, the CHSEL0_1 (EPWM_PDMACTL[4]) bit is set to 0. Hence the PDMA will service channel 0 for the capture data transfer. CAPMOD0_1 (EPWM_PDMACTL[2:1]) bits are set to 3. That means both of the rising and falling edge captured data will be transferred to the memory. The CAPORD0_1 (EPWM_PDMACTL[3]) bit is set to 1, so the rising edge data will be the first data to transfer and following is the falling edge data to transfer. As shown in Figure 6.12-49, the last assertions of the CRLIF0 and CFLIF0 signal have some overlap. The value of EPWM_RCAPDAT0 register is 11 will be loaded to EPWM_PDMACAP0_1 register to wait for transfer but not the EPWM_FCAPDAT0 value. The EPWM_PDMACAP0_1 register saves the data which will be transferred to the memory by PDMA. The HWDATA in this figure denotes the data which are being transferred by PDMA.

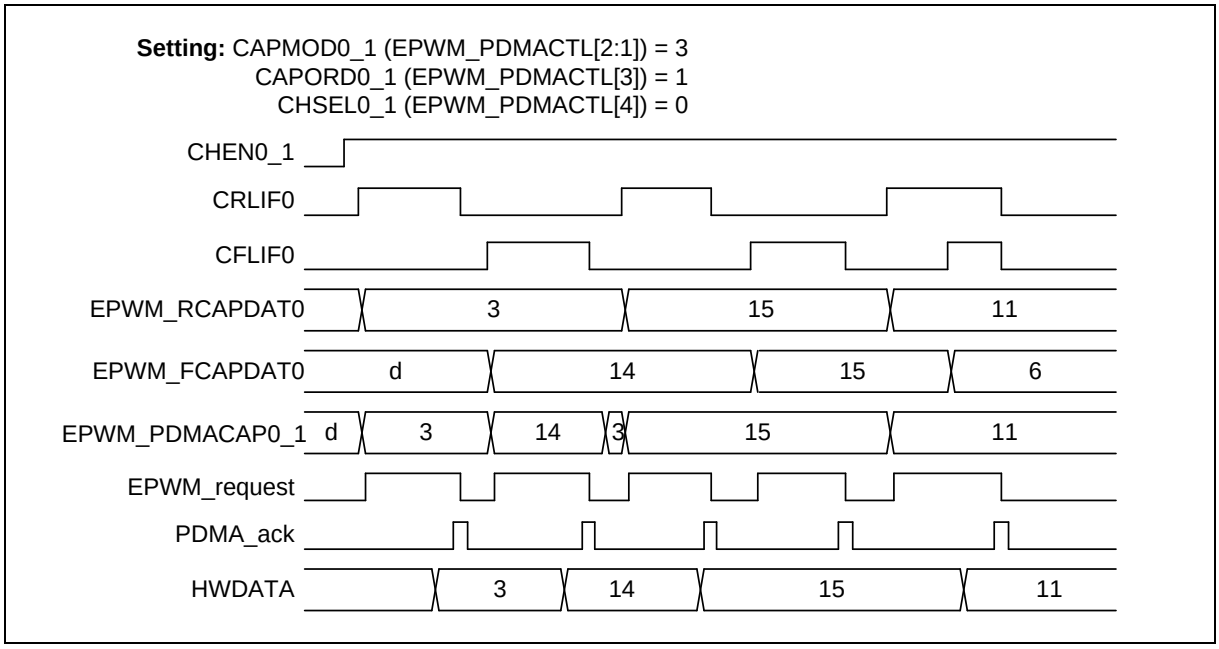


Figure 6.12-49 Capture PDMA Operation Waveform of Channel 0

6.12.5.30 Accumulator PDMA Function

The EPWM module supports the PDMA transfer function when accumulator interrupt happened. Figure 6.12-50 shows accumulator PDMA function architecture. When the corresponding PDMA enable bit APDMAENn (EPWM_APDMACTL[n], n=0~5) is set, accumulator module will send a request to PDMA controller when accumulator interrupt has happened, meaning that IFAIFn (EPWM_AINTSTS[n], n=0~5) is set 1. The PDMA controller will issue an acknowledge to accumulator after it has read memory data and send the data to the particular register (EPWM_PERIOD, etc.). So, user can use this function to change accumulator interrupt frequency.

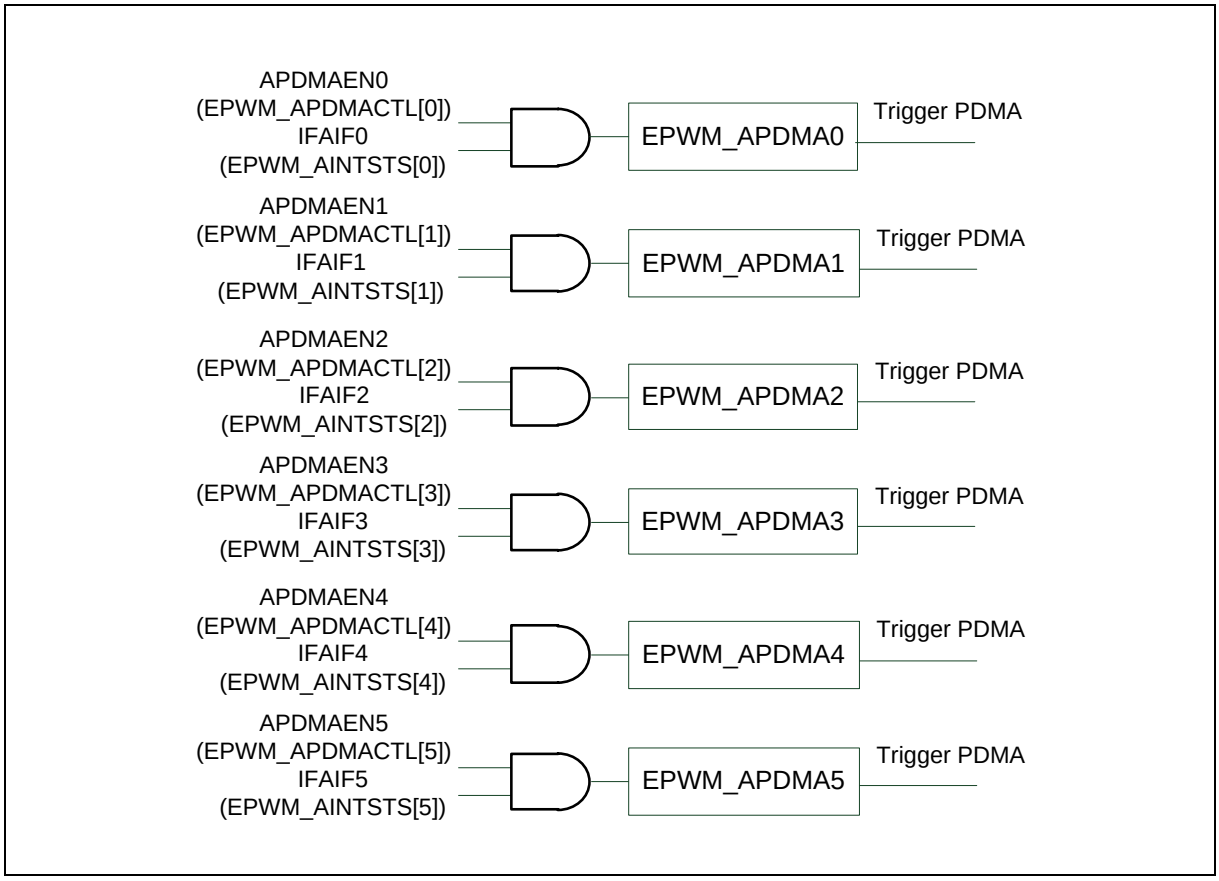


Figure 6.12-50 Accumulator PDMA Function Architecture

6.12.5.31 Accumulator Stop Mode

The EPWM module supports the Accumulator Stop Mode to stop counting when accumulator interrupt happened. Figure 6.10-39 shows Accumulator Stop Mode waveform. When the corresponding STPMOD (EPWM_IFAn[24], n=0~5) is set, accumulator module will disable CNTENn (EPWM_CNTEN[n], n=0~5) after counter finishs whole period.

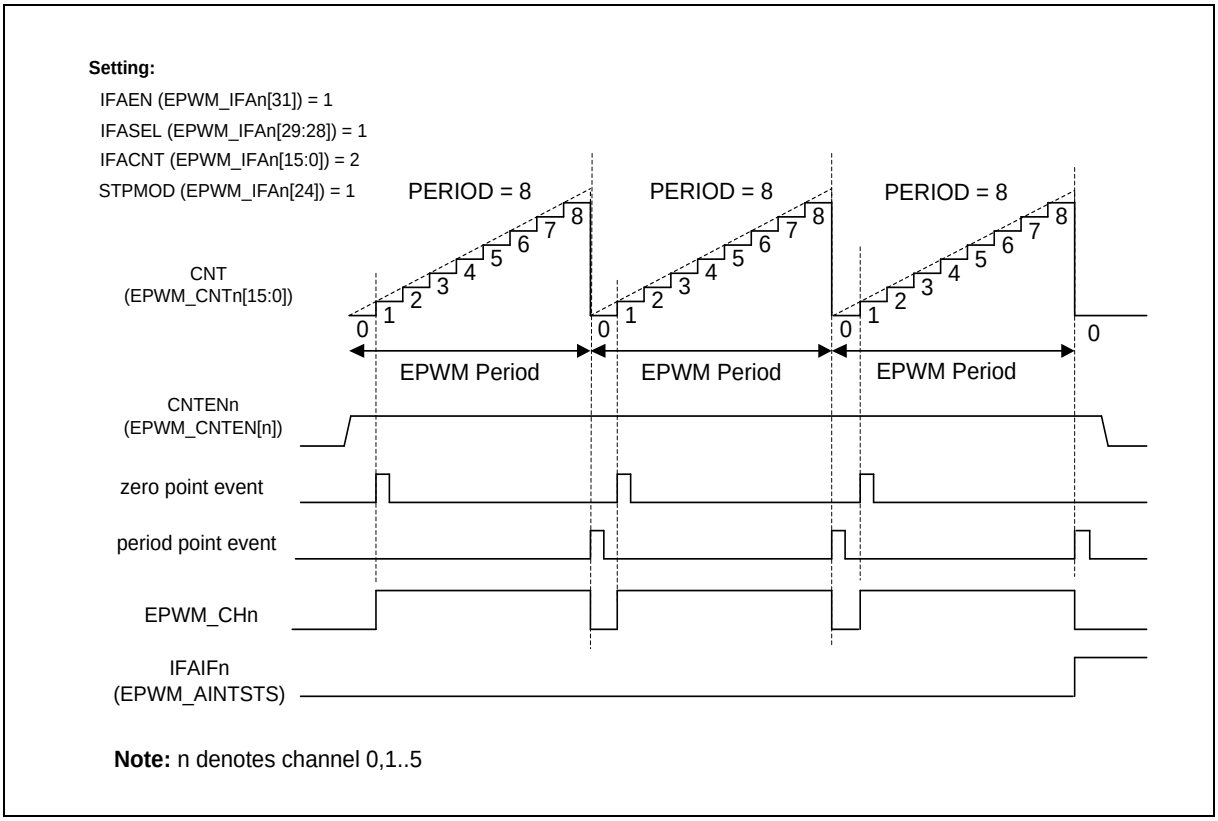


Figure 6.12-51 EPWM Accumulator Stop Mode Waveform

6.12.5.32 Fault Detect Function

The EPWM module supports Fault Detect Function to detect output short condition. Figure 6.12-52 shows fault detect function architecture. When the corresponding channel fault detect enable bit FDENn(EPWM_FDEN[n]) is set to 1, two statement will start detection. One is that EPWM output becomes from low to high or from high to low. The other is that POENn (EPWM_POEN[n], n=0~5) becomes from 0 to 1 or from 1 to 0.

6.12.5.33 External Pin Event Trigger

External pin event can trigger EPWM counter. Setting EXTETEN (EPWM_EXTETCTLn[0], n=0~5) bit to 1 can enable this function. Setting CNTACTS (EPWM_EXTETCTLn[5:4], n=0~5) can choose counter action: reset, start or reset and start. Setting EXTTRGS (EPWM_EXTETCTLn[11:8], n=0~5) can select external trigger source from INT0~INT7.

Trigger event comes from detecting INT0~INT7 rising, falling or both of rising and falling. More detail setting can be found in “INT Edge Detect and Trigger Function” section of the GPIO chapter

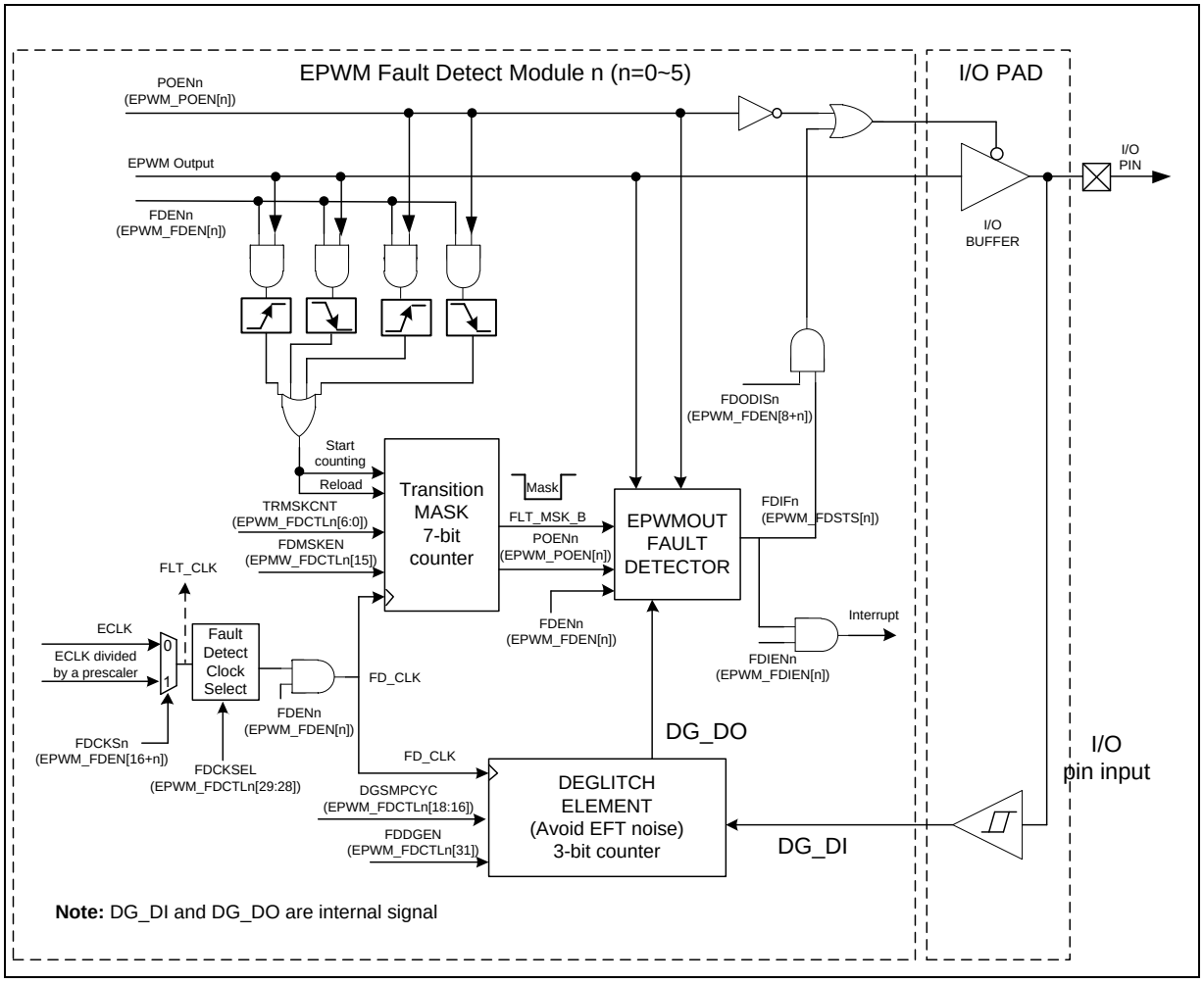


Figure 6.12-52 Fault Detect Function Architecture

Figure 6.12-53 shows Mask Function waveform example. Setting FDMSKEN (EPWM_FDCTLn[15], n=0~5) will enable the MASK Function. The detection will disable in the beginning until mask counter count from 0 to TRMSKCNT (EPWM_FDCTLn[6:0], n=0~5).

Figure 6.12-54 shows Deglitch Function waveform example. Setting FDDGEN (EPWM_FDCTLn[31], n=0~5) will enable Deglitch Function. The feedback signal will be sampled DGSMPCYC (EPWM_FDCTLn[18:16], n=0~5) +1 times and become valid input to be used for detection. FDCKSn (EPWM_FDEN[n], n=0~5) and FDCKSEL (EPWM_FDCTLn[29:28]) are clock setting for MASK and Deglitch Function. For more detail, please see the register description.

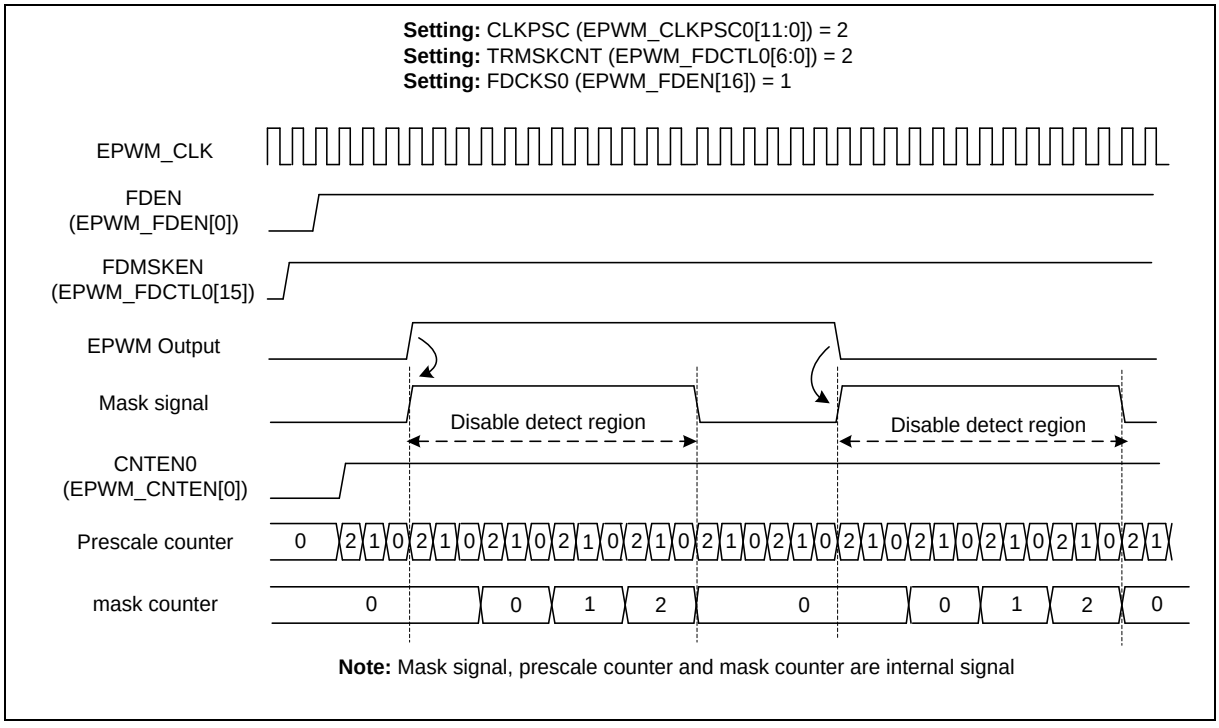


Figure 6.12-53 EPWM Mask Function Waveform

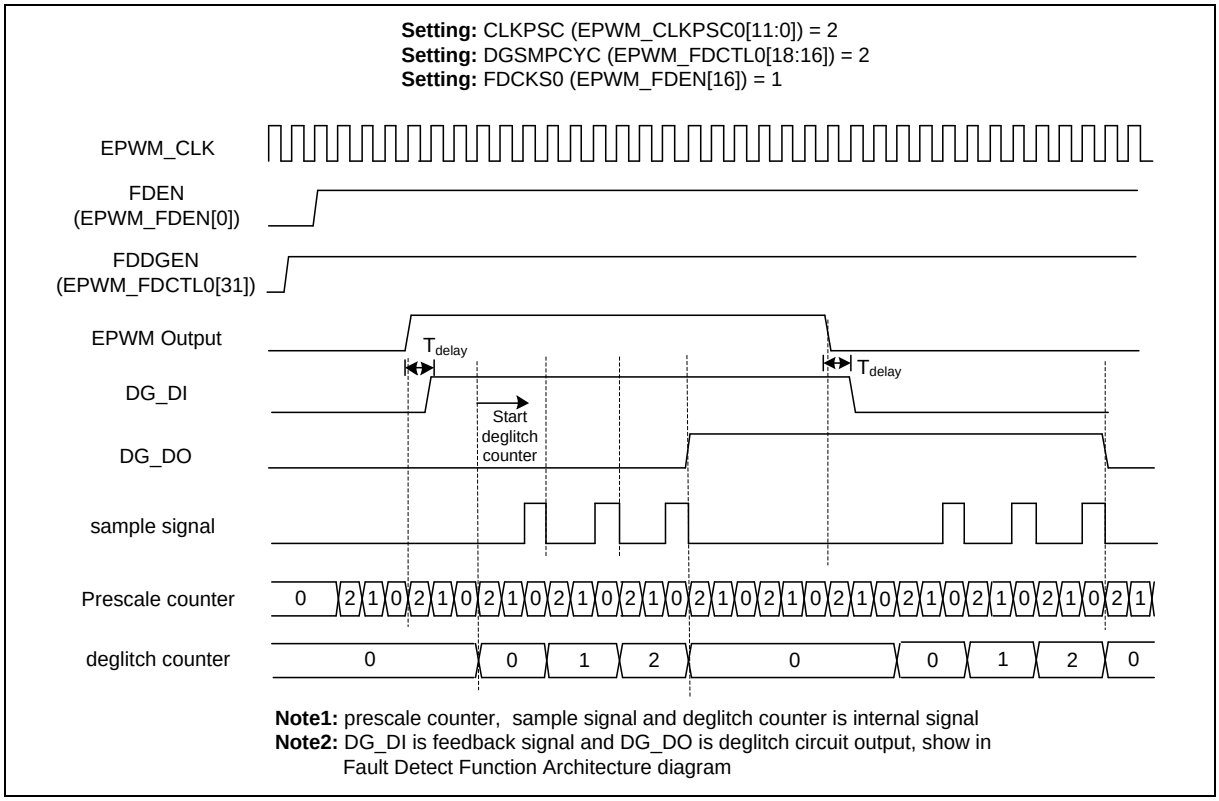


Figure 6.12-54 EPWM Deglitch Function Waveform

6.12.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EPWM Base Address: **EPWM0_BA = 0x4005_8000** **EPWM1_BA = 0x4005_9000**				
EPWM_CTL0 **x=0,1**	EPWMx_BA+0x00	R/W	EPWM Control Register 0	0x0000_0000
EPWM_CTL1 **x=0,1**	EPWMx_BA+0x04	R/W	EPWM Control Register 1	0x0000_0000
EPWM_SYNC **x=0,1**	EPWMx_BA+0x08	R/W	EPWM Synchronization Register	0x0000_0000
EPWM_SWSYNC **x=0,1**	EPWMx_BA+0x0C	W	EPWM Software Control Synchronization Register	0x0000_0000
EPWM_CLKSRC **x=0,1**	EPWMx_BA+0x10	R/W	EPWM Clock Source Register	0x0000_0000
EPWM_CNTEN **x=0,1**	EPWMx_BA+0x20	R/W	EPWM Counter Enable Register	0x0000_0000
EPWM_CNTCLR **x=0,1**	EPWMx_BA+0x24	R/W	EPWM Clear Counter Register	0x0000_0000
EPWM_LOAD **x=0,1**	EPWMx_BA+0x28	R/W	EPWM Load Register	0x0000_0000
EPWM_PERIOD0 **x=0,1**	EPWMx_BA+0x30	R/W	EPWM Period Register 0	0x0000_0000
EPWM_PERIOD1 **x=0,1**	EPWMx_BA+0x34	R/W	EPWM Period Register 1	0x0000_0000
EPWM_PERIOD2 **x=0,1**	EPWMx_BA+0x38	R/W	EPWM Period Register 2	0x0000_0000
EPWM_PERIOD3 **x=0,1**	EPWMx_BA+0x3C	R/W	EPWM Period Register 3	0x0000_0000
EPWM_PERIOD4 **x=0,1**	EPWMx_BA+0x40	R/W	EPWM Period Register 4	0x0000_0000
EPWM_PERIOD5 **x=0,1**	EPWMx_BA+0x44	R/W	EPWM Period Register 5	0x0000_0000
EPWM_CMPDAT0 **x=0,1**	EPWMx_BA+0x50	R/W	EPWM Comparator Register 0	0x0000_0000
EPWM_CMPDAT1 **x=0,1**	EPWMx_BA+0x54	R/W	EPWM Comparator Register 1	0x0000_0000
EPWM_CMPDAT2 **x=0,1**	EPWMx_BA+0x58	R/W	EPWM Comparator Register 2	0x0000_0000
EPWM_CMPDAT3	EPWMx_BA+0x5C	R/W	EPWM Comparator Register 3	0x0000_0000

x=0,1				
EPWM_CMPDAT4 **x=0,1**	EPWMx_BA+0x60	R/W	EPWM Comparator Register 4	0x0000_0000
EPWM_CMPDAT5 **x=0,1**	EPWMx_BA+0x64	R/W	EPWM Comparator Register 5	0x0000_0000
EPWM_PHS0_1 **x=0,1**	EPWMx_BA+0x80	R/W	EPWM Counter Phase Register 0/1	0x0000_0000
EPWM_PHS2_3 **x=0,1**	EPWMx_BA+0x84	R/W	EPWM Counter Phase Register 2/3	0x0000_0000
EPWM_PHS4_5 **x=0,1**	EPWMx_BA+0x88	R/W	EPWM Counter Phase Register 4/5	0x0000_0000
EPWM_CNT0 **x=0,1**	EPWMx_BA+0x90	R	EPWM Counter Register 0	0x0000_0000
EPWM_CNT1 **x=0,1**	EPWMx_BA+0x94	R	EPWM Counter Register 1	0x0000_0000
EPWM_CNT2 **x=0,1**	EPWMx_BA+0x98	R	EPWM Counter Register 2	0x0000_0000
EPWM_CNT3 **x=0,1**	EPWMx_BA+0x9C	R	EPWM Counter Register 3	0x0000_0000
EPWM_CNT4 **x=0,1**	EPWMx_BA+0xA0	R	EPWM Counter Register 4	0x0000_0000
EPWM_CNT5 **x=0,1**	EPWMx_BA+0xA4	R	EPWM Counter Register 5	0x0000_0000
EPWM_WGCTL0 **x=0,1**	EPWMx_BA+0xB0	R/W	EPWM Generation Register 0	0x0000_0000
EPWM_WGCTL1 **x=0,1**	EPWMx_BA+0xB4	R/W	EPWM Generation Register 1	0x0000_0000
EPWM_MSKEN **x=0,1**	EPWMx_BA+0xB8	R/W	EPWM Mask Enable Register	0x0000_0000
EPWM_MSK **x=0,1**	EPWMx_BA+0xBC	R/W	EPWM Mask Data Register	0x0000_0000
EPWM_BNF **x=0,1**	EPWMx_BA+0xC0	R/W	EPWM Brake Noise Filter Register	0x0000_0000
EPWM_FAILBRK **x=0,1**	EPWMx_BA+0xC4	R/W	EPWM System Fail Brake Control Register	0x0000_0000
EPWM_BRKCTL0_1 **x=0,1**	EPWMx_BA+0xC8	R/W	EPWM Brake Edge Detect Control Register 0/1	0x0000_0000
EPWM_BRKCTL2_3 **x=0,1**	EPWMx_BA+0xCC	R/W	EPWM Brake Edge Detect Control Register 2/3	0x0000_0000
EPWM_BRKCTL4_5 **x=0,1**	EPWMx_BA+0xD0	R/W	EPWM Brake Edge Detect Control Register 4/5	0x0000_0000

EPWM_POLCTL x=0,1	EPWMx_BA+0xD4	R/W	EPWM Pin Polar Inverse Register	0x0000_0000
EPWM_POEN x=0,1	EPWMx_BA+0xD8	R/W	EPWM Output Enable Register	0x0000_0000
EPWM_SWBRK x=0,1	EPWMx_BA+0xDC	W	EPWM Software Brake Control Register	0x0000_0000
EPWM_INTEN0 x=0,1	EPWMx_BA+0xE0	R/W	EPWM Interrupt Enable Register 0	0x0000_0000
EPWM_INTEN1 x=0,1	EPWMx_BA+0xE4	R/W	EPWM Interrupt Enable Register 1	0x0000_0000
EPWM_INTSTS0 x=0,1	EPWMx_BA+0xE8	R/W	EPWM Interrupt Flag Register 0	0x0000_0000
EPWM_INTSTS1 x=0,1	EPWMx_BA+0xEC	R/W	EPWM Interrupt Flag Register 1	0x0000_0000
EPWM_DACTRGEN x=0,1	EPWMx_BA+0xF4	R/W	EPWM Trigger DAC Enable Register	0x0000_0000
EPWM_EADCTS0 x=0,1	EPWMx_BA+0xF8	R/W	EPWM Trigger EADC Source Select Register 0	0x0000_0000
EPWM_EADCTS1 x=0,1	EPWMx_BA+0xFC	R/W	EPWM Trigger EADC Source Select Register 1	0x0000_0000
EPWM_FTCMPDAT0_1 x=0,1	EPWMx_BA+0x100	R/W	EPWM Free Trigger Compare Register 0/1	0x0000_0000
EPWM_FTCMPDAT2_3 x=0,1	EPWMx_BA+0x104	R/W	EPWM Free Trigger Compare Register 2/3	0x0000_0000
EPWM_FTCMPDAT4_5 x=0,1	EPWMx_BA+0x108	R/W	EPWM Free Trigger Compare Register 4/5	0x0000_0000
EPWM_SSCTL x=0,1	EPWMx_BA+0x110	R/W	EPWM Synchronous Start Control Register	0x0000_0000
EPWM_SSTRG x=0,1	EPWMx_BA+0x114	W	EPWM Synchronous Start Trigger Register	0x0000_0000
EPWM_LEBCTL x=0,1	EPWMx_BA+0x118	R/W	EPWM Leading Edge Blanking Control Register	0x0000_0000
EPWM_LEBCNT x=0,1	EPWMx_BA+0x11C	R/W	EPWM Leading Edge Blanking Counter Register	0x0000_0000
EPWM_STATUS x=0,1	EPWMx_BA+0x120	R/W	EPWM Status Register	0x0000_0000
EPWM_IFA0 x=0,1	EPWMx_BA+0x130	R/W	EPWM Interrupt Flag Accumulator Register 0	0x0000_0000
EPWM_IFA1 x=0,1	EPWMx_BA+0x134	R/W	EPWM Interrupt Flag Accumulator Register 1	0x0000_0000
EPWM_IFA2	EPWMx_BA+0x138	R/W	EPWM Interrupt Flag Accumulator Register 2	0x0000_0000

x=0,1				
EPWM_IFA3 **x=0,1**	EPWMx_BA+0x13C	R/W	EPWM Interrupt Flag Accumulator Register 3	0x0000_0000
EPWM_IFA4 **x=0,1**	EPWMx_BA+0x140	R/W	EPWM Interrupt Flag Accumulator Register 4	0x0000_0000
EPWM_IFA5 **x=0,1**	EPWMx_BA+0x144	R/W	EPWM Interrupt Flag Accumulator Register 5	0x0000_0000
EPWM_AINTSTS **x=0,1**	EPWMx_BA+0x150	R/W	EPWM Accumulator Interrupt Flag Register	0x0000_0000
EPWM_AINTEN **x=0,1**	EPWMx_BA+0x154	R/W	EPWM Accumulator Interrupt Enable Register	0x0000_0000
EPWM_APDMACTL **x=0,1**	EPWMx_BA+0x158	R/W	EPWM Accumulator PDMA Control Register	0x0000_0000
EPWM_FDEN **x=0,1**	EPWMx_BA+0x160	R/W	EPWM Fault Detect Enable Register	0x0000_0000
EPWM_FDCTL0 **x=0,1**	EPWMx_BA+0x164	R/W	EPWM Fault Detect Control Register 0	0x0000_0000
EPWM_FDCTL1 **x=0,1**	EPWMx_BA+0x168	R/W	EPWM Fault Detect Control Register 1	0x0000_0000
EPWM_FDCTL2 **x=0,1**	EPWMx_BA+0x16C	R/W	EPWM Fault Detect Control Register 2	0x0000_0000
EPWM_FDCTL3 **x=0,1**	EPWMx_BA+0x170	R/W	EPWM Fault Detect Control Register 3	0x0000_0000
EPWM_FDCTL4 **x=0,1**	EPWMx_BA+0x174	R/W	EPWM Fault Detect Control Register 4	0x0000_0000
EPWM_FDCTL5 **x=0,1**	EPWMx_BA+0x178	R/W	EPWM Fault Detect Control Register 5	0x0000_0000
EPWM_FDIEN **x=0,1**	EPWMx_BA+0x17C	R/W	EPWM Fault Detect Interrupt Enable Register	0x0000_0000
EPWM_FDSTS **x=0,1**	EPWMx_BA+0x180	R/W	EPWM Fault Detect Interrupt Flag Register	0x0000_0000
EPWM_EADCPSCCTL **x=0,1**	EPWMx_BA+0x184	R/W	EPWM Trigger EADC Prescale Control Register	0x0000_0000
EPWM_EADCPSC0 **x=0,1**	EPWMx_BA+0x188	R/W	EPWM Trigger EADC Prescale Register 0	0x0000_0000
EPWM_EADCPSC1 **x=0,1**	EPWMx_BA+0x18C	R/W	EPWM Trigger EADC Prescale Register 1	0x0000_0000
EPWM_EADCPSCNT0 **x=0,1**	EPWMx_BA+0x190	R/W	EPWM Trigger EADC Prescale Counter Register 0	0x0000_0000
EPWM_EADCPSCNT1 **x=0,1**	EPWMx_BA+0x194	R/W	EPWM Trigger EADC Prescale Counter Register 1	0x0000_0000

EPWM_CAPINEN x=0,1	EPWMx_BA+0x200	R/W	EPWM Capture Input Enable Register	0x0000_0000
EPWM_CAPCTL x=0,1	EPWMx_BA+0x204	R/W	EPWM Capture Control Register	0x0000_0000
EPWM_CAPSTS x=0,1	EPWMx_BA+0x208	R	EPWM Capture Status Register	0x0000_0000
EPWM_RCAPDAT0 x=0,1	EPWMx_BA+0x20C	R	EPWM Rising Capture Data Register 0	0x0000_0000
EPWM_FCAPDAT0 x=0,1	EPWMx_BA+0x210	R	EPWM Falling Capture Data Register 0	0x0000_0000
EPWM_RCAPDAT1 x=0,1	EPWMx_BA+0x214	R	EPWM Rising Capture Data Register 1	0x0000_0000
EPWM_FCAPDAT1 x=0,1	EPWMx_BA+0x218	R	EPWM Falling Capture Data Register 1	0x0000_0000
EPWM_RCAPDAT2 x=0,1	EPWMx_BA+0x21C	R	EPWM Rising Capture Data Register 2	0x0000_0000
EPWM_FCAPDAT2 x=0,1	EPWMx_BA+0x220	R	EPWM Falling Capture Data Register 2	0x0000_0000
EPWM_RCAPDAT3 x=0,1	EPWMx_BA+0x224	R	EPWM Rising Capture Data Register 3	0x0000_0000
EPWM_FCAPDAT3 x=0,1	EPWMx_BA+0x228	R	EPWM Falling Capture Data Register 3	0x0000_0000
EPWM_RCAPDAT4 x=0,1	EPWMx_BA+0x22C	R	EPWM Rising Capture Data Register 4	0x0000_0000
EPWM_FCAPDAT4 x=0,1	EPWMx_BA+0x230	R	EPWM Falling Capture Data Register 4	0x0000_0000
EPWM_RCAPDAT5 x=0,1	EPWMx_BA+0x234	R	EPWM Rising Capture Data Register 5	0x0000_0000
EPWM_FCAPDAT5 x=0,1	EPWMx_BA+0x238	R	EPWM Falling Capture Data Register 5	0x0000_0000
EPWM_PDMACTL x=0,1	EPWMx_BA+0x23C	R/W	EPWM PDMA Control Register	0x0000_0000
EPWM_PDMACAP0_1 x=0,1	EPWMx_BA+0x240	R	EPWM Capture Channel 01 PDMA Register	0x0000_0000
EPWM_PDMACAP2_3 x=0,1	EPWMx_BA+0x244	R	EPWM Capture Channel 23 PDMA Register	0x0000_0000
EPWM_PDMACAP4_5 x=0,1	EPWMx_BA+0x248	R	EPWM Capture Channel 45 PDMA Register	0x0000_0000
EPWM_CAPIEN x=0,1	EPWMx_BA+0x250	R/W	EPWM Capture Interrupt Enable Register	0x0000_0000
EPWM_CAPIF	EPWMx_BA+0x254	R/W	EPWM Capture Interrupt Flag Register	0x0000_0000

x=0,1				
EPWM_CAPNF0 x=0,1	EPWMx_BA+0x258	R/W	EPWM Capture Input Noise Filter Register 0	0x0000_0000
EPWM_CAPNF1 x=0,1	EPWMx_BA+0x25C	R/W	EPWM Capture Input Noise Filter Register 1	0x0000_0000
EPWM_CAPNF2 x=0,1	EPWMx_BA+0x260	R/W	EPWM Capture Input Noise Filter Register 2	0x0000_0000
EPWM_CAPNF3 x=0,1	EPWMx_BA+0x264	R/W	EPWM Capture Input Noise Filter Register 3	0x0000_0000
EPWM_CAPNF4 x=0,1	EPWMx_BA+0x268	R/W	EPWM Capture Input Noise Filter Register 4	0x0000_0000
EPWM_CAPNF5 x=0,1	EPWMx_BA+0x26C	R/W	EPWM Capture Input Noise Filter Register 5	0x0000_0000
EPWM_EXTETCTL0 x=0,1	EPWMx_BA+0x270	R/W	EPWM External Event Trigger Control Register 0	0x0000_0000
EPWM_EXTETCTL1 x=0,1	EPWMx_BA+0x274	R/W	EPWM External Event Trigger Control Register 1	0x0000_0000
EPWM_EXTETCTL2 x=0,1	EPWMx_BA+0x278	R/W	EPWM External Event Trigger Control Register 2	0x0000_0000
EPWM_EXTETCTL3 x=0,1	EPWMx_BA+0x27C	R/W	EPWM External Event Trigger Control Register 3	0x0000_0000
EPWM_EXTETCTL4 x=0,1	EPWMx_BA+0x280	R/W	EPWM External Event Trigger Control Register 4	0x0000_0000
EPWM_EXTETCTL5 x=0,1	EPWMx_BA+0x284	R/W	EPWM External Event Trigger Control Register 5	0x0000_0000
EPWM_SWEOFCTL x=0,1	EPWMx_BA+0x288	R/W	EPWM Software Event Output Force Control Register	0x0000_0000
EPWM_SWEOFTRG x=0,1	EPWMx_BA+0x28C	R/W	EPWM Software Event Output Force Trigger Register	0x0000_0000
EPWM_CLKPSC0 x=0,1	EPWMx_BA+0x290	R/W	EPWM Clock Prescale Register 0	0x0000_0000
EPWM_CLKPSC1 x=0,1	EPWMx_BA+0x294	R/W	EPWM Clock Prescale Register 1	0x0000_0000
EPWM_CLKPSC2 x=0,1	EPWMx_BA+0x298	R/W	EPWM Clock Prescale Register 2	0x0000_0000
EPWM_CLKPSC3 x=0,1	EPWMx_BA+0x29C	R/W	EPWM Clock Prescale Register 3	0x0000_0000
EPWM_CLKPSC4 x=0,1	EPWMx_BA+0x2A0	R/W	EPWM Clock Prescale Register 4	0x0000_0000
EPWM_CLKPSC5 x=0,1	EPWMx_BA+0x2A4	R/W	EPWM Clock Prescale Register 5	0x0000_0000

EPWM_RDTCNT0_1 x=0,1	EPWMx_BA+0x2A8	R/W	EPWM Rising Dead-time Counter Register 0/1	0x0000_0000
EPWM_RDTCNT2_3 x=0,1	EPWMx_BA+0x2AC	R/W	EPWM Rising Dead-time Counter Register 2/3	0x0000_0000
EPWM_RDTCNT4_5 x=0,1	EPWMx_BA+0x2B0	R/W	EPWM Rising Dead-time Counter Register 4/5	0x0000_0000
EPWM_FDTCNT0_1 x=0,1	EPWMx_BA+0x2B4	R/W	EPWM Falling Dead-time Counter Register 0/1	0x0000_0000
EPWM_FDTCNT2_3 x=0,1	EPWMx_BA+0x2B8	R/W	EPWM Falling Dead-time Counter Register 2/3	0x0000_0000
EPWM_FDTCNT4_5 x=0,1	EPWMx_BA+0x2BC	R/W	EPWM Falling Dead-time Counter Register 4/5	0x0000_0000
EPWM_DTCTL x=0,1	EPWMx_BA+0x2C0	R/W	EPWM Dead-time Control Register	0x0000_0000
EPWM_PBUF0 x=0,1	EPWMx_BA+0x304	R	EPWM PERIOD0 Buffer	0x0000_0000
EPWM_PBUF1 x=0,1	EPWMx_BA+0x308	R	EPWM PERIOD1 Buffer	0x0000_0000
EPWM_PBUF2 x=0,1	EPWMx_BA+0x30C	R	EPWM PERIOD2 Buffer	0x0000_0000
EPWM_PBUF3 x=0,1	EPWMx_BA+0x310	R	EPWM PERIOD3 Buffer	0x0000_0000
EPWM_PBUF4 x=0,1	EPWMx_BA+0x314	R	EPWM PERIOD4 Buffer	0x0000_0000
EPWM_PBUF5 x=0,1	EPWMx_BA+0x318	R	EPWM PERIOD5 Buffer	0x0000_0000
EPWM_CMPBUF0 x=0,1	EPWMx_BA+0x31C	R	EPWM CMPDAT0 Buffer	0x0000_0000
EPWM_CMPBUF1 x=0,1	EPWMx_BA+0x320	R	EPWM CMPDAT1 Buffer	0x0000_0000
EPWM_CMPBUF2 x=0,1	EPWMx_BA+0x324	R	EPWM CMPDAT2 Buffer	0x0000_0000
EPWM_CMPBUF3 x=0,1	EPWMx_BA+0x328	R	EPWM CMPDAT3 Buffer	0x0000_0000
EPWM_CMPBUF4 x=0,1	EPWMx_BA+0x32C	R	EPWM CMPDAT4 Buffer	0x0000_0000
EPWM_CMPBUF5 x=0,1	EPWMx_BA+0x330	R	EPWM CMPDAT5 Buffer	0x0000_0000
EPWM_FTCBUF0_1 x=0,1	EPWMx_BA+0x340	R	EPWM FTCMPDAT0_1 Buffer	0x0000_0000
EPWM_FTCBUF2_3	EPWMx_BA+0x344	R	EPWM FTCMPDAT2_3 Buffer	0x0000_0000

x=0,1				
EPWM_FTCBUF4_5 **x=0,1**	EPWMx_BA+0x348	R	EPWM FTCMPDAT4_5 Buffer	0x0000_0000
EPWM_FTCI **x=0,1**	EPWMx_BA+0x34C	R/W	EPWM FTCMPDAT Indicator Register	0x0000_0000
EPWM_CPSCBUF0 **x=0,1**	EPWMx_BA+0x350	R	EPWM CLKPSC0 Buffer	0x0000_0000
EPWM_CPSCBUF1 **x=0,1**	EPWMx_BA+0x354	R	EPWM CLKPSC1 Buffer	0x0000_0000
EPWM_CPSCBUF2 **x=0,1**	EPWMx_BA+0x358	R	EPWM CLKPSC2 Buffer	0x0000_0000
EPWM_CPSCBUF3 **x=0,1**	EPWMx_BA+0x35C	R	EPWM CLKPSC3 Buffer	0x0000_0000
EPWM_CPSCBUF4 **x=0,1**	EPWMx_BA+0x360	R	EPWM CLKPSC4 Buffer	0x0000_0000
EPWM_CPSCBUF5 **x=0,1**	EPWMx_BA+0x364	R	EPWM CLKPSC5 Buffer	0x0000_0000
EPWM_IFACNT0 **x=0,1**	EPWMx_BA+0x368	R	EPWM Interrupt Flag Accumulator Counter 0	0x0000_0000
EPWM_IFACNT1 **x=0,1**	EPWMx_BA+0x36C	R	EPWM Interrupt Flag Accumulator Counter 1	0x0000_0000
EPWM_IFACNT2 **x=0,1**	EPWMx_BA+0x370	R	EPWM Interrupt Flag Accumulator Counter 2	0x0000_0000
EPWM_IFACNT3 **x=0,1**	EPWMx_BA+0x374	R	EPWM Interrupt Flag Accumulator Counter 3	0x0000_0000
EPWM_IFACNT4 **x=0,1**	EPWMx_BA+0x378	R	EPWM Interrupt Flag Accumulator Counter 4	0x0000_0000
EPWM_IFACNT5 **x=0,1**	EPWMx_BA+0x37C	R	EPWM Interrupt Flag Accumulator Counter 5	0x0000_0000

6.12.7 Register Description

EPWM Control Register 0 (EPWM_CTL0)

Register	Offset	R/W	Description	Reset Value
EPWM_CTL0	EPWMx_BA+0x00	R/W	EPWM Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
DBGTRIOFF	**DBGHALT**	**Reserved**					**GROUPEN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**IMMLDEN5**	**IMMLDEN4**	**IMMLDEN3**	**IMMLDEN2**	**IMMLDEN1**	**IMMLDEN0**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**WINLDEN5**	**WINLDEN4**	**WINLDEN3**	**WINLDEN2**	**WINLDEN1**	**WINLDEN0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CTRLD5**	**CTRLD4**	**CTRLD3**	**CTRLD2**	**CTRLD1**	**CTRLD0**

Bits	Description	
[31]	**DBGTRIOFF**	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement effects EPWM output. EPWM pin will be forced as tri-state while ICE debug mode acknowledged. 1 = ICE debug mode acknowledgement disabled. EPWM pin will keep output no matter ICE debug mode acknowledged or not. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[30]	**DBGHALT**	**ICE Debug Mode Counter Halt (Write Protect)** If counter halt is enabled, EPWM all counters will keep current value until exit ICE debug mode. 0 = ICE debug mode counter halt Disabled. 1 = ICE debug mode counter halt Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[29:26]	**Reserved**	Reserved.
[24]	**GROUPEN**	**Group Function Enable Bit** 0 = The output waveform of each EPWM channel are independent. 1 = Unify the EPWM_CH2 and EPWM_CH4 to output the same waveform as EPWM_CH0 and unify the EPWM_CH3 and EPWM_CH5 to output the same waveform as EPWM_CH1.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**IMMLDENn**	**Immediately Load Enable Bits** 0 = PERIOD will load to PBUF at the end point of each period. CMP will load to CMPBUF at the end point or center point of each period by setting CTRLD bit. 1 = PERIOD/CMP will load to PBUF and CMPBUF immediately when software update PERIOD/CMP. **Note:** If IMMLDENn is enabled, WINLDENn and CTRLDn will be invalid.
[15:14]	**Reserved**	Reserved.

[8+n] n=0,1..5	**WINLDENn**	**Window Load Enable Bits** 0 = PERIOD will load to PBUF at the end point of each period. CMP will load to CMPBUF at the end point or center point of each period by setting CTRLD bit. 1 = PERIOD will load to PBUF at the end point of each period. CMP will load to CMPBUF at the end point of each period when valid reload window is set. The valid reload window is set by software write 1 to EPWM_LOAD register and cleared by hardware after load success.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CTRLDn**	**Center Re-load** In up-down counter type, PERIOD will load to PBUF at the end point of each period. CMP will load to CMPBUF at the center point of a period.

EPWM Control Register 1 (EPWM_CTL1)

Register	Offset	R/W	Description	Reset Value
EPWM_CTL1	EPWMx_BA+0x04	R/W	EPWM Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					OUTMODE4	OUTMODE2	OUTMODE0
23	22	21	20	19	18	17	16
Reserved		CNTMODE5	CNTMODE4	CNTMODE3	CNTMODE2	CNTMODE1	CNTMODE0
15	14	13	12	11	10	9	8
Reserved				CNTTYPE5		CNTTYPE4	
7	6	5	4	3	2	1	0
CNTTYPE3		CNTTYPE2		CNTTYPE1		CNTTYPE0	

Bits	Description	
[31:27]	**Reserved**	Reserved.
[24+n/2] n=0,2,4	**OUTMODEn**	**EPWM Output Mode** Each bit n controls the output mode of corresponding EPWM channel n. 0 = EPWM independent mode. 1 = EPWM complementary mode. **Note:** When operating in group function, these bits must all set to the same mode.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CNTMODEn**	**EPWM Counter Mode** 0 = Auto-reload mode. 1 = One-shot mode.
[15:12]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..5	**CNTTYPEn**	**EPWM Counter Behavior Type** 00 = Up counter type (supported in capture mode). 01 = Down count type (supported in capture mode). 10 = Up-down counter type. 11 = Reserved.

EPWM Synchronization Register (EPWM_SYNC)

Register	Offset	R/W	Description	Reset Value
EPWM_SYNC	EPWMx_BA+0x08	R/W	EPWM Synchronization Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					PHSDIR4	PHSDIR2	PHSDIR0
23	22	21	20	19	18	17	16
SINPINV	SFLTCNT			SFLTCSEL			SNFLTEN
15	14	13	12	11	10	9	8
Reserved		SINSRC4		SINSRC2		SINSRC0	
7	6	5	4	3	2	1	0
Reserved					PHSEN4	PHSEN2	PHSEN0

Bits	Description	
[31:27]	**Reserved**	Reserved.
[24+n/2] n=0,2,4	**PHSDIRn**	**EPWM Phase Direction Control** 0 = Control EPWM counter count decrement after synchronizing. 1 = Control EPWM counter count increment after synchronizing.
[23]	**SINPINV**	**SYNC Input Pin Inverse** 0 = The state of pin SYNC is passed to the negative edge detector. 1 = The inversed state of pin SYNC is passed to the negative edge detector.
[22:20]	**SFLTCNT**	**SYNC Edge Detector Filter Count** The register bits control the counter number of edge detector.
[19:17]	**SFLTCSEL**	**SYNC Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK. 001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4. 011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[16]	**SNFLTEN**	**EPWM0_SYNC_IN Noise Filter Enable Bits** 0 = Noise filter of input pin EPWM0_SYNC_IN Disabled. 1 = Noise filter of input pin EPWM0_SYNC_IN Enabled.
[15:14]	**Reserved**	Reserved.
[9+n:8+n] n=0,2,4	**SINSRCn**	**EPWM0_SYNC_IN Source Selection** 00 = Synchronize source from SYNC_IN or SWSYNC. 01 = Counter equal to 0. 10 = Counter equal to EPWM_CMPDATm, m denotes 1, 3, 5.

		11 = SYNC_OUT will not be generated.
[7:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**PHSENn**	**SYNC Phase Enable Bits** 0 = EPWM counter disable to load PHS value. 1 = EPWM counter enable to load PHS value.

EPWM Software Control Synchronization Register (EPWM_SWSYNC)

Register	Offset	R/W	Description	Reset Value
EPWM_SWSYNC	EPWMx_BA+0x0C	W	EPWM Software Control Synchronization Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					SWSYNC4	SWSYNC2	SWSYNC0

Bits	Description	
[31:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**SWSYNCn**	**Software SYNC Function (Write Only)** When SINSRCn (EPWM_SYNC[13:8]) is selected to 0, SYNC_OUT source comes from SYNC_IN or this bit.

EPWM Clock Source Register (EPWM_CLKSRC)

Register	Offset	R/W	Description	Reset Value
EPWM_CLKSRC	EPWMx_BA+0x10	R/W	EPWM Clock Source Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**ECLKSRC4**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**ECLKSRC2**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**ECLKSRC0**		

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:16]	**ECLKSRC4**	**EPWM_CH45 External Clock Source Select** 000 = EPWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.
[15:11]	**Reserved**	Reserved.
[10:8]	**ECLKSRC2**	**EPWM_CH23 External Clock Source Select** 000 = EPWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.
[7:3]	**Reserved**	Reserved.
[2:0]	**ECLKSRC0**	**EPWM_CH01 External Clock Source Select** 000 = EPWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.

EPWM Counter Enable Register (EPWM_CNTEN)

Register	Offset	R/W	Description	Reset Value
EPWM_CNTEN	EPWMx_BA+0x20	R/W	EPWM Counter Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CNTEN5**	**CNTEN4**	**CNTEN3**	**CNTEN2**	**CNTEN1**	**CNTEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CNTENn**	**EPWM Counter Enable Bits** 0 = EPWM Counter and clock prescaler stop running. 1 = EPWM Counter and clock prescaler start running.

EPWM Clear Counter Register (EPWM_CNTCLR)

Register	Offset	R/W	Description	Reset Value
EPWM_CNTCLR	EPWMx_BA+0x24	R/W	EPWM Clear Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CNTCLR5**	**CNTCLR4**	**CNTCLR3**	**CNTCLR2**	**CNTCLR1**	**CNTCLR0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CNTCLRn**	**Clear EPWM Counter Control Bit** It is automatically cleared by hardware. Each bit n controls the corresponding EPWM channel n. 0 = No effect. 1 = Clear 16-bit EPWM counter to 0000H.

EPWM Load Register (EPWM_LOAD)

Register	Offset	R/W	Description	Reset Value
EPWM_LOAD	EPWMx_BA+0x28	R/W	EPWM Load Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**LOAD5**	**LOAD4**	**LOAD3**	**LOAD2**	**LOAD1**	**LOAD0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**LOADn**	**Re-load EPWM Comparator Register Control Bit** This bit is software write to reload EPWM_CMPDATn, n=0~5. Hardware clear when current EPWM period end. Write Operation: 0 = No effect. 1 = Set load window of window loading mode. Read Operation: 0 = No load window is set. 1 = Load window is set. **Note:** This bit only use in window loading mode, WINLDENn(EPWM_CTL0[13:8]) = 1.

EPWM Period Register 0~5 (EPWM_PERIOD0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_PERIOD0	EPWMx_BA+0x30	R/W	EPWM Period Register 0	0x0000_0000
EPWM_PERIOD1	EPWMx_BA+0x34	R/W	EPWM Period Register 1	0x0000_0000
EPWM_PERIOD2	EPWMx_BA+0x38	R/W	EPWM Period Register 2	0x0000_0000
EPWM_PERIOD3	EPWMx_BA+0x3C	R/W	EPWM Period Register 3	0x0000_0000
EPWM_PERIOD4	EPWMx_BA+0x40	R/W	EPWM Period Register 4	0x0000_0000
EPWM_PERIOD5	EPWMx_BA+0x44	R/W	EPWM Period Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PERIOD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PERIOD							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PERIOD**	**EPWM Period Register** Up-Count mode: In this mode, EPWM counter counts from 0 to PERIOD, and restarts from 0. EPWM period time = (PERIOD+1) * (CLKPSC+1) * EPWM_CLK . Down-Count mode: In this mode, EPWM counter counts from PERIOD to 0, and restarts from PERIOD. EPWM period time = (PERIOD+1) * (CLKPSC+1) * EPWM_CLK . Up-Down-Count mode: In this mode, EPWM counter counts from 0 to PERIOD, then decrements to 0 and repeats again. EPWM period time = 2 * PERIOD * (CLKPSC+1) * EPWM_CLK.

EPWM Comparator Register 0~5 (EPWM_CMPDAT0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CMPDAT0	EPWMx_BA+0x50	R/W	EPWM Comparator Register 0	0x0000_0000
EPWM_CMPDAT1	EPWMx_BA+0x54	R/W	EPWM Comparator Register 1	0x0000_0000
EPWM_CMPDAT2	EPWMx_BA+0x58	R/W	EPWM Comparator Register 2	0x0000_0000
EPWM_CMPDAT3	EPWMx_BA+0x5C	R/W	EPWM Comparator Register 3	0x0000_0000
EPWM_CMPDAT4	EPWMx_BA+0x60	R/W	EPWM Comparator Register 4	0x0000_0000
EPWM_CMPDAT5	EPWMx_BA+0x64	R/W	EPWM Comparator Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMP							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMP**	**EPWM Comparator Register** CMP is used to compare with CNT (EPWM_CNTn[15:0]) bits to generate EPWM waveform, interrupt and trigger EADC/DAC. In independent mode, EPWM_CMPDATn, n=0,1..5 denote as 6 independent EPWM_CH0~5 compared point. In complementary mode, EPWM_CMPDAT0, EPWM_CMPDAT 2, EPWM_CMPDAT4 denote as first compared point, and EPWM_CMPDAT1, EPWM_CMPDAT3, EPWM_CMPDAT5 denote as second compared point for the corresponding 3 complementary pairs EPWM_CH0 and EPWM_CH1, EPWM_CH2 and EPWM_CH3, EPWM_CH4 and EPWM_CH5.

EPWM Counter Phase Register 0_1, 2_3, 4_5 (EPWM_PHS0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_PHS0_1	EPWMx_BA+0x80	R/W	EPWM Counter Phase Register 0/1	0x0000_0000
EPWM_PHS2_3	EPWMx_BA+0x84	R/W	EPWM Counter Phase Register 2/3	0x0000_0000
EPWM_PHS4_5	EPWMx_BA+0x88	R/W	EPWM Counter Phase Register 4/5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PHS							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PHS							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PHS**	**EPWM Synchronous Start Phase Bits** PHS determines the EPWM synchronous start phase value. These bits only use in synchronous function.

EPWM Counter Register 0~5 (EPWM_CNT0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CNT0	EPWMx_BA+0x90	R	EPWM Counter Register 0	0x0000_0000
EPWM_CNT1	EPWMx_BA+0x94	R	EPWM Counter Register 1	0x0000_0000
EPWM_CNT2	EPWMx_BA+0x98	R	EPWM Counter Register 2	0x0000_0000
EPWM_CNT3	EPWMx_BA+0x9C	R	EPWM Counter Register 3	0x0000_0000
EPWM_CNT4	EPWMx_BA+0xA0	R	EPWM Counter Register 4	0x0000_0000
EPWM_CNT5	EPWMx_BA+0xA4	R	EPWM Counter Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**DIRF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT							

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DIRF**	**EPWM Direction Indicator Flag (Read Only)** 0 = Counter is counting down. 1 = Counter is counting up.
[15:0]	**CNT**	**EPWM Data Register (Read Only)** User can monitor CNT to know the current value in 16-bit period counter.

EPWM Generation Register 0 (EPWM_WGCTL0)

Register	Offset	R/W	Description	Reset Value
EPWM_WGCTL0	EPWMx_BA+0xB0	R/W	EPWM Generation Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**PRDPCTL5**		**PRDPCTL4**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRDPCTL3		**PRDPCTL2**		**PRDPCTL1**		**PRDPCTL0**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**ZPCTL5**		**ZPCTL4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ZPCTL3		**ZPCTL2**		**ZPCTL1**		**ZPCTL0**	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[17+2n:16+2n] n=0,1..5	**PRDPCTLn**	**EPWM Period or Center Point Control** EPWM can control output level when EPWM counter counts to (PERIODn+1). 00 = Do nothing. 01 = EPWM period (center) point output Low. 10 = EPWM period (center) point output High. 11 = EPWM period (center) point output Toggle. **Note:** This bit is center point control when EPWM counter operating in up-down counter type.
[15:12]	**Reserved**	Reserved.
[1+2n:2n] n=0,1..5	**ZPCTLn**	**EPWM Zero Point Control** EPWM can control output level when EPWM counter counts to 0. 00 = Do nothing. 01 = EPWM zero point output Low. 10 = EPWM zero point output High. 11 = EPWM zero point output Toggle.

EPWM Generation Register 1 (EPWM_WGCTL1)

Register	Offset	R/W	Description	Reset Value
EPWM_WGCTL1	EPWMx_BA+0xB4	R/W	EPWM Generation Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**CMPDCTL5**		**CMPDCTL4**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPDCTL3		**CMPDCTL2**		**CMPDCTL1**		**CMPDCTL0**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CMPUCTL5**		**CMPUCTL4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPUCTL3		**CMPUCTL2**		**CMPUCTL1**		**CMPUCTL0**	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[17+2n:16+2n] n=0,1..5	**CMPDCTLn**	**EPWM Compare Down Point Control** EPWM can control output level when EPWM counter counts down to CMP. 00 = Do nothing. 01 = EPWM compare down point output Low. 10 = EPWM compare down point output High. 11 = EPWM compare down point output Toggle. **Note:** In complementary mode, CMPDCTL1, 3, 5 is used as another CMPDCTL for channel 0, 2, 4.
[15:12]	**Reserved**	Reserved.
[1+2n:2n] n=0,1..5	**CMPUCTLn**	**EPWM Compare Up Point Control** EPWM can control output level when EPWM counter counts up to CMP. 00 = Do nothing. 01 = EPWM compare up point output Low. 10 = EPWM compare up point output High. 11 = EPWM compare up point output Toggle. **Note:** In complementary mode, CMPUCTL1, 3, 5 is used as another CMPUCTL for channel 0, 2, 4.

EPWM Mask Enable Register (EPWM_MSKEN)

Register	Offset	R/W	Description	Reset Value
EPWM_MSKEN	EPWMx_BA+0xB8	R/W	EPWM Mask Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		MSKEN5	MSKEN4	MSKEN3	MSKEN2	MSKEN1	MSKEN0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**MSKENn**	**EPWM Mask Enable Bits** The EPWM output signal will be masked when this bit is enabled. The corresponding EPWM channel n will output MSKDATn (EPWM_MSK[5:0]) data. 0 = EPWM output signal is non-masked. 1 = EPWM output signal is masked and output MSKDATn data.

EPWM Mask DATA Register (EPWM_MSK)

Register	Offset	R/W	Description	Reset Value
EPWM_MSK	EPWMx_BA+0xBC	R/W	EPWM Mask Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		MSKDAT5	MSKDAT4	MSKDAT3	MSKDAT2	MSKDAT1	MSKDAT0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**MSKDATn**	**EPWM Mask Data Bit** This data bit control the state of EPWMn output pin, if corresponding mask function is enabled. 0 = Output logic low to EPWM channel n. 1 = Output logic high to EPWM channel n.

EPWM Brake Noise Filter Register (EPWM_BNF)

Register	Offset	R/W	Description	Reset Value
EPWM_BNF	EPWMx_BA+0xC0	R/W	EPWM Brake Noise Filter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							BK1SRC
23	22	21	20	19	18	17	16
Reserved							BK0SRC
15	14	13	12	11	10	9	8
BRK1PINV	BRK1FCNT			BRK1NFSEL			BRK1NFEN
7	6	5	4	3	2	1	0
BRK0PINV	BRK0FCNT			BRK0NFSEL			BRK0NFEN

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**BK1SRC**	**Brake 1 Pin Source Select** For EPWM0 setting: 0 = Brake 1 pin source come from EPWM0_BRAKE1. 1 = Brake 1 pin source come from EPWM1_BRAKE1. For EPWM1 setting: 0 = Brake 1 pin source come from EPWM1_BRAKE1. 1 = Brake 1 pin source come from EPWM0_BRAKE1.
[23:17]	**Reserved**	Reserved.
[16]	**BK0SRC**	**Brake 0 Pin Source Select** For EPWM0 setting: 0 = Brake 0 pin source come from EPWM0_BRAKE0. 1 = Brake 0 pin source come from EPWM1_BRAKE0. For EPWM1 setting: 0 = Brake 0 pin source come from EPWM1_BRAKE0. 1 = Brake 0 pin source come from EPWM0_BRAKE0.
[15]	**BRK1PINV**	**Brake 1 Pin Inverse** 0 **=** brake pin event will be detected if EPWMx_BRAKE1 pin status transfer from low to high in edge-detect, or pin status is high in level-detect. 1 = brake pin event will be detected if EPWMx_BRAKE1 pin status transfer from high to low in edge-detect, or pin status is low in level-detect.
[14:12]	**BRK1FCNT**	**Brake 1 Edge Detector Filter Count** The register bits control the Brake1 filter counter to count from 0 to BRK1FCNT.
[11:9]	**BRK1NFSEL**	**Brake 1 Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK. 001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4.

		011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[8]	**BRK1NFEN**	**EPWM Brake 1 Noise Filter Enable Bit** 0 = Noise filter of EPWM Brake 1 Disabled. 1 = Noise filter of EPWM Brake 1 Enabled.
[7]	**BRK0PINV**	**Brake 0 Pin Inverse** 0 **=** brake pin event will be detected if EPWMx_BRAKE0 pin status transfer from low to high in edge-detect, or pin status is high in level-detect. 1 = brake pin event will be detected if EPWMx_BRAKE0 pin status transfer from high to low in edge-detect, or pin status is low in level-detect.
[6:4]	**BRK0FCNT**	**Brake 0 Edge Detector Filter Count** The register bits control the Brake0 filter counter to count from 0 to BRK0FCNT.
[3:1]	**BRK0NFSEL**	**Brake 0 Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK. 001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4. 011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[0]	**BRK0NFEN**	**EPWM Brake 0 Noise Filter Enable Bit** 0 = Noise filter of EPWM Brake 0 Disabled. 1 = Noise filter of EPWM Brake 0 Enabled.

EPWM System Fail Brake Control Register (EPWM_FAILBRK)

Register	Offset	R/W	Description	Reset Value
EPWM_FAILBRK	EPWMx_BA+0xC4	R/W	EPWM System Fail Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**CORBRKEN**	**RAMBRKEN**	**BODBRKEN**	**CSSBRKEN**

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CORBRKEN**	**Core Lockup Detection Trigger EPWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by Core lockup detection Disabled. 1 = Brake Function triggered by Core lockup detection Enabled.
[2]	**RAMBRKEN**	**SRAM Parity Error Detection Trigger EPWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by SRAM parity error detection Disabled. 1 = Brake Function triggered by SRAM parity error detection Enabled.
[1]	**BODBRKEN**	**Brown-out Detection Trigger EPWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by BOD Disabled. 1 = Brake Function triggered by BOD Enabled.
[0]	**CSSBRKEN**	**Clock Security System Detection Trigger EPWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by CSS detection Disabled. 1 = Brake Function triggered by CSS detection Enabled.

EPWM Brake Edge Detect Control Register 0_1, 2_3, 4_5 (EPWM_BRKCTL0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_BRKCTL0_1	EPWMx_BA+0xC8	R/W	EPWM Brake Edge Detect Control Register 0/1	0x0000_0000
EPWM_BRKCTL2_3	EPWMx_BA+0xCC	R/W	EPWM Brake Edge Detect Control Register 2/3	0x0000_0000
EPWM_BRKCTL4_5	EPWMx_BA+0xD0	R/W	EPWM Brake Edge Detect Control Register 4/5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	EADC2LBEN	EADC1LBEN	EADC0LBEN	Reserved			
23	22	21	20	19	18	17	16
Reserved	EADC2EBEN	EADC1EBEN	EADC0EBEN	BRKAODD		BRKAEVEN	
15	14	13	12	11	10	9	8
SYSLBEN	Reserved	BRKP1LEN	BRKP0LEN	CPO3LBEN	CPO2LBEN	CPO1LBEN	CPO0LBEN
7	6	5	4	3	2	1	0
SYSEBEN	Reserved	BRKP1EEN	BRKP0EEN	CPO3EBEN	CPO2EBEN	CPO1EBEN	CPO0EBEN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**EADC2LBEN**	**Enable EADC2 Result Monitor as Level-detect Brake Source (Write Protect)** 0 = EADC2RM as level-detect brake source Disabled. 1 = EADC2RM as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[29]	**EADC1LBEN**	**Enable EADC1 Result Monitor as Level-detect Brake Source (Write Protect)** 0 = EADC1RM as level-detect brake source Disabled. 1 = EADC1RM as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[28]	**EADC0LBEN**	**Enable EADC0 Result Monitor as Level-detect Brake Source (Write Protect)** 0 = EADC0RM as level-detect brake source Disabled. 1 = EADC0RM as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[27:23]	**Reserved**	Reserved.
[22]	**EADC2EBEN**	**Enable EADC2 Result Monitor as Edge-detect Brake Source (Write Protect)** 0 = EADC1RM as edge-detect brake source Disabled. 1 = EADC1RM as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[21]	**EADC1EBEN**	**Enable EADC1 Result Monitor as Edge-detect Brake Source (Write Protect)** 0 = EADC1RM as edge-detect brake source Disabled. 1 = EADC1RM as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[20]	**EADC0EBEN**	**Enable EADC0 Result Monitor as Edge-detect Brake Source (Write Protect)**

		0 = EADC0RM as edge-detect brake source Disabled. 1 = EADC0RM as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[19:18]	**BRKAODD**	**EPWM Brake Action Select for Odd Channel (Write Protect)** 00 = EPWMx brake event will not affect odd channels output. 01 = EPWM odd channel output tri-state when EPWMx brake event happened. 10 = EPWM odd channel output low level when EPWMx brake event happened. 11 = EPWM odd channel output high level when EPWMx brake event happened. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[17:16]	**BRKAEVEN**	**EPWM Brake Action Select for Even Channel (Write Protect)** 00 = EPWMx brake event will not affect even channels output. 01 = EPWM even channel output tri-state when EPWMx brake event happened. 10 = EPWM even channel output low level when EPWMx brake event happened. 11 = EPWM even channel output high level when EPWMx brake event happened. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[15]	**SYSLBEN**	**Enable System Fail As Level-detect Brake Source (Write Protect)** 0 = System Fail condition as level-detect brake source Disabled. 1 = System Fail condition as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[14]	**Reserved**	Reserved.
[13]	**BRKP1LEN**	**Enable BKP1 Pin As Level-detect Brake Source (Write Protect)** 0 = EPWMx_BRAKE1 pin as level-detect brake source Disabled. 1 = EPWMx_BRAKE1 pin as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[12]	**BRKP0LEN**	**Enable BKP0 Pin As Level-detect Brake Source (Write Protect)** 0 = EPWMx_BRAKE0 pin as level-detect brake source Disabled. 1 = EPWMx_BRAKE0 pin as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[11]	**CPO3LBEN**	**Enable ACMP3_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = ACMP3_O as level-detect brake source Disabled. 1 = ACMP3_O as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[10]	**CPO2LBEN**	**Enable ACMP2_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = ACMP2_O as level-detect brake source Disabled. 1 = ACMP2_O as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[9]	**CPO1LBEN**	**Enable ACMP1_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = ACMP1_O as level-detect brake source Disabled. 1 = ACMP1_O as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[8]	**CPO0LBEN**	**Enable ACMP0_O Digital Output As Level-detect Brake Source (Write Protect)** 0 = ACMP0_O as level-detect brake source Disabled. 1 = ACMP0_O as level-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7]	**SYSEBEN**	**Enable System Fail As Edge-detect Brake Source (Write Protect)**

		0 = System Fail condition as edge-detect brake source Disabled. 1 = System Fail condition as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[6]	**Reserved**	Reserved.
[5]	**BRKP1EEN**	**Enable EPWMx_BRAKE1 Pin As Edge-detect Brake Source (Write Protect)** 0 = EPWMx_BRAKE1 pin as edge-detect brake source Disabled. 1 = EPWMx_BRAKE1 pin as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[4]	**BRKP0EEN**	**Enable EPWMx_BRAKE0 Pin As Edge-detect Brake Source (Write Protect)** 0 = EPWMx_BRAKE0 pin as edge-detect brake source Disabled. 1 = EPWMx_BRAKE0 pin as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[3]	**CPO3EBEN**	**Enable ACMP3_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = ACMP3_O as edge-detect brake source Disabled. 1 = ACMP3_O as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[2]	**CPO2EBEN**	**Enable ACMP2_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = ACMP2_O as edge-detect brake source Disabled. 1 = ACMP2_O as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[1]	**CPO1EBEN**	**Enable ACMP1_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = ACMP1_O as edge-detect brake source Disabled. 1 = ACMP1_O as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[0]	**CPO0EBEN**	**Enable ACMP0_O Digital Output As Edge-detect Brake Source (Write Protect)** 0 = ACMP0_O as edge-detect brake source Disabled. 1 = ACMP0_O as edge-detect brake source Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Pin Polar Inverse Control (EPWM_POLCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_POLCTL	EPWMx_BA+0xD4	R/W	EPWM Pin Polar Inverse Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		PINV5	PINV4	PINV3	PINV2	PINV1	PINV0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**PINVn**	**EPWM PIN Polar Inverse Control** The register controls polarity state of EPWMx_CHn output pin. 0 = EPWMx_CHn output pin polar inverse Disabled. 1 = EPWMx_CHn output pin polar inverse Enabled.

EPWM Output Enable Register (EPWM_POEN)

Register	Offset	R/W	Description	Reset Value
EPWM_POEN	EPWMx_BA+0xD8	R/W	EPWM Output Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**POEN5**	**POEN4**	**POEN3**	**POEN2**	**POEN1**	**POEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**POENn**	**EPWM Pin Output Enable Bits** 0 = EPWMx_CHn pin at tri-state. 1 = EPWMx_CHn pin in output mode.

EPWM Software Brake Control Register (EPWM_SWBRK)

Register	Offset	R/W	Description	Reset Value
EPWM_SWBRK	EPWMx_BA+0xDC	W	EPWM Software Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**BRKLTRG4**	**BRKLTRG2**	**BRKLTRG0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**BRKETRG4**	**BRKETRG2**	**BRKETRG0**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[8+n/2] n=0,2,4	**BRKLTRGn**	**EPWM Level Brake Software Trigger (Write Only) (Write Protect)** Write 1 to this bit will trigger level brake, and set BRKLIFn to 1 in EPWM_INTSTS1 register. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**BRKETRGn**	**EPWM Edge Brake Software Trigger (Write Only) (Write Protect)** Write 1 to this bit will trigger edge brake, and set BRKEIFn to 1 in EPWM_INTSTS1 register. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Interrupt Enable Register 0 (EPWM_INTEN0)

Register	Offset	R/W	Description	Reset Value
EPWM_INTEN0	EPWMx_BA+0xE0	R/W	EPWM Interrupt Enable Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIEN5	CMPDIEN4	CMPDIEN3	CMPDIEN2	CMPDIEN1	CMPDIEN0
23	22	21	20	19	18	17	16
Reserved		CMPUIEN5	CMPUIEN4	CMPUIEN3	CMPUIEN2	CMPUIEN1	CMPUIEN0
15	14	13	12	11	10	9	8
Reserved		PIEN5	PIEN4	PIEN3	PIEN2	PIEN1	PIEN0
7	6	5	4	3	2	1	0
Reserved		ZIEN5	ZIEN4	ZIEN3	ZIEN2	ZIEN1	ZIEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CMPDIENn**	**EPWM Compare Down Count Interrupt Enable Bits** 0 = Compare down count interrupt Disabled. 1 = Compare down count interrupt Enabled. **Note:** In complementary mode, CMPDIEN1, 3, 5 is used as another CMPDIEN for channel 0, 2, 4.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CMPUIENn**	**EPWM Compare Up Count Interrupt Enable Bits** 0 = Compare up count interrupt Disabled. 1 = Compare up count interrupt Enabled. **Note:** In complementary mode, CMPUIEN1, 3, 5 is used as another CMPUIEN for channel 0, 2, 4.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**PIENn**	**EPWM Period Point Interrupt Enable Bits** 0 = Period point interrupt Disabled. 1 = Period point interrupt Enabled. **Note 1:** When up-down counter type period point means center point. **Note 2:** Odd channels will read always 0 at complementary mode.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**ZIENn**	**EPWM Zero Point Interrupt Enable Bits** 0 = Zero point interrupt Disabled. 1 = Zero point interrupt Enabled. **Note:** Odd channels will read always 0 at complementary mode.

EPWM Interrupt Enable Register 1 (EPWM_INTEN1)

Register	Offset	R/W	Description	Reset Value
EPWM_INTEN1	EPWMx_BA+0xE4	R/W	EPWM Interrupt Enable Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**BRKLIEN4_5**	**BRKLIEN2_3**	**BRKLIEN0_1**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**BRKEIEN4_5**	**BRKEIEN2_3**	**BRKEIEN0_1**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**BRKLIEN4_5**	**EPWM Level-detect Brake Interrupt Enable for Channel4/5 (Write Protect)** 0 = Level-detect Brake interrupt for channel4/5 Disabled. 1 = Level-detect Brake interrupt for channel4/5 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[9]	**BRKLIEN2_3**	**EPWM Level-detect Brake Interrupt Enable for Channel2/3 (Write Protect)** 0 = Level-detect Brake interrupt for channel2/3 Disabled. 1 = Level-detect Brake interrupt for channel2/3 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[8]	**BRKLIEN0_1**	**EPWM Level-detect Brake Interrupt Enable for Channel0/1 (Write Protect)** 0 = Level-detect Brake interrupt for channel0/1 Disabled. 1 = Level-detect Brake interrupt for channel0/1 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:3]	**Reserved**	Reserved.
[2]	**BRKEIEN4_5**	**EPWM Edge-detect Brake Interrupt Enable for Channel4/5 (Write Protect)** 0 = Edge-detect Brake interrupt for channel4/5 Disabled. 1 = Edge-detect Brake interrupt for channel4/5 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[1]	**BRKEIEN2_3**	**EPWM Edge-detect Brake Interrupt Enable for Channel2/3 (Write Protect)** 0 = Edge-detect Brake interrupt for channel2/3 Disabled. 1 = Edge-detect Brake interrupt for channel2/3 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[0]	**BRKEIEN0_1**	**EPWM Edge-detect Brake Interrupt Enable for Channel0/1 (Write Protect)** 0 = Edge-detect Brake interrupt for channel0/1 Disabled. 1 = Edge-detect Brake interrupt for channel0/1 Enabled. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Interrupt Flag Register 0 (EPWM_INTSTS0)

Register	Offset	R/W	Description	Reset Value
EPWM_INTSTS0	EPWMx_BA+0xE8	R/W	EPWM Interrupt Flag Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIF5	CMPDIF4	CMPDIF3	CMPDIF2	CMPDIF1	CMPDIF0
23	22	21	20	19	18	17	16
Reserved		CMPUIF5	CMPUIF4	CMPUIF3	CMPUIF2	CMPUIF1	CMPUIF0
15	14	13	12	11	10	9	8
Reserved		PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
7	6	5	4	3	2	1	0
Reserved		ZIF5	ZIF4	ZIF3	ZIF2	ZIF1	ZIF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CMPDIFn**	**EPWM Compare Down Count Interrupt Flag** Flag is set by hardware when EPWM counter down count and reaches EPWM_CMPDATn, software can clear this bit by writing 1 to it. **Note:** In complementary mode, CMPDIF1, 3, 5 is used as another CMPDIF for channel 0, 2, 4.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CMPUIFn**	**EPWM Compare Up Count Interrupt Flag** Flag is set by hardware when EPWM counter up count and reaches EPWM_CMPDATn, software can clear this bit by writing 1 to it. **Note:** In complementary mode, CMPUIF1, 3, 5 is used as another CMPUIF for channel 0, 2, 4.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**PIFn**	**EPWM Period Point Interrupt Flag** This bit is set by hardware when EPWM counter reaches EPWM_PERIODn. **Note:** This bit can be cleared to 0 by software writing 1.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**ZIFn**	**EPWM Zero Point Interrupt Flag** This bit is set by hardware when EPWM counter reaches 0. **Note:** This bit can be cleared to 0 by software writing 1

EPWM Interrupt Flag Register 1 (EPWM_INTSTS1)

Register	Offset	R/W	Description	Reset Value
EPWM_INTSTS1	EPWMx_BA+0xEC	R/W	EPWM Interrupt Flag Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		BRKLSTS5	BRKLSTS4	BRKLSTS3	BRKLSTS2	BRKLSTS1	BRKLSTS0
23	22	21	20	19	18	17	16
Reserved		BRKESTS5	BRKESTS4	BRKESTS3	BRKESTS2	BRKESTS1	BRKESTS0
15	14	13	12	11	10	9	8
Reserved		BRKLIF5	BRKLIF4	BRKLIF3	BRKLIF2	BRKLIF1	BRKLIF0
7	6	5	4	3	2	1	0
Reserved		BRKEIF5	BRKEIF4	BRKEIF3	BRKEIF2	BRKEIF1	BRKEIF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**BRKLSTSn**	**EPWM Channel N Level-detect Brake Status (Read Only)** 0 = EPWM channel n level-detect brake state is released. 1 = When EPWM channel n level-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the EPWM channel n at brake state. **Note:** This bit is read only and auto cleared by hardware. When enabled brake source return to high level, EPWM will release brake state until current EPWM period finished. The EPWM waveform will start output from next full EPWM period.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**BRKESTSn**	**EPWM Channel N Edge-detect Brake Status (Read Only)** 0 = EPWM channel n edge-detect brake state is released. 1 = When EPWM channel n edge-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the EPWM channel n at brake state. **Note:** This bit is read only and auto cleared by hardware. When edge-detect brake interrupt flag is cleared, EPWM will release brake state until current EPWM period finished. The EPWM waveform will start output from next full EPWM period.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**BRKLIFn**	**EPWM Channel N Level-detect Brake Interrupt Flag (Write Protect)** 0 = EPWM channel n level-detect brake event do not happened. 1 = When EPWM channel n level-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**BRKEIFn**	**EPWM Channel N Edge-detect Brake Interrupt Flag (Write Protect)** 0 = EPWM channel n edge-detect brake event do not happened. 1 = When EPWM channel n edge-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Trigger DAC Enable Register (EPWM_DACTRGEN)

Register	Offset	R/W	Description	Reset Value
EPWM_DACTRGEN	EPWMx_BA+0xF4	R/W	EPWM Trigger DAC Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CDTRGEN5	CDTRGEN4	CDTRGEN3	CDTRGEN2	CDTRGEN1	CDTRGEN0
23	22	21	20	19	18	17	16
Reserved		CUTRGEN5	CUTRGEN4	CUTRGEN3	CUTRGEN2	CUTRGEN1	CUTRGEN0
15	14	13	12	11	10	9	8
Reserved		PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
7	6	5	4	3	2	1	0
Reserved		ZTE5	ZTE4	ZTE3	ZTE2	ZTE1	ZTE0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**CDTRGENn**	**EPWM Compare Down Count Point Trigger DAC Enable Bits** EPWM can trigger DAC to start action when EPWM counter down count to CMP if this bit is set to1. 0 = EPWM Compare Down count point trigger DAC function Disabled. 1 = EPWM Compare Down count point trigger DAC function Enabled. **Note 1:** This bit should keep at 0 when EPWM counter operating in up counter type. **Note 2:** In complementary mode, CDTRGEN1, 3, 5 is used as another CDTRGEN for channel 0, 2, 4.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**CUTRGENn**	**EPWM Compare Up Count Point Trigger DAC Enable Bits** EPWM can trigger DAC to start action when EPWM counter counts up to CMP if this bit is set to1. 0 = EPWM Compare Up point trigger DAC function Disabled. 1 = EPWM Compare Up point trigger DAC function Enabled. **Note 1:** This bit should keep at 0 when EPWM counter operating in down counter type. **Note 2:** In complementary mode, CUTRGEN1, 3, 5 is used as another CUTRGEN for channel 0, 2, 4.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**PTEn**	**EPWM Period Point Trigger DAC Enable Bits** EPWM can trigger DAC to start action when EPWM counter counts up to (PERIODn+1) if this bit is set to1. 0 = EPWM period point trigger DAC function Disabled. 1 = EPWM period point trigger DAC function Enabled.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**ZTEn**	**EPWM Zero Point Trigger DAC Enable Bits** EPWM can trigger DAC to start action when EPWM counter down count to zero if this bit is set to1.

		0 = EPWM period point trigger DAC function Disabled. 1 = EPWM period point trigger DAC function Enabled.

EPWM Trigger EADC Source Select Register 0 (EPWM_EADCTS0)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCTS0	EPWMx_BA+0xF8	R/W	EPWM Trigger EADC Source Select Register 0	0x0000_0000

31	30	29	28	27	26	25	24
TRGEN3	**Reserved**		**TRGSEL3**				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TRGEN2	**Reserved**		**TRGSEL2**				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TRGEN1	**Reserved**		**TRGSEL1**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TRGEN0	**Reserved**		**TRGSEL0**				

Bits	Description	
[31]	**TRGEN3**	**EPWM_CH3 Trigger EADC Enable Bit** 0 = EPWM_CH3 Trigger EADC function Disabled. 1 = EPWM_CH3 Trigger EADC function Enabled.
[30:29]	**Reserved**	Reserved.
[28:24]	**TRGSEL3**	**EPWM_CH3 Trigger EADC Source Select** 00000 = EPWM_CH2 zero point. 00001 = EPWM_CH2 period point. 00010 = EPWM_CH2 zero or period point. 00011 = EPWM_CH2 up-count compared point. 00100 = EPWM_CH2 down-count compared point. 00101 = EPWM_CH3 zero point. 00110 = EPWM_CH3 period point. 00111 = EPWM_CH3 zero or period point. 01000 = EPWM_CH3 up-count compared point. 01001 = EPWM_CH3 down-count compared point. 01010 = EPWM_CH0 up-count free trigger compared point. 01011 = EPWM_CH0 down-count free trigger compared point. 01100 = EPWM_CH2 up-count free trigger compared point. 01101 = EPWM_CH2 down-count free trigger compared point. 01110 = EPWM_CH4 up-count free trigger compared point. 01111 = EPWM_CH4 down-count free trigger compared point. 10000 = EPWM_CH2 Interrupt Flag Accumulator Interrupt. 10001 = EPWM_CH3 Interrupt Flag Accumulator Interrupt.
[23]	**TRGEN2**	**EPWM_CH2 Trigger EADC Enable Bit** 0 = EPWM_CH2 Trigger EADC function Disabled. 1 = EPWM_CH2 Trigger EADC function Enabled.
[22:21]	**Reserved**	Reserved.

[20:16]	**TRGSEL2**	**EPWM_CH2 Trigger EADC Source Select** 00000 = EPWM_CH2 zero point. 00001 = EPWM_CH2 period point. 00010 = EPWM_CH2 zero or period point. 00011 = EPWM_CH2 up-count compared point. 00100 = EPWM_CH2 down-count compared point. 00101 = EPWM_CH3 zero point. 00110 = EPWM_CH3 period point. 00111 = EPWM_CH3 zero or period point. 01000 = EPWM_CH3 up-count compared point. 01001 = EPWM_CH3 down-count compared point. 01010 = EPWM_CH0 up-count free trigger compared point. 01011 = EPWM_CH0 down-count free trigger compared point. 01100 = EPWM_CH2 up-count free trigger compared point. 01101 = EPWM_CH2 down-count free trigger compared point. 01110 = EPWM_CH4 up-count free trigger compared point. 01111 = EPWM_CH4 down-count free trigger compared point. 10000 = EPWM_CH2 Interrupt Flag Accumulator Interrupt. 10001 = EPWM_CH3 Interrupt Flag Accumulator Interrupt.
[15]	**TRGEN1**	**EPWM_CH1 Trigger EADC Enable Bit** 0 = EPWM_CH1 Trigger EADC function Disabled. 1 = EPWM_CH1 Trigger EADC function Enabled.
[14:13]	**Reserved**	Reserved.
[12:8]	**TRGSEL1**	**EPWM_CH1 Trigger EADC Source Select** 00000 = EPWM_CH0 zero point. 00001 = EPWM_CH0 period point. 00010 = EPWM_CH0 zero or period point. 00011 = EPWM_CH0 up-count compared point. 00100 = EPWM_CH0 down-count compared point. 00101 = EPWM_CH1 zero point. 00110 = EPWM_CH1 period point. 00111 = EPWM_CH1 zero or period point. 01000 = EPWM_CH1 up-count compared point. 01001 = EPWM_CH1 down-count compared point. 01010 = EPWM_CH0 up-count free trigger compared point. 01011 = EPWM_CH0 down-count free trigger compared point. 01100 = EPWM_CH2 up-count free trigger compared point. 01101 = EPWM_CH2 down-count free trigger compared point. 01110 = EPWM_CH4 up-count free trigger compared point. 01111 = EPWM_CH4 down-count free trigger compared point. 10000 = EPWM_CH0 Interrupt Flag Accumulator Interrupt. 10001 = EPWM_CH1 Interrupt Flag Accumulator Interrupt.
[7]	**TRGEN0**	**EPWM_CH0 Trigger EADC Enable Bit** 0 = EPWM_CH0 Trigger EADC function Disabled. 1 = EPWM_CH0 Trigger EADC function Enabled.
[6:5]	**Reserved**	Reserved.

[4:0]	**TRGSEL0**	**EPWM_CH0 Trigger EADC Source Select** 00000 = EPWM_CH0 zero point. 00001 = EPWM_CH0 period point. 00010 = EPWM_CH0 zero or period point. 00011 = EPWM_CH0 up-count compared point. 00100 = EPWM_CH0 down-count compared point. 00101 = EPWM_CH1 zero point. 00110 = EPWM_CH1 period point. 00111 = EPWM_CH1 zero or period point. 01000 = EPWM_CH1 up-count compared point. 01001 = EPWM_CH1 down-count compared point. 01010 = EPWM_CH0 up-count free trigger compared point. 01011 = EPWM_CH0 down-count free trigger compared point. 01100 = EPWM_CH2 up-count free trigger compared point. 01101 = EPWM_CH2 down-count free trigger compared point. 01110 = EPWM_CH4 up-count free trigger compared point. 01111 = EPWM_CH4 down-count free trigger compared point. 10000 = EPWM_CH0 Interrupt Flag Accumulator Interrupt. 10001 = EPWM_CH1 Interrupt Flag Accumulator Interrupt.

EPWM Trigger EADC Source Select Register 1 (EPWM_EADCTS1)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCTS1	EPWMx_BA+0xFC	R/W	EPWM Trigger EADC Source Select Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TRGEN5	Reserved		TRGSEL5				
7	6	5	4	3	2	1	0
TRGEN4	Reserved		TRGSEL4				

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**TRGEN5**	**EPWM_CH5 Trigger EADC Enable Bit** 0 = EPWM_CH5 Trigger EADC function Disabled. 1 = EPWM_CH5 Trigger EADC function Enabled.
[14:13]	**Reserved**	Reserved.
[12:8]	**TRGSEL5**	**EPWM_CH5 Trigger EADC Source Select** 00000 = EPWM_CH4 zero point. 00001 = EPWM_CH4 period point. 00010 = EPWM_CH4 zero or period point. 00011 = EPWM_CH4 up-count compared point. 00100 = EPWM_CH4 down-count compared point. 00101 = EPWM_CH5 zero point. 00110 = EPWM_CH5 period point. 00111 = EPWM_CH5 zero or period point. 01000 = EPWM_CH5 up-count compared point. 01001 = EPWM_CH5 down-count compared point. 01010 = EPWM_CH0 up-count free trigger compared point. 01011 = EPWM_CH0 down-count free trigger compared point. 01100 = EPWM_CH2 up-count free trigger compared point. 01101 = EPWM_CH2 down-count free trigger compared point. 01110 = EPWM_CH4 up-count free trigger compared point. 01111 = EPWM_CH4 down-count free trigger compared point. 10000 = EPWM_CH4 Interrupt Flag Accumulator Interrupt. 10001 = EPWM_CH5 Interrupt Flag Accumulator Interrupt.
[7]	**TRGEN4**	**EPWM_CH4 Trigger EADC Enable Bit** 0 = EPWM_CH4 Trigger EADC function Disabled. 1 = EPWM_CH4 Trigger EADC function Enabled.

[6:5]	**Reserved**	Reserved.
[4:0]	**TRGSEL4**	**EPWM_CH4 Trigger EADC Source Select** 00000 = EPWM_CH4 zero point. 00001 = EPWM_CH4 period point. 00010 = EPWM_CH4 zero or period point. 00011 = EPWM_CH4 up-count compared point. 00100 = EPWM_CH4 down-count compared point. 00101 = EPWM_CH5 zero point. 00110 = EPWM_CH5 period point. 00111 = EPWM_CH5 zero or period point. 01000 = EPWM_CH5 up-count compared point. 01001 = EPWM_CH5 down-count compared point. 01010 = EPWM_CH0 up-count free trigger compared point. 01011 = EPWM_CH0 down-count free trigger compared point. 01100 = EPWM_CH2 up-count free trigger compared point. 01101 = EPWM_CH2 down-count free trigger compared point. 01110 = EPWM_CH4 up-count free trigger compared point. 01111 = EPWM_CH4 down-count free trigger compared point. 10000 = EPWM_CH4 Interrupt Flag Accumulator Interrupt. 10001 = EPWM_CH5 Interrupt Flag Accumulator Interrupt.

EPWM Free Trigger Compare Register 0_1, 2_3, 4_5 (EPWM_FTCMPDAT0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_FTCMPDAT0_1	EPWMx_BA+0x100	R/W	EPWM Free Trigger Compare Register 0/1	0x0000_0000
EPWM_FTCMPDAT2_3	EPWMx_BA+0x104	R/W	EPWM Free Trigger Compare Register 2/3	0x0000_0000
EPWM_FTCMPDAT4_5	EPWMx_BA+0x108	R/W	EPWM Free Trigger Compare Register 4/5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FTCMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FTCMP							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FTCMP**	**EPWM Free Trigger Compare Register** FTCMP use to compare with even CNT (EPWM_CNTm[15:0], m=0,2,4) to trigger EADC. EPWM_FTCMPDAT0_1, EPWM_FTCMPDAT2_3, EPWM_FTCMPDAT4_5 corresponding complementary pairs EPWM_CH0and EPWM_CH1, EPWM_CH2 and EPWM_CH3, EPWM_CH4 and EPWM_CH5.

EPWM Synchronous Start Control Register (EPWM_SSCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_SSCTL	EPWMx_BA+0x110	R/W	EPWM Synchronous Start Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						SSRC	
7	6	5	4	3	2	1	0
Reserved		SSEN5	SSEN4	SSEN3	SSEN2	SSEN1	SSEN0

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:8]	**SSRC**	**EPWM Synchronous Start Source Select Bits** 00 = Synchronous start source come from EPWM0. 01 = Synchronous start source come from EPWM1. 10 = Synchronous start source come from BPWM0. 11 = Synchronous start source come from BPWM1.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**SSENn**	**EPWM Synchronous Start Function Enable Bits** When synchronous start function is enabled, the EPWM counter enable register (EPWM_CNTEN) can be enabled by writing EPWM synchronous start trigger bit (CNTSEN). 0 = EPWM synchronous start function Disabled. 1 = EPWM synchronous start function Enabled.

EPWM Synchronous Start Trigger Register (EPWM_SSTRG)

Register	Offset	R/W	Description	Reset Value
EPWM_SSTRG	EPWMx_BA+0x114	W	EPWM Synchronous Start Trigger Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTSEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTSEN**	**EPWM Counter Synchronous Start Enable (Write Only)** PMW counter synchronous enable function is used to make selected EPWM channels (include EPWM0_CHx and EPWM1_CHx) start counting at the same time. Writing this bit to 1 will also set the counter enable bit (CNTENn, n denotes channel 0 to 5) if correlated EPWM channel counter synchronous start function is enabled.

EPWM Leading Edge Blanking Control Register (EPWM_LEBCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_LEBCTL	EPWMx_BA+0x118	R/W	EPWM Leading Edge Blanking Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						TRGTYPE	
15	14	13	12	11	10	9	8
Reserved					SRCEN4	SRCEN2	SRCEN0
7	6	5	4	3	2	1	0
Reserved							LEBEN

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**TRGTYPE**	**EPWM Leading Edge Blanking Trigger Type** 0 = When detect leading edge blanking source rising edge, blanking counter start counting. 1 = When detect leading edge blanking source falling edge, blanking counter start counting. 2 = When detect leading edge blanking source rising or falling edge, blanking counter start counting. 3 = Reserved.
[15:11]	**Reserved**	Reserved.
[10]	**SRCEN4**	**EPWM Leading Edge Blanking Source From EPWM_CH4 Enable Bit** 0 = EPWM Leading Edge Blanking Source from EPWM_CH4 Disabled. 1 = EPWM Leading Edge Blanking Source from EPWM_CH4 Enabled.
[9]	**SRCEN2**	**EPWM Leading Edge Blanking Source From EPWM_CH2 Enable Bit** 0 = EPWM Leading Edge Blanking Source from EPWM_CH2 Disabled. 1 = EPWM Leading Edge Blanking Source from EPWM_CH2 Enabled.
[8]	**SRCEN0**	**EPWM Leading Edge Blanking Source From EPWM_CH0 Enable Bit** 0 = EPWM Leading Edge Blanking Source from EPWM_CH0 Disabled. 1 = EPWM Leading Edge Blanking Source from EPWM_CH0 Enabled.
[7:1]	**Reserved**	Reserved.
[0]	**LEBEN**	**EPWM Leading Edge Blanking Enable Bit** 0 = EPWM Leading Edge Blanking Disabled. 1 = EPWM Leading Edge Blanking Enabled.

EPWM Leading Edge Blanking Counter Register (EPWM_LEBCNT)

Register	Offset	R/W	Description	Reset Value
EPWM_LEBCNT	EPWMx_BA+0x11C	R/W	EPWM Leading Edge Blanking Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							LEBCNT
7	6	5	4	3	2	1	0
LEBCNT							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**LEBCNT**	**EPWM Leading Edge Blanking Counter** This counter value decides leading edge blanking window size. Blanking window size = LEBCNT+1, and LEB counter clock base is ECLK.

EPWM Status Register (EPWM_STATUS)

Register	Offset	R/W	Description	Reset Value
EPWM_STATUS	EPWMx_BA+0x120	R/W	EPWM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							DACTRGF
23	22	21	20	19	18	17	16
Reserved		EADCTRGF5	EADCTRGF4	EADCTRGF3	EADCTRGF2	EADCTRGF1	EADCTRGF0
15	14	13	12	11	10	9	8
Reserved					SYNCINF4	SYNCINF2	SYNCINF0
7	6	5	4	3	2	1	0
Reserved		CNTMAXF5	CNTMAXF4	CNTMAXF3	CNTMAXF2	CNTMAXF1	CNTMAXF0

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**DACTRGF**	**DAC Start of Conversion Flag** 0 = No DAC start of conversion trigger event has occurred. 1 = A DAC start of conversion trigger event has occurred. **Note:** This bit can be cleared by software writing 1.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**EADCTRGFn**	**EADC Start of Conversion Flag** 0 = No EADC start of conversion trigger event has occurred. 1 = An EADC start of conversion trigger event has occurred. **Note:** This bit can be cleared by software writing 1.
[15:11]	**Reserved**	Reserved.
[8+n/2] n=0,2,4	**SYNCINFn**	**Input Synchronization Latched Flag** 0 = No SYNC_IN event has occurred. 1 = A SYNC_IN event has occurred. **Note:** This bit can be cleared by software writing 1.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CNTMAXFn**	**Time-base Counter Equal to 0xFFFF Latched Flag** 0 = The time-base counter never reached its maximum value 0xFFFF. 1 = The time-base counter reached its maximum value. **Note:** This bit can be cleared by software writing 1.

EPWM Interrupt Flag Accumulator Register (EPWM_IFAn)

Register	Offset	R/W	Description	Reset Value
EPWM_IFA0	EPWMx_BA+0x130	R/W	EPWM Interrupt Flag Accumulator Register 0	0x0000_0000
EPWM_IFA1	EPWMx_BA+0x134	R/W	EPWM Interrupt Flag Accumulator Register 1	0x0000_0000
EPWM_IFA2	EPWMx_BA+0x138	R/W	EPWM Interrupt Flag Accumulator Register 2	0x0000_0000
EPWM_IFA3	EPWMx_BA+0x13C	R/W	EPWM Interrupt Flag Accumulator Register 3	0x0000_0000
EPWM_IFA4	EPWMx_BA+0x140	R/W	EPWM Interrupt Flag Accumulator Register 4	0x0000_0000
EPWM_IFA5	EPWMx_BA+0x144	R/W	EPWM Interrupt Flag Accumulator Register 5	0x0000_0000

31	30	29	28	27	26	25	24
IFAEN	Reserved	IFASEL		Reserved			STPMOD
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IFACNT							
7	6	5	4	3	2	1	0
IFACNT							

Bits	Description	
[31]	**IFAEN**	**EPWM_CHn Interrupt Flag Accumulator Enable Bits** 0 = EPWM_CHn interrupt flag accumulator Disabled. 1 = EPWM_CHn interrupt flag accumulator Enabled.
[30]	**Reserved**	Reserved.
[29:28]	**IFASEL**	**EPWM_CHn Interrupt Flag Accumulator Source Select** 00 = EPWM_CHn zero point. 01 = EPWM_CHn period in channel n. 10 = EPWM_CHn up-count compared point. 11 = EPWM_CHn down-count compared point.
[27:25]	**Reserved**	Reserved.
[24]	**STPMOD**	**EPWM_CHn Accumulator Stop Mode Enable Bits** 0 = EPWM_CHn Stop Mode Disabled. 1 = EPWM_CHn Stop Mode Enabled.
[23:16]	**Reserved**	Reserved.
[15:0]	**IFACNT**	**EPWM_CHn Interrupt Flag Counter** The register sets the count number which defines (IFACNT+1) times of EPWM_CHn period occurs to set bit IFAIFn to request the EPWM period interrupt. EPWM flag will be set in every IFACNT[15:0] times of EPWM period.

EPWM Accumulator Interrupt Flag Register (EPWM_AINTSTS)

Register	Offset	R/W	Description	Reset Value
EPWM_AINTSTS	EPWMx_BA+0x150	R/W	EPWM Accumulator Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		IFAIF5	IFAIF4	IFAIF3	IFAIF2	IFAIF1	IFAIF0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**IFAIFn**	**EPWM_CHn Interrupt Flag Accumulator Interrupt Flag** Flag is set by hardware when condition match IFASEL in EPWM_IFAn register, software can clear this bit by writing 1 to it.

EPWM Accumulator Interrupt Enable Register (EPWM_AINTEN)

Register	Offset	R/W	Description	Reset Value
EPWM_AINTEN	EPWMx_BA+0x154	R/W	EPWM Accumulator Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**IFAIEN5**	**IFAIEN4**	**IFAIEN3**	**IFAIEN2**	**IFAIEN1**	**IFAIEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**IFAIENn**	**EPWM_CHn Interrupt Flag Accumulator Interrupt Enable Bits** 0 = Interrupt Flag accumulator interrupt Disabled. 1 = Interrupt Flag accumulator interrupt Enabled.

EPWM Accumulator PDMA Control Register (EPWM_APDMACTL)

Register	Offset	R/W	Description	Reset Value
EPWM_APDMACTL	EPWMx_BA+0x158	R/W	EPWM Accumulator PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		APDMAEN5	APDMAEN4	APDMAEN3	APDMAEN2	APDMAEN1	APDMAEN0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**APDMAENn**	**Channel n Accumulator PDMA Enable Bits** 0 = Channel n PDMA function Disabled. 1 = Channel n PDMA function Enabled for the channel n to trigger PDMA to transfer memory data to register.

EPWM Fault Detect Enable Register (EPWM_FDEN)

Register	Offset	R/W	Description	Reset Value
EPWM_FDEN	EPWMx_BA+0x160	R/W	EPWM Fault Detect Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		FDCKS5	FDCKS4	FDCKS3	FDCKS2	FDCKS1	FDCKS0
15	14	13	12	11	10	9	8
Reserved		FDODIS5	FDODIS4	FDODIS3	FDODIS2	FDODIS1	FDODIS0
7	6	5	4	3	2	1	0
Reserved		FDEN5	FDEN4	FDEN3	FDEN2	FDEN1	FDEN0

Bits	Description	
[31:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**FDCKSn**	**EPWM Channel n Fault Detect Clock Source Select Bit** 0 = EPWMx_CLK, x denotes 0 or 1. 1 = EPWMx_CLK divide by prescaler, x denotes 0 or 1.
[8+n] n=0,1..5	**FDODISn**	**EPWM Channel n Output Fault Detect Disable Bit** 0 = EPWM detect fault and output Enabled. 1 = EPWM detect fault and output Disabled.
[n] n=0,1..5	**FDENn**	**EPWM Fault Detect Function Enable Bit** 0 = Fault detect function Disabled. 1 = Fault detect function Enabled.

EPWM Fault Detect Control Register 0~5 (EPWM_FDCTL0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_FDCTL0	EPWMx_BA+0x164	R/W	EPWM Fault Detect Control Register 0	0x0000_0000
EPWM_FDCTL1	EPWMx_BA+0x168	R/W	EPWM Fault Detect Control Register 1	0x0000_0000
EPWM_FDCTL2	EPWMx_BA+0x16C	R/W	EPWM Fault Detect Control Register 2	0x0000_0000
EPWM_FDCTL3	EPWMx_BA+0x170	R/W	EPWM Fault Detect Control Register 3	0x0000_0000
EPWM_FDCTL4	EPWMx_BA+0x174	R/W	EPWM Fault Detect Control Register 4	0x0000_0000
EPWM_FDCTL5	EPWMx_BA+0x178	R/W	EPWM Fault Detect Control Register 5	0x0000_0000

31	30	29	28	27	26	25	24
FDDGEN	**Reserved**	**FDCKSEL**		**Reserved**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**DGSMPCYC**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FDMSKEN	**Reserved**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**TRMSKCNT**						

Bits	Description	
[31]	**FDDGEN**	**Fault Detect Deglitch Enable Bit** 0 = Fault detect deglitch function Disabled. 1 = Fault detect deglitch function Enabled.
[30]	**Reserved**	Reserved.
[29:28]	**FDCKSEL**	**EPWM Channel Fault Detect Clock Select** 00 = FLT_CLK/1. 01 = FLT_CLK/2. 10 = FLT_CLK/4. 11 = FLT_CLK/8. **Note:** FLT_CLK is FDCKSn (EPWM_FDENn[16+n], n=0,1..5) selected clock.
[27:19]	**Reserved**	Reserved.
[18:16]	**DGSMPCYC**	**Deglitch Sampling Cycle** 1. FDCKS is set to 0: Sampling detect signal each EPWMx_CLK * (2^FDCKSEL) period and detect DGSMPCYC+1 times 2. FDCKS is set to 1: Sampling detect signal each EPWMx_CLK * CLKPSC * (2^FDCKSEL) period and detect DGSMPCYC+1 times **Note:** CLKPSC (EPWM_CLKPSCn[11:0]) is 0:

		TRMSKCNT >= DGSMPCYC + 2. FDCKS is 1 and CLKPSC (EPWM_CLKPSCn[11:0]) is 1: TRMSKCNT >= DGSMPCYC + 1. FDCKS is 1 and CLKPSC (EPWM_CLKPSCn[11:0]) is 2: TRMSKCNT >= DGSMPCYC.
[15]	**FDMSKEN**	**Fault Detect Mask Enable Bit** 0 = Fault detect mask function Disabled. 1 = Fault detect mask function Enabled.
[14:7]	**Reserved**	Reserved.
[6:0]	**TRMSKCNT**	**Transition Mask Counter** The fault detect result will be masked before counter count from 0 to TRMSKCNT. 3. FDCKS is set to 0: Mask time is EPWMx_CLK * (2^FDCKSEL) * (TRMSKCNT+2) 4. FDCKS is set to 1: Mask time EPWMx_CLK * CLKPSC * (2^FDCKSEL) * (TRMSKCNT+2) **Note:** CLKPSC (EPWM_CLKPSCn[11:0]) is 0: TRMSKCNT >= DGSMPCYC + 2. FDCKS is 1 and CLKPSC (EPWM_CLKPSCn[11:0]) is 1: TRMSKCNT >= DGSMPCYC + 1. FDCKS is 1 and CLKPSC (EPWM_CLKPSCn[11:0]) is 2: TRMSKCNT >= DGSMPCYC.

EPWM Fault Detect Interrupt Enable Register (EPWM_FDIEN)

Register	Offset	R/W	Description	Reset Value
EPWM_FDIEN	EPWMx_BA+0x17C	R/W	EPWM Fault Detect Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**FDIEN5**	**FDIEN4**	**FDIEN3**	**FDIEN2**	**FDIEN1**	**FDIEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n]	**FDIENn**	**EPWM Channel n Fault Detect Interrupt Enable Bit** 0 = EPWM Channel n Fault Detect Interrupt Disabled. 1 = EPWM Channel n Fault Detect Interrupt Enabled.

EPWM Fault Detect Interrupt Flag Register (EPWM_FDSTS)

Register	Offset	R/W	Description	Reset Value
EPWM_FDSTS	EPWMx_BA+0x180	R/W	EPWM Fault Detect Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		FDIF5	FDIF4	FDIF3	FDIF2	FDIF1	FDIF0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**FDIFn**	**EPWM Channel n Fault Detect Interrupt Flag Bit** Fault Detect Interrupt Flag will be set when EPWM output short. Software can clear this bit by writing 1 to it.

EPWM Trigger EADC Prescale Control Register (EPWM_EADCPSCCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSCCTL	EPWMx_BA+0x184	R/W	EPWM Trigger EADC Prescale Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		PSCEN5	PSCEN4	PSCEN3	PSCEN2	PSCEN1	PSCEN0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**PSCENn**	**EPWM Trigger EADC Pre-scale Function Enable Bits** 0 = EPWM Trigger EADC Pre-scale Function Disabled. 1 = EPWM Trigger EADC Pre-scale Function Enabled.

EPWM Trigger EADC Prescale Register 0 (EPWM_EADCPSC0)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSC0	EPWMx_BA+0x188	R/W	EPWM Trigger EADC Prescale Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**EADCPSC3**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**EADCPSC2**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**EADCPSC1**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**EADCPSC0**			

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**EADCPSC3**	**EPWM Channel 3 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC3+1) times of EPWM_CH3 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF3.
[23:20]	**Reserved**	Reserved.
[19:16]	**EADCPSC2**	**EPWM Channel 2 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC2+1) times of EPWM_CH2 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF2.
[15:12]	**Reserved**	Reserved.
[11:8]	**EADCPSC1**	**EPWM Channel 1 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC1+1) times of EPWM_CH1 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF1.
[7:4]	**Reserved**	Reserved.
[3:0]	**EADCPSC0**	**EPWM Channel 0 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC0+1) times of EPWM_CH0 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF0.

EPWM Trigger EADC Prescale Register 1 (EPWM_EADCPSC1)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSC1	EPWMx_BA+0x18C	R/W	EPWM Trigger EADC Prescale Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**EADCPSC5**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**EADCPSC4**			

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**EADCPSC5**	**EPWM Channel 5 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC5+1) times of EPWM_CH5 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF5.
[7:4]	**Reserved**	Reserved.
[3:0]	**EADCPSC4**	**EPWM Channel 4 Trigger EADC Prescale** The register sets the count number which defines (EADCPSC4+1) times of EPWM_CH4 trigger EADC event occurs to trigger EADC and set trigger EADC flag bit EADCTRGF4.

EPWM Trigger EADC Prescale Counter Register 0 (EPWM_EADCPSCNT0)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSCNT0	EPWMx_BA+0x190	R/W	EPWM Trigger EADC Prescale Counter Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**PSCNT3**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**PSCNT2**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**PSCNT1**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PSCNT0**			

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**PSCNT3**	**EPWM Trigger EADC Prescale Counter 3** User can monitor PSCNT3 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN3 is 0. **Note 2:** Write data limitation: PSCNT3 < EADCPSC3.
[23:20]	**Reserved**	Reserved.
[19:16]	**PSCNT2**	**EPWM Trigger EADC Prescale Counter 2** User can monitor PSCNT2 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN2 is 0. **Note 2:** Write data limitation: PSCNT2 < EADCPSC2.
[15:12]	**Reserved**	Reserved.
[11:8]	**PSCNT1**	**EPWM Trigger EADC Prescale Counter 1** User can monitor PSCNT1 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN1 is 0. **Note 2:** Write data limitation: PSCNT1 < EADCPSC1.
[7:4]	**Reserved**	Reserved.
[3:0]	**PSCNT0**	**EPWM Trigger EADC Prescale Counter 0** User can monitor PSCNT0 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** user can write only when PSCEN0 is 0. **Note 2:** Write data limitation: PSCNT0 < EADCPSC0.

EPWM Trigger EADC Prescale Counter Register 1 (EPWM_EADCPSCNT1)

Register	Offset	R/W	Description	Reset Value
EPWM_EADCPSCNT1	EPWMx_BA+0x194	R/W	EPWM Trigger EADC Prescale Counter Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**PSCNT5**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PSCNT4**			

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**PSCNT5**	**EPWM Trigger EADC Prescale Counter 5** User can monitor PSCNT5 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** User can write only when PSCEN5 is 0. **Note 2:** Write data limitation: PSCNT5 < EADCPSC5.
[7:4]	**Reserved**	Reserved.
[3:0]	**PSCNT4**	**EPWM Trigger EADC Prescale Counter 4** User can monitor PSCNT4 to know the current value in 4-bit trigger EADC prescale counter. **Note 1:** User can write only when PSCEN4 is 0. **Note 2:** Write data limitation: PSCNT4 < EADCPSC4.

EPWM Capture Input Enable Register (EPWM_CAPINEN)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPINEN	EPWMx_BA+0x200	R/W	EPWM Capture Input Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CAPINEN5**	**CAPINEN4**	**CAPINEN3**	**CAPINEN2**	**CAPINEN1**	**CAPINEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPINENn**	**Capture Input Enable Bits** 0 = EPWM Channel capture input path Disabled. The input of EPWM channel capture function is always regarded as 0. 1 = EPWM Channel capture input path Enabled. The input of EPWM channel capture function comes from correlative multifunction pin.

EPWM Capture Control Register (EPWM_CAPCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPCTL	EPWMx_BA+0x204	R/W	EPWM Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		FCRLDEN5	FCRLDEN4	FCRLDEN3	FCRLDEN2	FCRLDEN1	FCRLDEN0
23	22	21	20	19	18	17	16
Reserved		RCRLDEN5	RCRLDEN4	RCRLDEN3	RCRLDEN2	RCRLDEN1	RCRLDEN0
15	14	13	12	11	10	9	8
Reserved		CAPINV5	CAPINV4	CAPINV3	CAPINV2	CAPINV1	CAPINV0
7	6	5	4	3	2	1	0
Reserved		CAPEN5	CAPEN4	CAPEN3	CAPEN2	CAPEN1	CAPEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[24+n] n=0,1..5	**FCRLDENn**	**Falling Capture Reload Enable Bits** 0 = Falling capture reload counter Disabled. 1 = Falling capture reload counter Enabled.
[23:22]	**Reserved**	Reserved.
[16+n] n=0,1..5	**RCRLDENn**	**Rising Capture Reload Enable Bits** 0 = Rising capture reload counter Disabled. 1 = Rising capture reload counter Enabled.
[15:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CAPINVn**	**Capture Inverter Enable Bits** 0 = Capture source inverter Disabled. 1 = Capture source inverter Enabled. Reverse the input signal from GPIO.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPENn**	**Capture Function Enable Bits** 0 = Capture function Disabled. EPWM_RCAPDATn/EPWM_FCAPDATn register will not be updated. 1 = Capture function Enabled. Capture latched the EPWM counter value when detected rising or falling edge of input signal and saved to RCAPDAT (Rising latch) and FCAPDAT (Falling latch).

EPWM Capture Status Register (EPWM_CAPSTS)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPSTS	EPWMx_BA+0x208	R	EPWM Capture Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CFLIFOV5**	**CFLIFOV4**	**CFLIFOV3**	**CFLIFOV2**	**CFLIFOV1**	**CFLIFOV0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CRLIFOV5**	**CRLIFOV4**	**CRLIFOV3**	**CRLIFOV2**	**CRLIFOV1**	**CRLIFOV0**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CFLIFOVn**	**Capture Falling Latch Interrupt Flag Overrun Status (Read Only)** This flag indicates if falling latch happened when the corresponding CFLIFn(EPWM_CAPIF[8+n]) is 1. **Note:** This bit will be cleared automatically when user clears corresponding CFLIFn(EPWM_CAPIF[8+n]).
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CRLIFOVn**	**Capture Rising Latch Interrupt Flag Overrun Status (Read Only)** This flag indicates if rising latch happened when the corresponding CRLIFn(EPWM_CAPIF[n]) is 1. **Note:** This bit will be cleared automatically when user clears corresponding CRLIFn(EPWM_CAPIF[n]).

EPWM Rising Capture Data Register 0~5 (EPWM_RCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_RCAPDAT0	EPWMx_BA+0x20C	R	EPWM Rising Capture Data Register 0	0x0000_0000
EPWM_RCAPDAT1	EPWMx_BA+0x214	R	EPWM Rising Capture Data Register 1	0x0000_0000
EPWM_RCAPDAT2	EPWMx_BA+0x21C	R	EPWM Rising Capture Data Register 2	0x0000_0000
EPWM_RCAPDAT3	EPWMx_BA+0x224	R	EPWM Rising Capture Data Register 3	0x0000_0000
EPWM_RCAPDAT4	EPWMx_BA+0x22C	R	EPWM Rising Capture Data Register 4	0x0000_0000
EPWM_RCAPDAT5	EPWMx_BA+0x234	R	EPWM Rising Capture Data Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RCAPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RCAPDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RCAPDAT**	**EPWM Rising Capture Data Register (Read Only)** When rising capture condition happened, the EPWM counter value will be saved in this register.

EPWM Falling Capture Data Register 0~5 (EPWM_FCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_FCAPDAT0	EPWMx_BA+0x210	R	EPWM Falling Capture Data Register 0	0x0000_0000
EPWM_FCAPDAT1	EPWMx_BA+0x218	R	EPWM Falling Capture Data Register 1	0x0000_0000
EPWM_FCAPDAT2	EPWMx_BA+0x220	R	EPWM Falling Capture Data Register 2	0x0000_0000
EPWM_FCAPDAT3	EPWMx_BA+0x228	R	EPWM Falling Capture Data Register 3	0x0000_0000
EPWM_FCAPDAT4	EPWMx_BA+0x230	R	EPWM Falling Capture Data Register 4	0x0000_0000
EPWM_FCAPDAT5	EPWMx_BA+0x238	R	EPWM Falling Capture Data Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FCAPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FCAPDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FCAPDAT**	**EPWM Falling Capture Data Register (Read Only)** When falling capture condition happened, the EPWM counter value will be saved in this register.

EPWM PDMA Control Register (EPWM_PDMACTL)

Register	Offset	R/W	Description	Reset Value
EPWM_PDMACTL	EPWMx_BA+0x23C	R/W	EPWM PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			CHSEL4_5	CAPORD4_5	CAPMOD4_5		CHEN4_5
15	14	13	12	11	10	9	8
Reserved			CHSEL2_3	CAPORD2_3	CAPMOD2_3		CHEN2_3
7	6	5	4	3	2	1	0
Reserved			CHSEL0_1	CAPORD0_1	CAPMOD0_1		CHEN0_1

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**CHSEL4_5**	**Select Channel 4/5 to Do PDMA Transfer** 0 = Channel4. 1 = Channel5.
[19]	**CAPORD4_5**	**Capture Channel 4/5 Rising/Falling Order** Set this bit to determine whether the EPWM_RCAPDAT4/5 or EPWM_FCAPDAT4/5 is the first captured data transferred to memory through PDMA when CAPMOD4_5 =11. 0 = EPWM_FCAPDAT4/5 is the first captured data to memory. 1 = EPWM_RCAPDAT4/5 is the first captured data to memory.
[18:17]	**CAPMOD4_5**	**Select EPWM_RCAPDAT4/5 or EPWM_FCAPDAT4/5 to Do PDMA Transfer** 00 = Reserved. 01 = EPWM_RCAPDAT4/5. 10 = EPWM_FCAPDAT4/5. 11 = Both EPWM_RCAPDAT4/5 and EPWM_FCAPDAT4/5.
[16]	**CHEN4_5**	**Channel 4/5 PDMA Enable Bit** 0 = Channel 4/5 PDMA function Disabled. 1 = Channel 4/5 PDMA function Enabled for the channel 4/5 captured data and transfer to memory.
[15:13]	**Reserved**	Reserved.
[12]	**CHSEL2_3**	**Select Channel 2/3 to Do PDMA Transfer** 0 = Channel2. 1 = Channel3.
[11]	**CAPORD2_3**	**Capture Channel 2/3 Rising/Falling Order** Set this bit to determine whether the EPWM_RCAPDAT2/3 or EPWM_FCAPDAT2/3 is the first captured data transferred to memory through PDMA when CAPMOD2_3 =11. 0 = EPWM_FCAPDAT2/3 is the first captured data to memory.

		1 = EPWM_RCAPDAT2/3 is the first captured data to memory.
[10:9]	**CAPMOD2_3**	**Select EPWM_RCAPDAT2/3 or EPWM_FCAODAT2/3 to Do PDMA Transfer** 00 = Reserved. 01 = EPWM_RCAPDAT2/3. 10 = EPWM_FCAPDAT2/3. 11 = Both EPWM_RCAPDAT2/3 and EPWM_FCAPDAT2/3.
[8]	**CHEN2_3**	**Channel 2/3 PDMA Enable Bit** 0 = Channel 2/3 PDMA function Disabled. 1 = Channel 2/3 PDMA function Enabled for the channel 2/3 captured data and transfer to memory.
[7:5]	**Reserved**	Reserved.
[4]	**CHSEL0_1**	**Select Channel 0/1 to Do PDMA Transfer** 0 = Channel0. 1 = Channel1.
[3]	**CAPORD0_1**	**Capture Channel 0/1 Rising/Falling Order** Set this bit to determine whether the EPWM_RCAPDAT0/1 or EPWM_FCAPDAT0/1 is the first captured data transferred to memory through PDMA when CAPMOD0_1 =11. 0 = EPWM_FCAPDAT0/1 is the first captured data to memory. 1 = EPWM_RCAPDAT0/1 is the first captured data to memory.
[2:1]	**CAPMOD0_1**	**Select EPWM_RCAPDAT0/1 or EPWM_FCAPDAT0/1 to Do PDMA Transfer** 00 = Reserved. 01 = EPWM_RCAPDAT0/1. 10 = EPWM_FCAPDAT0/1. 11 = Both EPWM_RCAPDAT0/1 and EPWM_FCAPDAT0/1.
[0]	**CHEN0_1**	**Channel 0/1 PDMA Enable Bit** 0 = Channel 0/1 PDMA function Disabled. 1 = Channel 0/1 PDMA function Enabled for the channel 0/1 captured data and transfer to memory.

EPWM Capture Channel 0_1, 2_3, 4_5 PDMA Register (EPWM_PDMACAP 0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_PDMACAP0_1	EPWMx_BA+0x240	R	EPWM Capture Channel 01 PDMA Register	0x0000_0000
EPWM_PDMACAP2_3	EPWMx_BA+0x244	R	EPWM Capture Channel 23 PDMA Register	0x0000_0000
EPWM_PDMACAP4_5	EPWMx_BA+0x248	R	EPWM Capture Channel 45 PDMA Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CAPBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAPBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CAPBUF**	**EPWM Capture PDMA Register (Read Only)** This register is used as a buffer to transfer EPWM capture rising or falling data to memory by PDMA.

EPWM Capture Interrupt Enable Register (EPWM_CAPIEN)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPIEN	EPWMx_BA+0x250	R/W	EPWM Capture Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		CAPFIEN5	CAPFIEN4	CAPFIEN3	CAPFIEN2	CAPFIEN1	CAPFIEN0
7	6	5	4	3	2	1	0
Reserved		CAPRIEN5	CAPRIEN4	CAPRIEN3	CAPRIEN2	CAPRIEN1	CAPRIEN0

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CAPFIENn**	**EPWM Capture Falling Latch Interrupt Enable Bits** 0 = Capture falling edge latch interrupt Disabled. 1 = Capture falling edge latch interrupt Enabled.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CAPRIENn**	**EPWM Capture Rising Latch Interrupt Enable Bits** 0 = Capture rising edge latch interrupt Disabled. 1 = Capture rising edge latch interrupt Enabled.

EPWM Capture Interrupt Flag Register (EPWM_CAPIF)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPIF	EPWMx_BA+0x254	R/W	EPWM Capture Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CFLIF5**	**CFLIF4**	**CFLIF3**	**CFLIF2**	**CFLIF1**	**CFLIF0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CRLIF5**	**CRLIF4**	**CRLIF3**	**CRLIF2**	**CRLIF1**	**CRLIF0**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[8+n] n=0,1..5	**CFLIFn**	**EPWM Capture Falling Latch Interrupt Flag** 0 = No capture falling latch condition happened. 1 = Capture falling latch condition happened, and this flag will be set to high. **Note 1:** When Capture with PDMA operating, EPWM_CAPIF corresponding channel CFLIFn will be cleared by hardware after PDMA transfer data. **Note 2:** This bit is cleared by writing 1 to it.
[7:6]	**Reserved**	Reserved.
[n] n=0,1..5	**CRLIFn**	**EPWM Capture Rising Latch Interrupt Flag** 0 = No capture rising latch condition happened. 1 = Capture rising latch condition happened, this flag will be set to high. **Note 1:** When Capture with PDMA operating, EPWM_CAPIF corresponding channel CRLIFn will be cleared by hardware after PDMA transfer data. **Note 2:** This bit is cleared by writing 1 to it.

EPWM Capture Input Noise Filter Register (EPWM_CAPNF0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CAPNF0	EPWMx_BA+0x258	R/W	EPWM Capture Input Noise Filter Register 0	0x0000_0000
EPWM_CAPNF1	EPWMx_BA+0x25C	R/W	EPWM Capture Input Noise Filter Register 1	0x0000_0000
EPWM_CAPNF2	EPWMx_BA+0x260	R/W	EPWM Capture Input Noise Filter Register 2	0x0000_0000
EPWM_CAPNF3	EPWMx_BA+0x264	R/W	EPWM Capture Input Noise Filter Register 3	0x0000_0000
EPWM_CAPNF4	EPWMx_BA+0x268	R/W	EPWM Capture Input Noise Filter Register 4	0x0000_0000
EPWM_CAPNF5	EPWMx_BA+0x26C	R/W	EPWM Capture Input Noise Filter Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**CAPNFCNT**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**CAPNFSEL**			**Reserved**			**CAPNFEN**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:8]	**CAPNFCNT**	**Capture Edge Detector Noise Filter Count** The register bits control the capture filter counter to count from 0 to CAPNFCNT.
[6:4]	**CAPNFSEL**	**Capture Edge Detector Noise Filter Clock Selection** 000 = Filter clock = PCLK. 001 = Filter clock = PCLK/2. 010 = Filter clock = PCLK/4. 011 = Filter clock = PCLK/8. 100 = Filter clock = PCLK/16. 101 = Filter clock = PCLK/32. 110 = Filter clock = PCLK/64. 111 = Filter clock = PCLK/128.
[3:1]	**Reserved**	Reserved.
[0]	**CAPNFEN**	**Capture Noise Filter Enable** 0 = Capture Noise Filter function Disabled. 1 = Capture Noise Filter function Enabled.

EPWM External Event Trigger Control Register 0~5 (EPWMx_EXTETCTL0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_EXTETCTL0	EPWMx_BA+0x270	R/W	EPWM External Event Trigger Control Register 0	0x0000_0000
EPWM_EXTETCTL1	EPWMx_BA+0x274	R/W	EPWM External Event Trigger Control Register 1	0x0000_0000
EPWM_EXTETCTL2	EPWMx_BA+0x278	R/W	EPWM External Event Trigger Control Register 2	0x0000_0000
EPWM_EXTETCTL3	EPWMx_BA+0x27C	R/W	EPWM External Event Trigger Control Register 3	0x0000_0000
EPWM_EXTETCTL4	EPWMx_BA+0x280	R/W	EPWM External Event Trigger Control Register 4	0x0000_0000
EPWM_EXTETCTL5	EPWMx_BA+0x284	R/W	EPWM External Event Trigger Control Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				EXTTRGS			
7	6	5	4	3	2	1	0
Reserved		CNTACTS		Reserved			EXTETEN

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**EXTTRGS**	**External Trigger Selection** 0000 = INT0. 0001 = INT1. 0010 = INT2. 0011 = INT3. 0100 = INT4. 0101 = INT5. 0110 = INT6. 0111 = INT7. Other = Reserved.
[5:4]	**CNTACTS**	**Counter Action Selection** 00 = Counter reset. 01 = Counter start. 10 = Counter reset and start. 11 = Reserved.
[3:1]	**Reserved**	Reserved.
[0]	**EXETEN**	**External Event Trigger Enable Bit** 0 = External Event Trigger function Disabled.

		1 = External Event Trigger function Enabled.

EPWM Software Event Output Force Control Register (EPWMx_SWEOFCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_SWEOFCTL	EPWMx_BA+0x288	R/W	EPWM Software Event Output Force Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**OUTACTS5**		**OUTACTS4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OUTACTS3		**OUTACTS2**		**OUTACTS1**		**OUTACTS0**	

Bits	Description	
[31:12]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..5	**OUTACTSn**	**Output Action Selection** 00 = Do nothing. 01 = EPWM output Low. 10 = EPWM output High. 11 = EPWM output Toggle.

EPWM Software Event Output Force Trigger Register (EPWMx_SWEOFTRG)

Register	Offset	R/W	Description	Reset Value
EPWM_SWEOFTRG	EPWMx_BA+0x28C	R/W	EPWM Software Event Output Force Trigger Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**SWETRG5**	**SWETRG4**	**SWETRG3**	**SWETRG2**	**SWETRG1**	**SWETRG0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[n] n=0,1..5	**SWETRGn**	**Software Event Trigger** Write 1 to this bit will change EPWM output status according to OUTACTSn in EPWMx_SWEOFCTL setting. **Note:** This bit will auto cleared by hardware.

EPWM Clock Prescale Register 0~5 (EPWM_CLKPSC0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CLKPSC0	EPWMx_BA+0x290	R/W	EPWM Clock Prescale Register 0	0x0000_0000
EPWM_CLKPSC1	EPWMx_BA+0x294	R/W	EPWM Clock Prescale Register 1	0x0000_0000
EPWM_CLKPSC2	EPWMx_BA+0x298	R/W	EPWM Clock Prescale Register 2	0x0000_0000
EPWM_CLKPSC3	EPWMx_BA+0x29C	R/W	EPWM Clock Prescale Register 3	0x0000_0000
EPWM_CLKPSC4	EPWMx_BA+0x2A0	R/W	EPWM Clock Prescale Register 4	0x0000_0000
EPWM_CLKPSC5	EPWMx_BA+0x2A4	R/W	EPWM Clock Prescale Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CLKPSC**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLKPSC							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CLKPSC**	**EPWM Counter Clock Prescale** The clock of EPWM counter is decided by clock prescaler. Each EPWM pair shares one EPWM counter clock prescaler. The clock of EPWM counter is divided by (CLKPSC+ 1).

EPWM Rising Dead-time Counter Register 0_1, 2_3, 4_5 (EPWM_RDTCNT0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_RDTCNT0_1	EPWMx_BA+0x2A8	R/W	EPWM Rising Dead-time Counter Register 0/1	0x0000_0000
EPWM_RDTCNT2_3	EPWMx_BA+0x2AC	R/W	EPWM Rising Dead-time Counter Register 2/3	0x0000_0000
EPWM_RDTCNT4_5	EPWMx_BA+0x2B0	R/W	EPWM Rising Dead-time Counter Register 4/5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				RDTCNT			
7	6	5	4	3	2	1	0
RDTCNT							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**RDTCNT**	**Rising Dead-time Counter (Write Protect)** The Rising dead-time can be calculated from the following formula: RDTCKSEL=0: Rising Dead-time = (RDTCNT[11:0]+1) * EPWM_CLK period. RDTCKSEL=1: Rising Dead-time = (RDTCNT[11:0]+1) * EPWM_CLK period * (CLKPSC+1). **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Falling Dead-time Counter Register 0_1, 2_3, 4_5 (EPWM_FDTCNT0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_FDTCNT0_1	EPWMx_BA+0x2B4	R/W	EPWM Falling Dead-time Counter Register 0/1	0x0000_0000
EPWM_FDTCNT2_3	EPWMx_BA+0x2B8	R/W	EPWM Falling Dead-time Counter Register 2/3	0x0000_0000
EPWM_FDTCNT4_5	EPWMx_BA+0x2BC	R/W	EPWM Falling Dead-time Counter Register 4/5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				FDTCNT			
7	6	5	4	3	2	1	0
FDTCNT							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**FDTCNT**	**Falling Dead-time Counter (Write Protect)** The dead-time can be calculated from the following formula: FDTCKSEL=0: Falling Dead-time = (FDTCNT[11:0]+1) * EPWM_CLK period. FDTCKSEL=1: Falling Dead-time = (FDTCNT[11:0]+1) * EPWM_CLK period * (CLKPSC+1). **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Dead-time Control Register (EPWM_DTCTL)

Register	Offset	R/W	Description	Reset Value
EPWM_DTCTL	EPWMx_BA+0x2C0	R/W	EPWM Dead-time Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					DTCKSEL4	DTCKSEL2	DTCKSEL0
15	14	13	12	11	10	9	8
Reserved					FDTEN4	FDTEN2	FDTEN0
7	6	5	4	3	2	1	0
Reserved					RDTEN4	RDTEN2	RDTEN0

Bits	Description	
[31:17]	**Reserved**	Reserved.
[n/2+16]	**DTCKSELn**	**Dead-time Clock Select for EPWM Pair (Write Protect)** 0 = Dead-time clock source from EPWM_CLK. 1 = Dead-time clock source from prescaler output. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[15:11]	**Reserved**	Reserved.
[n/2+8] n=0,2,4	**FDTENn**	**Enable Falling Dead-time Insertion for EPWM Pair (Write Protect)** Falling Dead-time insertion is only active when this pair of complementary EPWM is enabled. If falling dead- time insertion is inactive, the outputs of pin pair are complementary without any delay. 0 = Falling Dead-time insertion Disabled on the pin pair. 1 = Falling Dead-time insertion Enabled on the pin pair. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.
[7:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**RDTENn**	**Enable Rising Dead-time Insertion for EPWM Pair (Write Protect)** Rising Dead-time insertion is only active when this pair of complementary EPWM is enabled. If rising dead- time insertion is inactive, the outputs of pin pair are complementary without any delay. 0 = Rising Dead-time insertion Disabled on the pin pair. 1 = Rising Dead-time insertion Enabled on the pin pair. **Note:** This bit is write protected. Refer to SYS_REGLCTL register.

EPWM Period Register Buffer 0~5 (EPWM_PBUF0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_PBUF0	EPWMx_BA+0x304	R	EPWM PERIOD0 Buffer	0x0000_0000
EPWM_PBUF1	EPWMx_BA+0x308	R	EPWM PERIOD1 Buffer	0x0000_0000
EPWM_PBUF2	EPWMx_BA+0x30C	R	EPWM PERIOD2 Buffer	0x0000_0000
EPWM_PBUF3	EPWMx_BA+0x310	R	EPWM PERIOD3 Buffer	0x0000_0000
EPWM_PBUF4	EPWMx_BA+0x314	R	EPWM PERIOD4 Buffer	0x0000_0000
EPWM_PBUF5	EPWMx_BA+0x318	R	EPWM PERIOD5 Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PBUF**	**EPWM Period Register Buffer (Read Only)** Used as PERIOD active register.

EPWM Comparator Register Buffer0~5 (EPWM_CMPBUF0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CMPBUF0	EPWMx_BA+0x31C	R	EPWM CMPDAT0 Buffer	0x0000_0000
EPWM_CMPBUF1	EPWMx_BA+0x320	R	EPWM CMPDAT1 Buffer	0x0000_0000
EPWM_CMPBUF2	EPWMx_BA+0x324	R	EPWM CMPDAT2 Buffer	0x0000_0000
EPWM_CMPBUF3	EPWMx_BA+0x328	R	EPWM CMPDAT3 Buffer	0x0000_0000
EPWM_CMPBUF4	EPWMx_BA+0x32C	R	EPWM CMPDAT4 Buffer	0x0000_0000
EPWM_CMPBUF5	EPWMx_BA+0x330	R	EPWM CMPDAT5 Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPBUF**	**EPWM Comparator Register Buffer (Read Only)** Used as CMP active register.

EPWM_FTCMPDAT Buffer (EPWM_FTCBUF0_1,2_3,4_5)

Register	Offset	R/W	Description	Reset Value
EPWM_FTCBUF0_1	EPWMx_BA+0x340	R	EPWM FTCMPDAT0_1 Buffer	0x0000_0000
EPWM_FTCBUF2_3	EPWMx_BA+0x344	R	EPWM FTCMPDAT2_3 Buffer	0x0000_0000
EPWM_FTCBUF4_5	EPWMx_BA+0x348	R	EPWM FTCMPDAT4_5 Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FTCMPBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FTCMPBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FTCMPBUF**	**EPWM FTCMPDAT Buffer (Read Only)** Used as FTCMP active buffer.

EPWM FTCMPDAT Indicator Register (EPWM_FTCI)

Register	Offset	R/W	Description	Reset Value
EPWM_FTCI	EPWMx_BA+0x34C	R/W	EPWM FTCMPDAT Indicator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**FTCMD4**	**FTCMD2**	**FTCMD0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**FTCMU4**	**FTCMU2**	**FTCMU0**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[8+n/2] n=0,2,4	**FTCMDn**	**EPWM FTCMPDAT Down Indicator** Indicator is set by hardware when EPWM counter down counts and reaches EPWM_FTCMPDATn. Software can clear this bit by writing 1 to it.
[7:3]	**Reserved**	Reserved.
[n/2] n=0,2,4	**FTCMUn**	**EPWM FTCMPDAT Up Indicator** Indicator is set by hardware when EPWM counter up counts and reaches EPWM_FTCMPDATn. Software can clear this bit by writing 1 to it.

EPWM CLKPSC Buffer 0~5 (EPWM_CPSCBUF0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_CPSCBUF0	EPWMx_BA+0x350	R	EPWM CLKPSC0 Buffer	0x0000_0000
EPWM_CPSCBUF1	EPWMx_BA+0x354	R	EPWM CLKPSC1 Buffer	0x0000_0000
EPWM_CPSCBUF2	EPWMx_BA+0x358	R	EPWM CLKPSC2 Buffer	0x0000_0000
EPWM_CPSCBUF3	EPWMx_BA+0x35C	R	EPWM CLKPSC3 Buffer	0x0000_0000
EPWM_CPSCBUF4	EPWMx_BA+0x360	R	EPWM CLKPSC4 Buffer	0x0000_0000
EPWM_CPSCBUF5	EPWMx_BA+0x364	R	EPWM CLKPSC5 Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CPSCBUF**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CPSCBUF							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CPSCBUF**	**EPWM Counter Clock Prescale Buffer** Used as EPWM counter clock pre-scare active register.

EPWM Interrupt Flag Accumulator Counter0~5 (EPWM_IFACNT0~5)

Register	Offset	R/W	Description	Reset Value
EPWM_IFACNT0	EPWMx_BA+0x368	R	EPWM Interrupt Flag Accumulator Counter 0	0x0000_0000
EPWM_IFACNT1	EPWMx_BA+0x36C	R	EPWM Interrupt Flag Accumulator Counter 1	0x0000_0000
EPWM_IFACNT2	EPWMx_BA+0x370	R	EPWM Interrupt Flag Accumulator Counter 2	0x0000_0000
EPWM_IFACNT3	EPWMx_BA+0x374	R	EPWM Interrupt Flag Accumulator Counter 3	0x0000_0000
EPWM_IFACNT4	EPWMx_BA+0x378	R	EPWM Interrupt Flag Accumulator Counter 4	0x0000_0000
EPWM_IFACNT5	EPWMx_BA+0x37C	R	EPWM Interrupt Flag Accumulator Counter 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ACUCNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACUCNT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**ACUCNT**	**Accumulator Counter (Read Only)** This value indicates how many interrupt are accumulated when using interrupt flag accumulator function.

6.13 Enhanced Quadrature Encoder Interface (EQEI)

6.13.1 Overview

There are four EQEI controllers in this device. The Enhanced Quadrature Encoder Interface (EQEI) decodes speed of rotation and motion sensor information. It can be used in any application that uses a quadrature encoder for feedback.

6.13.2 Features

6.13.2.1 Enhanced Quadrature Encoder Interface (EQEI) Features

- Up to four EQEI controllers, EQEI0, EQEI1, EQEI2 and EQEI3.
- Two EQEI phase inputs, QEA and QEB; One Index input.
- A 32-bit up/down Quadrature Encoder Pulse Counter (EQEI_CNT)
- A 32-bit software-latch Quadrature Encoder Pulse Counter Hold Register (EQEI_CNTHOLD)
- A 32-bit Quadrature Encoder Pulse Counter Index Latch Register (EQEI_CNTLATCH)
- A 32-bit Quadrature Encoder Pulse Counter Compare Register (EQEI_CNTCMP) with a Pre-set Maximum Count Register (EQEI_CNTMAX)
- A 32-bit up count Unit Timer Pulse Counter (EQEI_UTCNT)
- A 32-bit Unit Timer Pulse Counter Compare Register (EQEI_UTCMP)
- One EQEI control register (EQEI_CTL) and one EQEI Status Register (EQEI_STATUS)
- Four Quadrature encoder pulse counter operation modes:
  - Supports X4 free-counting mode
  - Supports X2 free-counting mode
  - Supports X4 compare-counting mode
  - Supports X2 compare-counting mode
- Two Quadrature encoder phase counter operation modes:
  - Supports X1 1-phase 2 input compare-counting mode
  - Supports X2 1-phase 2 input compare-counting mode
- Two Quadrature encoder directional counter operation modes:
  - Supports X1 1-phase 1 input compare-counting mode
  - Supports X2 1-phase 1 input compare-counting mode
- Supports swap function for input signals QEA and QEB
- Supports for detecting the occurrence of phase error from input signals QEA and QEB
- Supports one times index signal reset function for Quadrature encoder pulse counter
- Encoder Pulse Width measurement mode
- Input frequency of QEA/QEB/IDX without noise filter must lower than PCLK/4
- Input frequency of QEA/QEB/IDX with noise filter must lower than Noise Filter Clock/8

6.13.3 Block Diagram

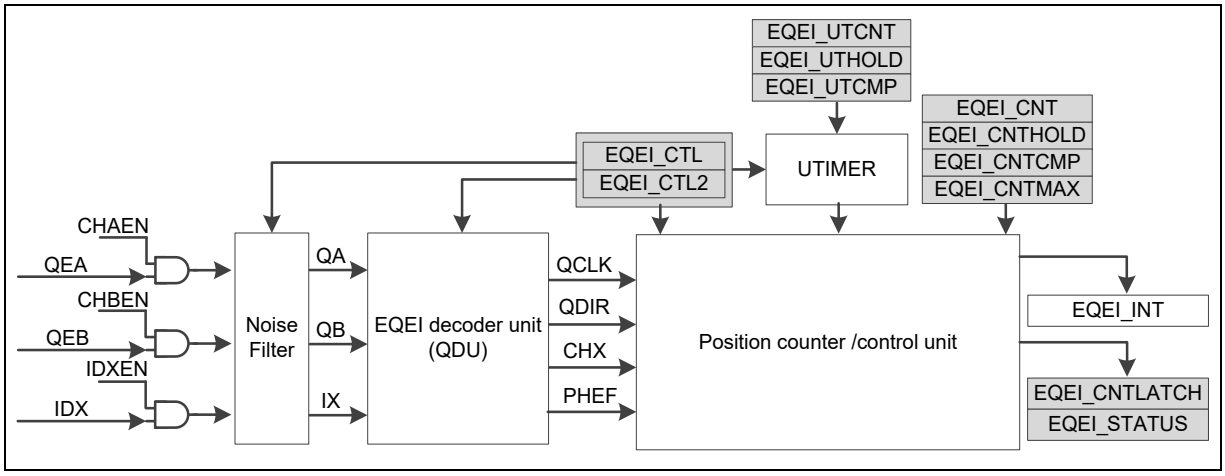


Figure 6.13-1 EQEI Block Diagram

The EQEI controller inputs, QEA and QEB, accept the outputs from a quadrature encoded source, such as incremental optical shaft encoder. Two channels, A and B, nominally 90 degrees out of phase, are required. A quadrature encoder usually provides an index signal (to pin IDX) which can be used to indicate an absolute position. There is a noise filter and polarity control for each signal before EQEI control unit.

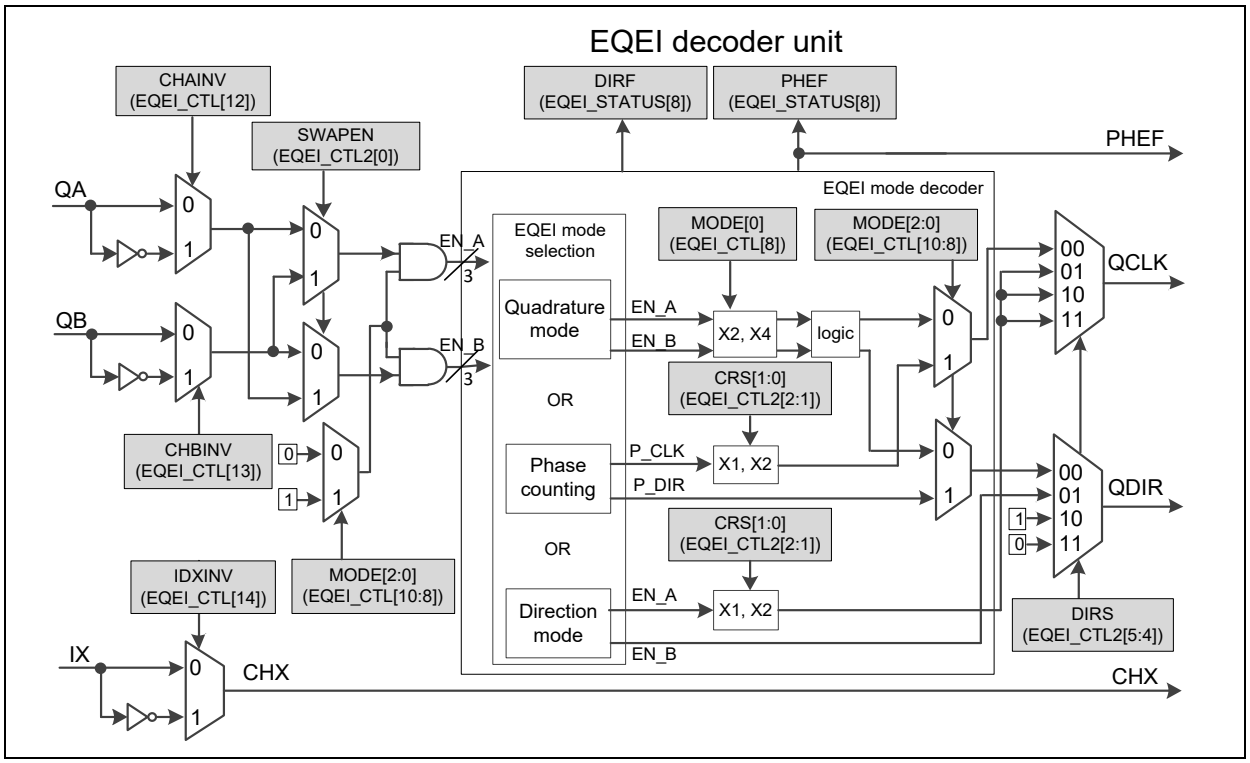


Figure 6.13-2 EQEI Decoder Unit

EQEI Decoder Unit (QDU) module contains inverse and swap function for input signals QA and QB, EQEI mode selection, EQEI phase counter clock rate selection, and the mechanism for detecting phase error. Users can use the function they want by controlling the registers.

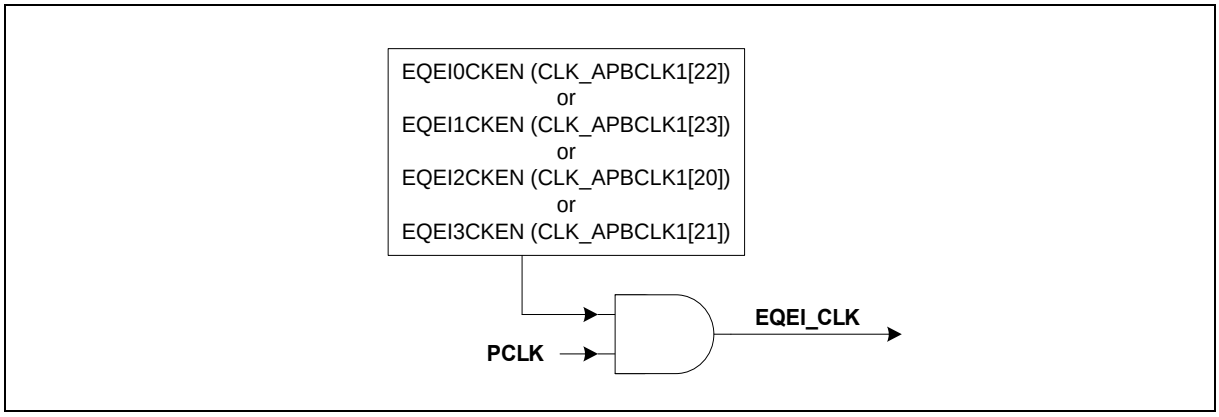


Figure 6.13-3 EQEI Clock Source Control

6.13.4 Basic Configuration

6.13.4.1 EQEI0 Basic Configuration

- Clock Source Configuration
  - Enable EQEI0 peripheral clock in EQEI0CKEN (CLK_APBCLK1[22]).
- Reset Configuration
  - Reset EQEI0 controller in EQEI0RST (SYS_IPRST2[22]).

6.13.4.2 EQEI1 Basic Configuration

- Clock Source Configuration
  - Enable EQEI1 peripheral clock in EQEI1CKEN (CLK_APBCLK1[23]).
- Reset Configuration
  - Reset EQEI1 controller in EQEI1RST (SYS_IPRST2[23]).

6.13.4.3 EQEI2 Basic Configuration

- Clock Source Configuration
  - Enable EQEI2 peripheral clock in EQEI2CKEN (CLK_APBCLK1[20]).
- Reset Configuration
  - Reset EQEI2 controller in EQEI2RST (SYS_IPRST2[20]).

6.13.4.4 EQEI3 Basic Configuration

- Clock Source Configuration
  - Enable EQEI3 peripheral clock in EQEI3CKEN (CLK_APBCLK1[21]).
- Reset Configuration
  - Reset EQEI3 controller in EQEI3RST (SYS_IPRST2[21]).

6.13.5 Functional Description

The EQEI control logic detects the relation of phase lead/lag between the filtered signals CHA and CHB and CHX to produce direction indication bit DIRF(EQEI_STATUS[8]) to control the pulse counter. The comparator/reload logic compares the pulse counter and maximum count and control the function of reloading pulse counter in compare-counting mode. In Free-counting mode the pulse counter CNT(EQEI_CNT[31:0]) will count until the 0xFFFF_FFFF value; while in Compare-counting mode the pulse counter will counts until the CNTMAX (EQEI_CNTMAX[31:0]) and the pulse counter will be reset to 0 to restart the next cyclic counting.

6.13.5.1 Input Noise Filter

Each pin of EQEI inputs is equipped with a noise filter which can filter the unwanted noise form. The QEA, QEB and IDX noise filters can be disabled through bits NFDIS (EQEI_CTL[3]). If enabled, the capture logic is required to sample 4 consecutive same capture input value in order to recognize an edge as a capture event. In Figure 6.13-6, it is a possible implementation of digital noise filter; the interval between pulses requirement for input capture is 4 EQEI_CLK clocks width. Any pulse width less than or equal to 3 EQEI_CLK clocks will not have any trigger. The timing requirement through Noise Filter is also shown in Figure 6.13-6. CHA, CHB and CHX are the outputs of QEA, QEB and IDX respectively after going through noise filter and polarity control. Refer to Figure 6.13-4. If the noise filter is disabled the input signals QEA, QEB and IDX are passed to the internal signals CHA, CHB and CHX respectively without any delay.

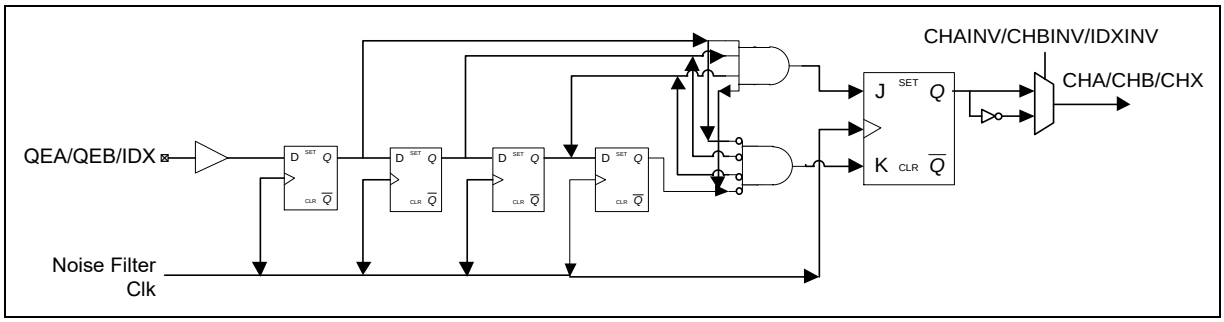


Figure 6.13-4 Noise Filter

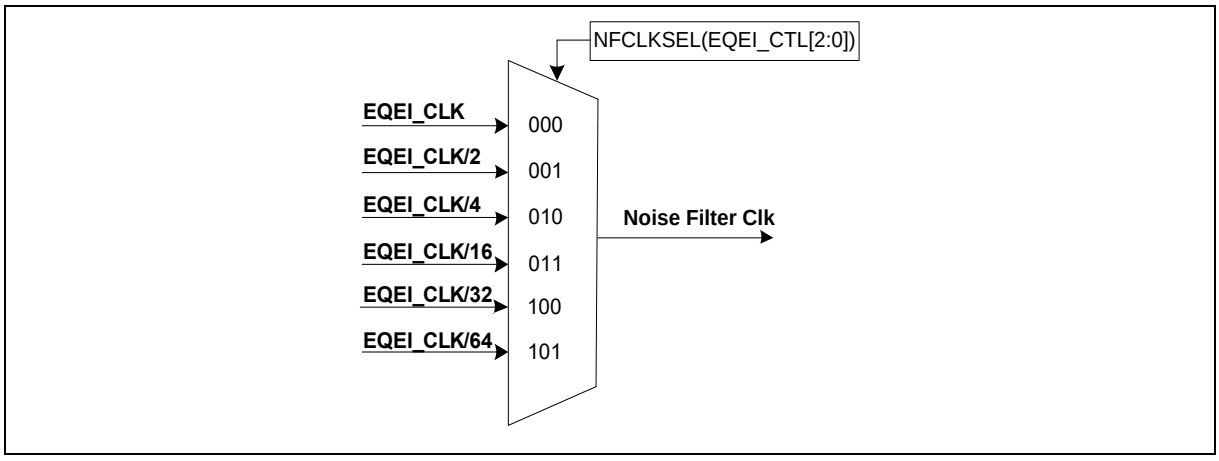


Figure 6.13-5 Noise Filter Sampling Clock Selection

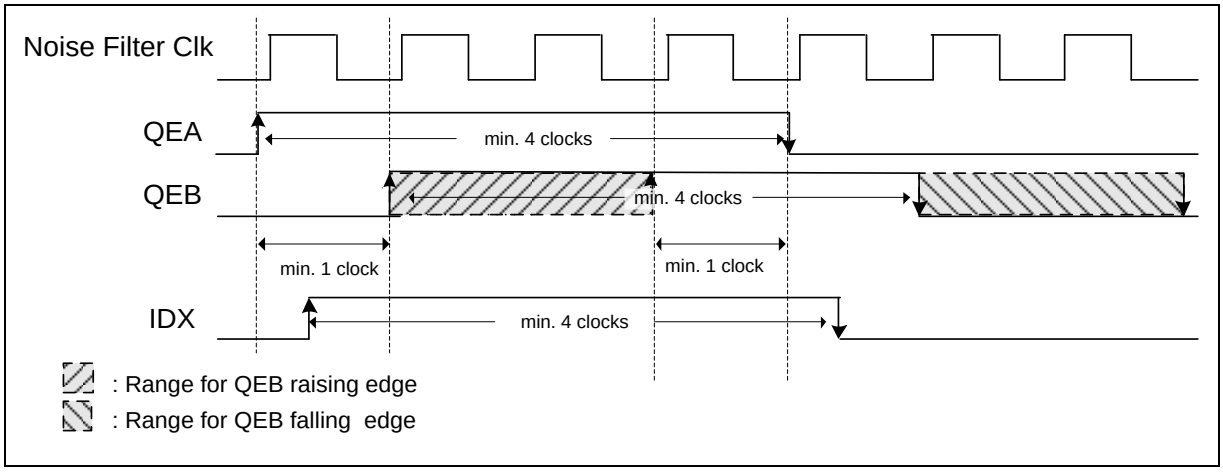


Figure 6.13-6 QEA/QEB/IDX Timing Requirement Through Noise Filter

6.13.5.2 Operation of Quadrature Encoder Interface

There are seven Quadrature encoder pulse counter operation modes:

- Mode0: X4 Quadrature free-counting mode
- Mode1: X2 Quadrature free-counting mode
- Mode2: X4 Quadrature compare-counting mode
- Mode3: X2 Quadrature compare-counting mode
- Mode4: 1-phase 2 input phase counting mode type 1
- Mode5: 1-phase 2 input phase counting mode type 2
- Mode6: directional counting mode.

6.13.5.3 Free-counting Mode

The quadrature encoder pulse counter CNT(EQEI_CNT[31:0]) up or down counts according to the direction indication bit DIRF (EQEI_STATUS[8]). When overflow or underflow occurs, it sets flag OVUNF (EQEI_STATUS[2]). Refer to Figure 6.13-7 and Figure 6.13-8 for detailed timing.

6.13.5.4 Compare-counting Mode

The pulse counter up or down counts according to the direction indication bit DIRF (EQEI_STATUS[8]). On up counting, flag OVUNF (EQEI_STATUS[2]) will be asserted when CNT(EQEI_CNT[31:0]) overflows from CNTMAX (EQEI_CNTMAX[31:0]) to 0 on the next CHA edge for X2 counting mode, and on CHA/CHB edge for X4 counting mode. On down counting, flag OVUNF (EQEI_STATUS[2]) will be asserted when CNT(EQEI_CNT[31:0]) underflows from 0 to CNTMAX (EQEI_CNTMAX[31:0]) on the next CHA edge for X2 counting mode, and on CHA/CHB edge for X4 counting mode. This mode provides the position of a rotor to user. If a quadrature encoder output 1024 pulses to CHA per round, user can write EQEI_MAXCNT and CNTCMP(EQEI_CNTCMP[31:0]) with 4095 in X4 mode or 2047 in x2 mode and reset CNT(EQEI_CNT[31:0]) at initial before compare-counting mode is active. When the CNT(EQEI_CNT[31:0]) overflows from CNTCMP(EQEI_CNTCMP[31:0]), here CNTCMP(EQEI_CNTCMP[31:0]) should be preset the same value as CNTMAX (EQEI_CNTMAX[31:0]), it means rotor runs one round on next CHA/CHB edge. Refer to Figure 6.13-7 and Figure 6.13-8 for detailed timing.

6.13.5.5 X4/X2 Counting Modes

In **X4 Counting mode**, the pulse counter increases or decreases one on every CHA and CHB edge based on the phase relationship of CHA and CHB signals.

EQEI X4 Counting mode provides a finer resolution of the rotor position, since the counter increments or decrements more frequently for each QEA/QEB input pulse pair than in EQEI x2 mode. This mode is selected by setting the EQEI Counting Mode Selection bits MODE(EQEI_CTL[10:8]) to 000 or 010. In this mode, the EQEI logic detects every edge on every QEA and QEB input edges.

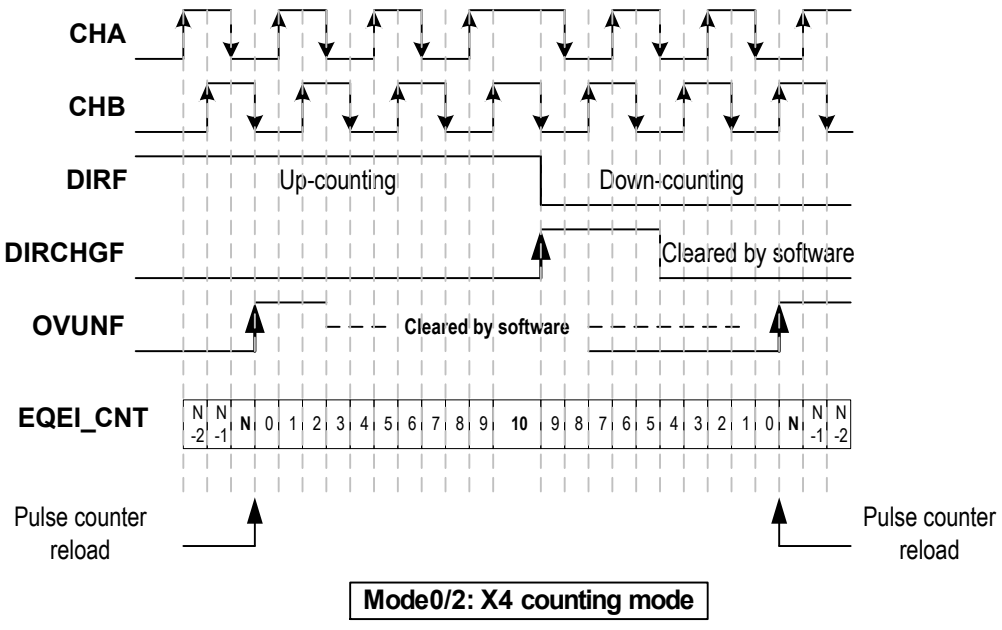
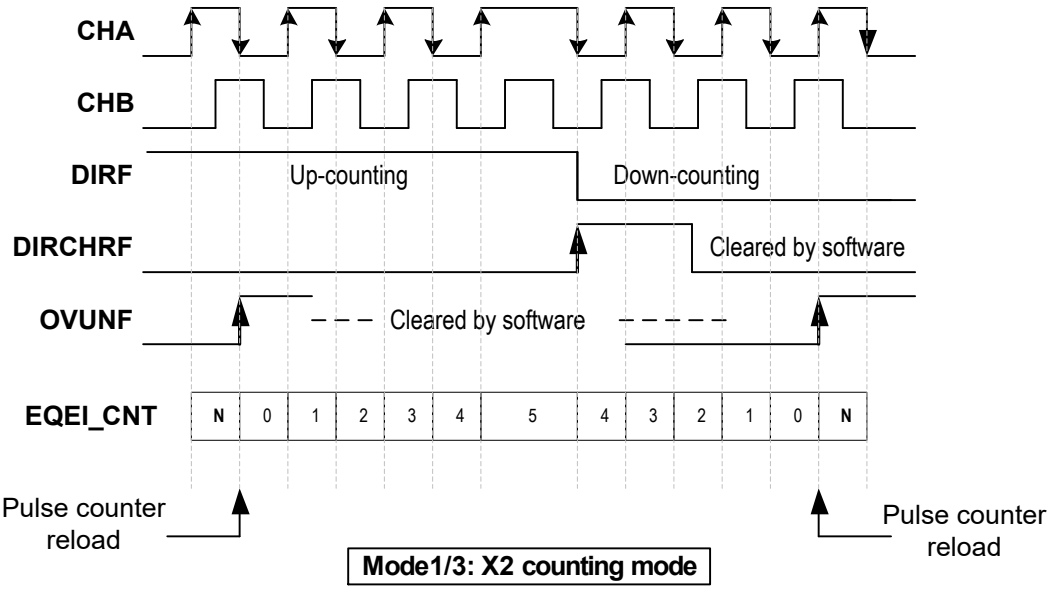


Figure 6.13-7 X4 Counting Mode

In **X2 Counting mode**, the pulse counter increases or decreases one on every CHA edge based on the phase relationship of CHA and CHB signals.

EQEI X2 Counting mode is selected by setting the EQEI Counting Mode Selection bits (EQEI_CTL[10:8]) to 001 or 011. In this mode, the EQEI logic detects every edge on the QEA input only. Every rising and falling edge on the QEA signal clocks the pulse counter.



Note:
Mode1: X2 free-counting mode, N=0xFFFF_FFFF
Mode3: X2 compare-counting mode, N=CNTMAX
CHA & CHB are output signals of digital filter

Figure 6.13-8 X2 Counting Mode

6.13.5.6 Direction of Counting

If CHA leads CHB, the pulse counter is increased by 1. If CHA lags CHB, the pulse counter is decreased by 1. The EQEI control logic generates a signal that sets the DIRF (EQEI_STATUS[8]); this in turn determines the direction of the count. When CHA leads CHB, DIRF(EQEI_STATUS[8]) is set as 1, and the position counter increments on every active edge. When CHA lags CHB, DIRF(EQEI_STATUS[8]) is cleared, and the position counter decrements on every active edge. Refer to Table 6.13-1.

Current Signal Detected	Previous Signal Detected				DIR (Counting Direction)
	Rising		Falling		
	CHA	CHB	CHA	CHB	
CHA rising				√	1 (Increment)
		√			0 (Decrement)
			√		Toggle (direction change)
CHA falling				√	0 (Decrement)
		√			1 (Increment)
	√				Toggle (direction change)
CHB rising	√				1 (Increment)
			√		0 (Decrement)
				√	Toggle (direction change)
CHB falling			√		1 (Increment)

	√				0 (Decrement)
		√			Toggle (direction change)

Table 6.13-1 Direction of Counting

6.13.5.7 Up-Counting

Under the forward direction the DIRF(EQEI_STATUS[8]) bit is 1 when up-counting. Software needs to clear the OVUNF (EQEI_STATUS[2]) flag. For the free-counting mode the CNT(EQEI_CNT[31:0]) counter will count until it matches 0xFFFF_FFFF and next edges on the forward direction will set the bit OVUNF (EQEI_STATUS[2]) high and reset CNT(EQEI_CNT[31:0]) to 0. For compare-counting mode the CNT(EQEI_CNT[31:0]) counter counts until the CNTMAX (EQEI_CNTMAX[31:0]) value and next edges on the forward direction will set the bit OVUNF (EQEI_STATUS[2]) high and reset CNT(EQEI_CNT[31:0]) to 0. Changes of direction trigger a down-count and CNT(EQEI_CNT[31:0]) decreasing in counter value. For X2 mode, only CHA edge will set OVUNF (EQEI_STATUS[2]) while for X4 mode both CHA and CHB edges will set OVUNF (EQEI_STATUS[2]).

6.13.5.8 Down-Counting

A change of direction will cause the counter to down count for X2/X4 counting mode. It is indicated with the DIRF(EQEI_STATUS[8]) bit as 0 and DIRCHGF (EQEI_STATUS[3]) flag is set to 1. At this stage the CNT(EQEI_CNT[31:0]) will start to down-count. In free-counting mode the pulse counter will reload with 0xFFFF_FFFF when it down counts to 0 and sets OVUNF (EQEI_STATUS[2]) to high in the next edge. The pulse counter will reload with CNTMAX (EQEI_CNTMAX[31:0]) when it down counts to 0 in compare-counting mode and sets OVUNF (EQEI_STATUS[2]) to high in the next edge. For X2 mode, only CHA edge will set OVUNF (EQEI_STATUS[2]) while for X4 mode both CHA and CHB edges will set OVUNF (EQEI_STATUS[2]).

6.13.5.9 Compare Function

The compare function in EQEI controller is to compare the dynamic counting CNT(EQEI_CNT[31:0]) with the compare register CNTCMP(EQEI_CNTCMP[31:0]). When CNT(EQEI_CNT[31:0]) up or down counts and reaches CNTCMP(EQEI_CNTCMP[31:0]), the flag CMPF(EQEI_STATUS[1]) will be set. Set bit CMP_EN (EQEI_CTL[28]) to one to enable the compare function otherwise disable it.

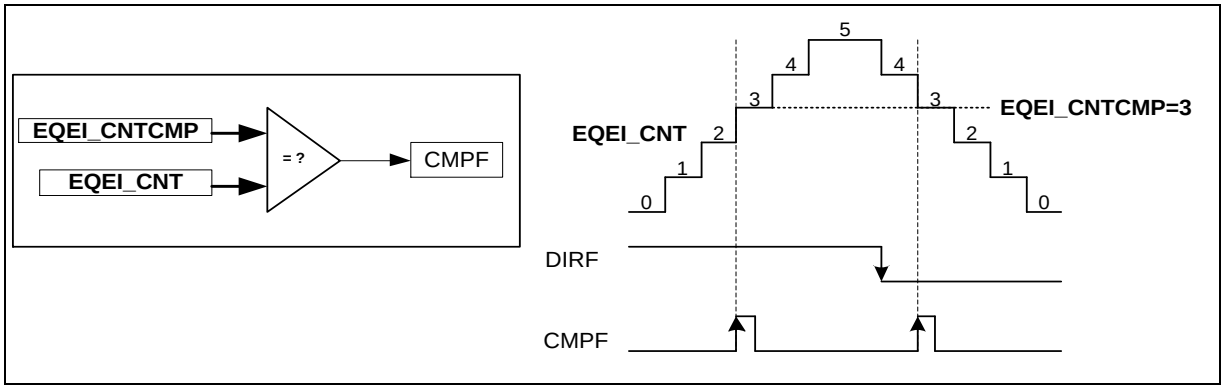


Figure 6.13-9 Compare Operation

6.13.5.10 1-phase 2 Input Phase Counting Mode

There are two different kinds of 1-phase 2 input phase counting modes in EQEI mode. The mode1 is used for counting pulse signals when the initial input signals are level high, and the other one, mode2, is used for counting level low signals. In phase counting mode, user can select clock rate X1 or X2 to trigger pulse counter by setting the EQEI Swap Function Clock Rate bits (EQEI_CTL2[2:1]).

Phase counting mode, initial high (mode1)

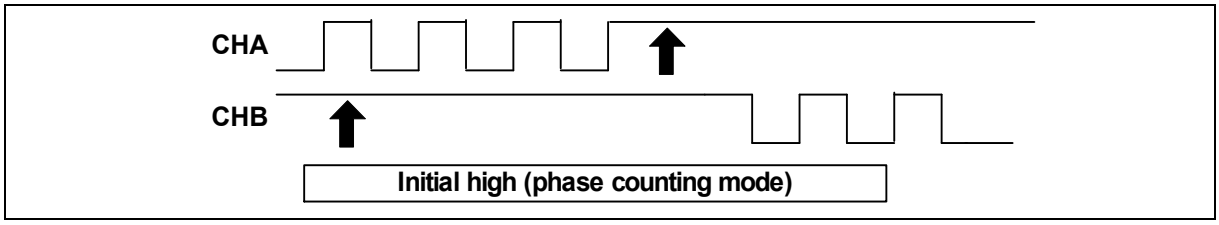


Figure 6.13-10 Phase Counting Mode, Mode1

Phase counting mode, initial low (mode2)

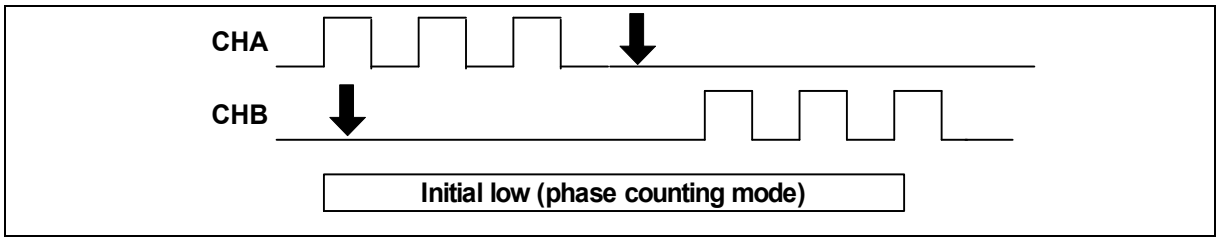


Figure 6.13-11 Phase Counting Mode, Mode2

X1/X2, 1-phase 2 input phase counting mode

In **X1 phase counting mode**, the pulse counter increases one on every CHA rising or falling edge and decrease one on every CHB rising or falling edge. User can select rising edge or falling edge to trigger pulse counter by setting the EQEI Clock Rate Setting without Quadrature Mode bits (EQEI_CTL2[2:1]) to 00 or 01. In this mode, every rising or falling edge on the QEA and QEB signal clocks the pulse counter.

In **X2 phase counting mode**, the pulse counter increases one on every CHA rising and falling edge and decreases one on every CHB rising and falling edge. User can trigger pulse counter with both rising and falling edge by setting the EQEI Clock Rate Setting without Quadrature Mode bits (EQEI_CTL2[2:1]) to 10. In this mode, every rising and falling edge on the QEA and QEB signal clocks the pulse counter.

X1 Phase counting mode, initial high (mode1), rising trigger

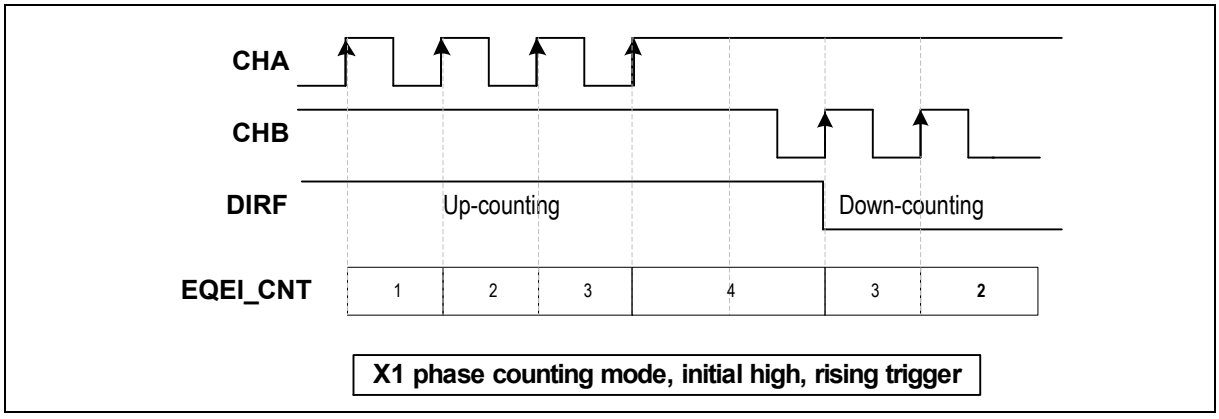


Figure 6.13-12 X1 Phase Counting Mode, Mode1, Rising Trigger

X1 Phase counting mode, initial high (mode1), falling trigger

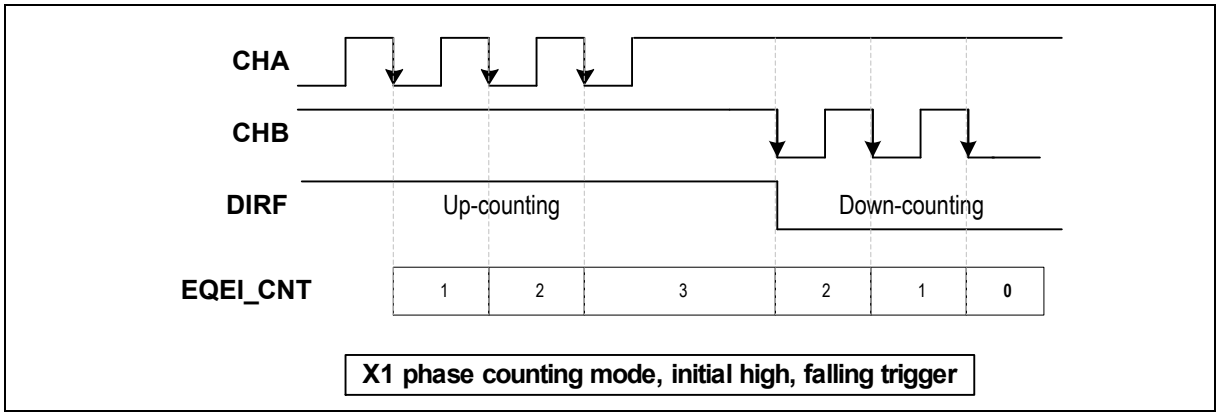


Figure 6.13-13 X1 Phase Counting Mode, Mode1, Falling Trigger

X2 Phase counting mode, initial high (mode1)

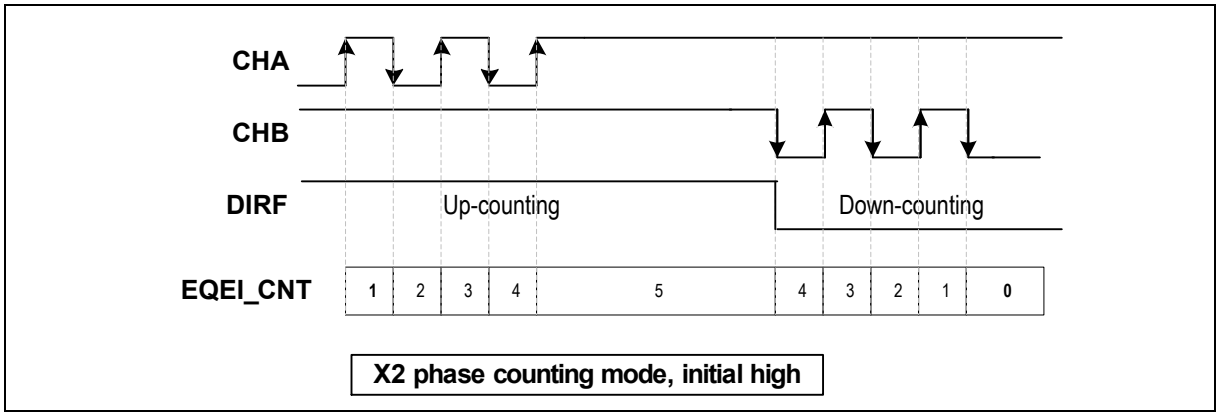


Figure 6.13-14 X2 Phase Counting Mode, Mode1

X1 Phase counting mode, initial low (mode2), rising trigger

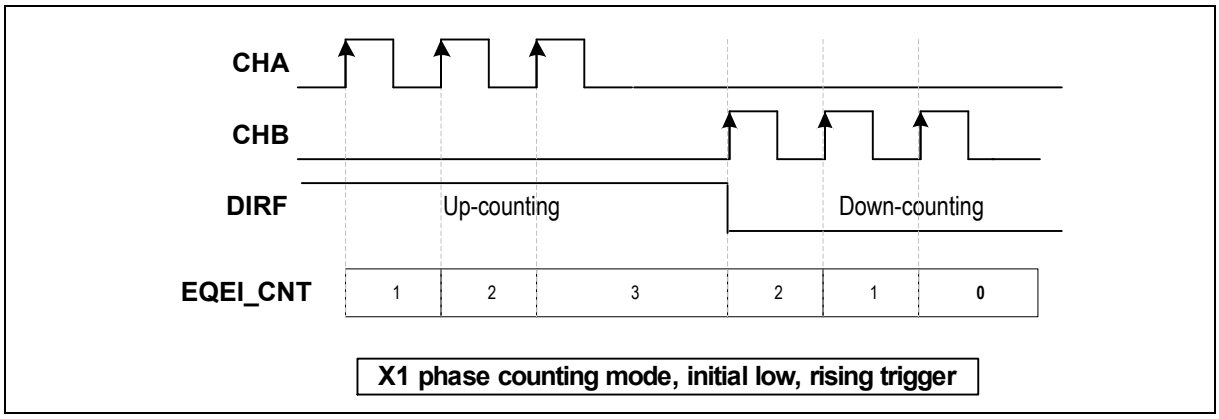


Figure 6.13-15 X1 Phase Counting Mode, Mode2, Rising Trigger

X1 Phase counting mode, initial low (mode2), falling trigger

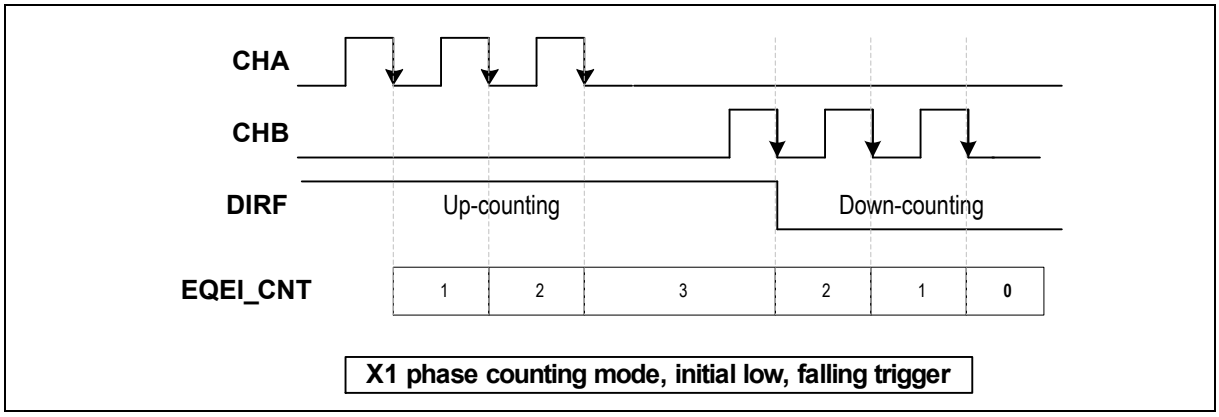


Figure 6.13-16 X1 Phase Counting Mode, Mode2, Falling Trigger

X2 Phase counting mode, initial low (mode2)

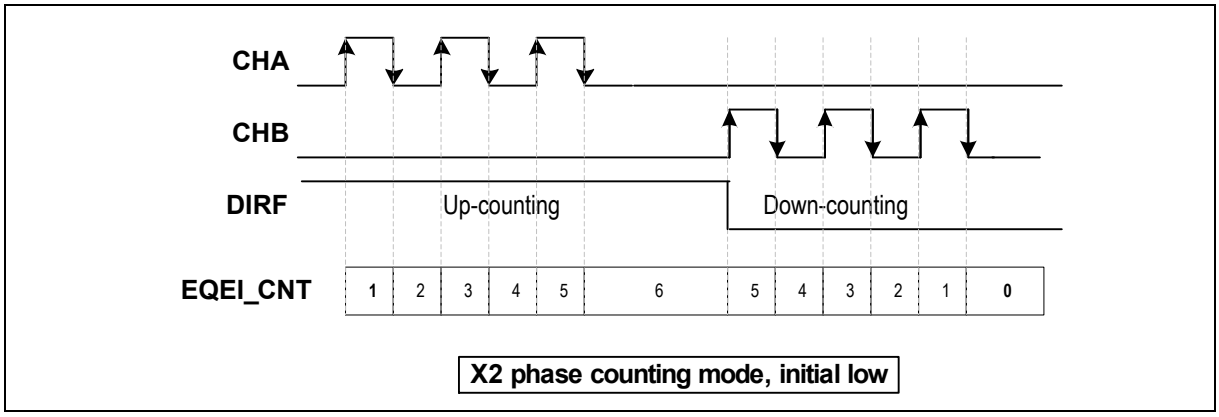


Figure 6.13-17 X2 Phase Counting Mode, Mode2

6.13.5.11 1-phase 1 input directional counting mode

There are three different kinds of 1-phase 1 input directional counting modes in EQEI mode. The first one is called “direction count mode”, which needs to input direction and clock signals from CHA and CHB. When user entered the signals into CHA and CHB, it will determine the pulse counter should add or subtract, and it also determine when the pulse counter should be driven. The second mode in 1-phase 1 input directional counting mode is called “direction up count mode”. In direction up count mode, user only needs to input clock to CHA and sets the register DIRSRC (EQEI_CTL2[5:4]) to 10. Then the pulse counter adds one automatically when the input clock triggers the counter. The last mode in EQEI mode is called “direction down count mode”. When user sets the register DIRSRC (EQEI_CTL2[5:4]) to 11 and input clock signal to CHA, then the pulse counter starts to substrate one by clock trigger automatically.

X1/X2, 1-phase 1 input directional counting mode

In **X1 directional counting mode**, the pulse counter increases or decreases one on every CHA rising or falling edge. User can select rising edge or falling edge to trigger pulse counter by setting the EQEI Clock Rate Setting without Quadrature Mode bits (EQEI_CTL2[2:1]) to 00 or 01. In this mode, every rising or falling edge on CHA signal clocks the pulse counter.

In **X2 directional counting mode**, the pulse counter increases or decreases one on every CHA rising and falling edge. User can trigger pulse counter with both rising and falling edge by setting the EQEI Clock Rate Setting without Quadrature Mode bits (EQEI_CTL2[2:1]) to 10. In this mode, every rising and falling edge on CHA signal clocks the pulse counter.

Direction count mode

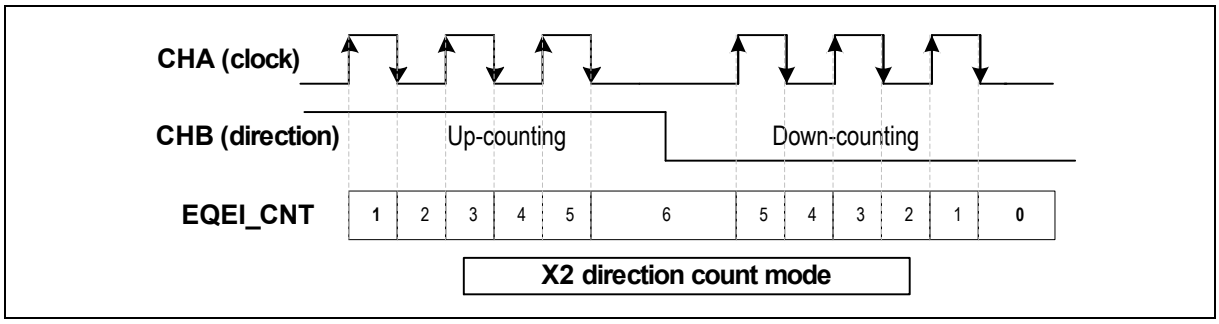


Figure 6.13-18 X2 Direction Count Mode

Direction up count mode

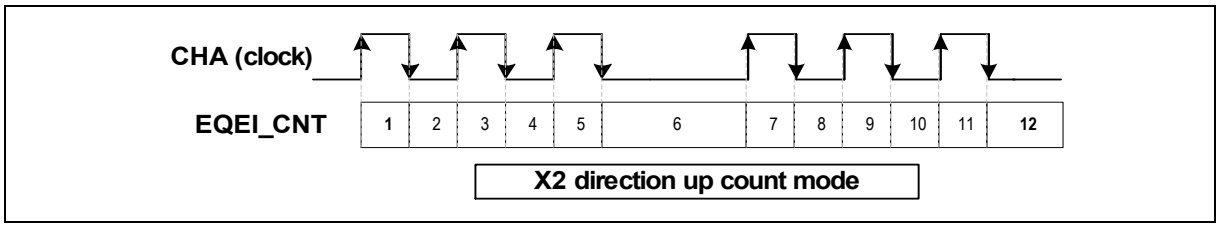


Figure 6.13-19 x2 Direction Up Count Mode

Direction down count mode

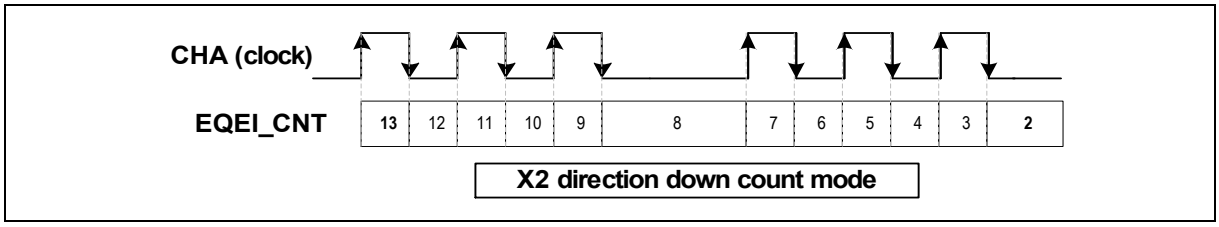


Figure 6.13-20 X2 Direction Down Count Mode

6.13.5.12 EQEI Swap Function for QA and QB

Swap function is a method to alter the destination of the input signals, which can help user to use EQEI mode more conveniently. User can shift QA and QB by controlling the register SWAPEN (EQEI_CTL2[0]) to 1. Normally the swap function is disabled if user did not set the register. Refer to Table 6.13-1.

MODE	QEA	QEB	SWAPEN (EQEI_CTL2[0])	DIRSRC (EQEI_CTL2[5:4])	QCLK	QDIR
Direction-count	CLK	DIR	0	00	CLK	DIR
Direction-count	DIR	CLK	1	00	CLK	DIR
Up count	CLK	x	0	10	CLK	1
Up count	x	CLK	1	10	CLK	1
Down count	CLK	x	0	11	CLK	0
Down count	x	CLK	1	11	CLK	0

MODE	QEA	QEB	SWAPEN (EQEI_CTL2[0])	DIRSRC (EQEI_CTL2[5:4])	QCLK	QDIR
Phase counting	A	B	0	00	CLK	DIR
Phase counting	B	A	1	00	CLK	DIR
MODE	QEA	QEB	SWAPEN (EQEI_CTL2[0])	DIRSRC (EQEI_CTL2[5:4])	QCLK	QDIR
Quadrature mode	A	B	0	00	CLK	DIR
Quadrature mode	A	B	1	00	CLK	DIR
Quadrature mode	B	A	0	00	CLK	DIR
Quadrature mode	B	A	1	00	CLK	DIR

Table 6.13-2 Swap Function Table

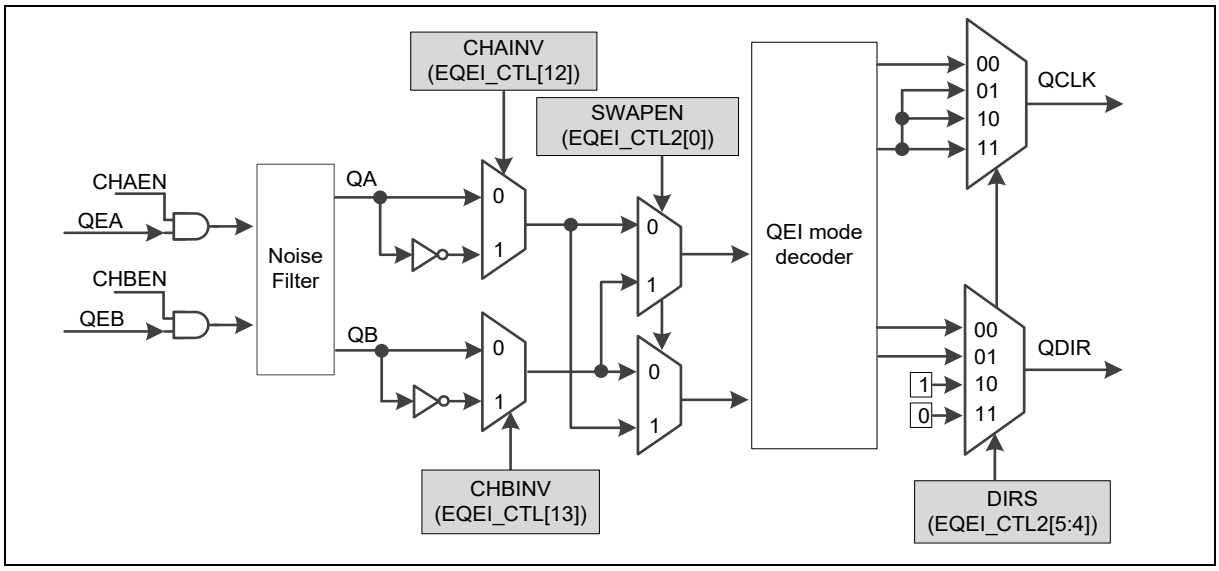


Figure 6.13-21 EQEI Swap Function for QA and QB

6.13.5.13 An unit timer counter in EQEI mode

There is a 32 bits UTCNT(EQEI_UTCNT) in EQEI mode which is triggered by clock. User can set the register UTEN(EQEI_CTL2[8]) to enable the unit timer counter to start counting. And user can also set the register UTHOLDEN(EQEI_CTL2[9]) to 1 to capture CNT(EQEI_CNT[31:0]) value and store into EQEI Counter Hold Register CNTHOLD(EQEI_CNTHOLD[31:0]) when the flag UTIEF(EQEI_STATUS[17]) is 1. If the unit timer counter value matches UTCMP(EQEI_UTCMP[31:0]), then the register UTIEF (EQEI_STATUS[17]) will be set. Unit timer can also help user to reset EQEI position counter when the register UTEVTRST(EQEI_CTL2[10]) is set to 1.There are three ways to initial unit timer value. The first one is written by software if UTEN (EQEI_CTL2[8]) = 0. The second one is the unit timer counter value matches UTCMP(EQEI_UTCMP[31:0]). And the last one is to set the register IDXRLDEN(EQEI_CTL[27]) =1 and IDXRSTUTS(EQEI_CTL2[11]=1, this would make the unit timer reset with EQEI_CNT when IDX reset event happens.

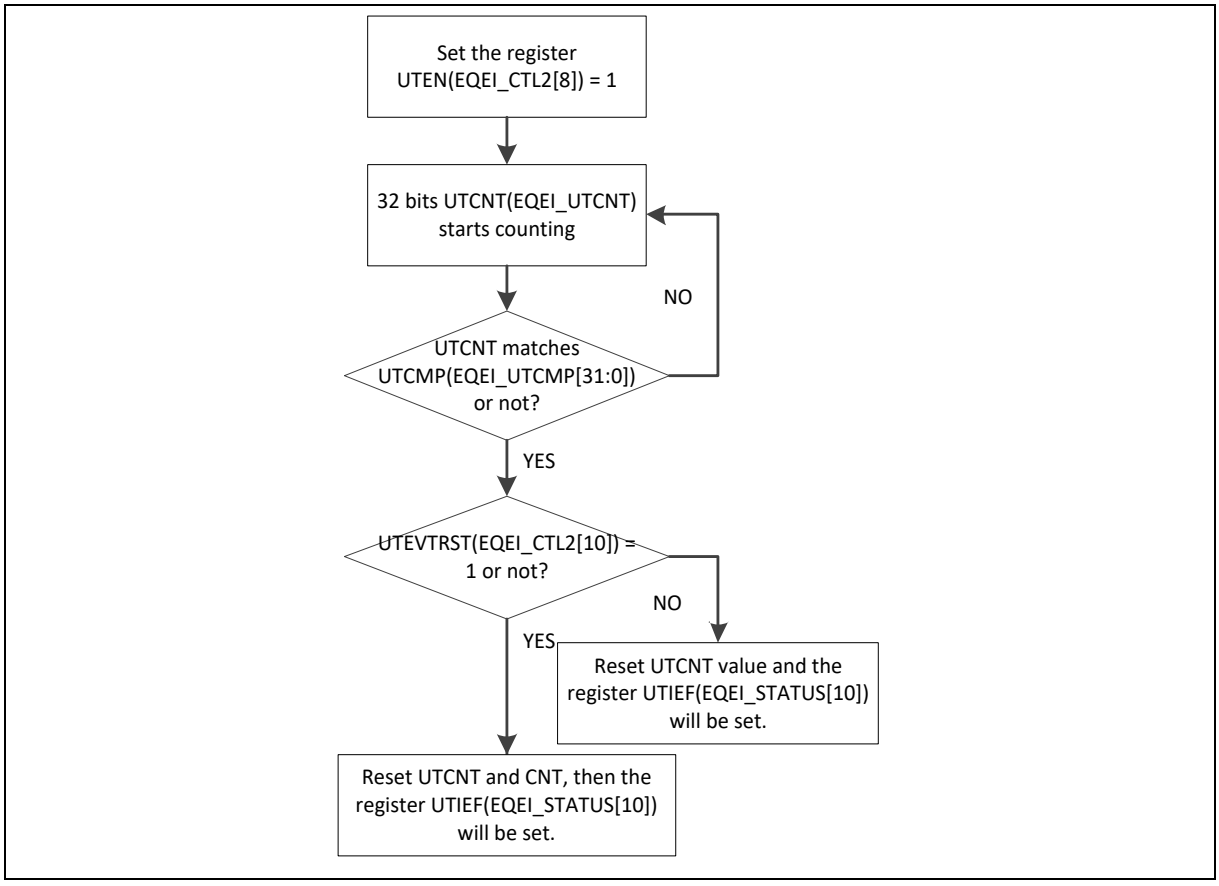


Figure 6.13-22 EQEI Unit Timer Counter Control Flow Chart

6.13.5.14 Reload Counter by Pin IDX

The CNT(EQEI_CNT[31:0]) counter can be reset to 0 or reload with the content of CNTMAX (EQEI_CNTMAX[31:0]) by the signal CHX (the filtered and polarity-set output of pin IDX) trigger. When the IDX Reload bit IDXRLD_EN(EQEI_CTL[27]) is set, a rising edge of CHX causes EQEI controller to reset the CNT(EQEI_CNT[31:0]) to 0 if the counter is in up-counting; if the counter is in down-counting the rising edge of CHX causes the EQEI controller to reload the CNT(EQEI_CNT[31:0]) with the content of CNTMAX (EQEI_CNTMAX[31:0]). Refer to Figure 6.13-23 for details.

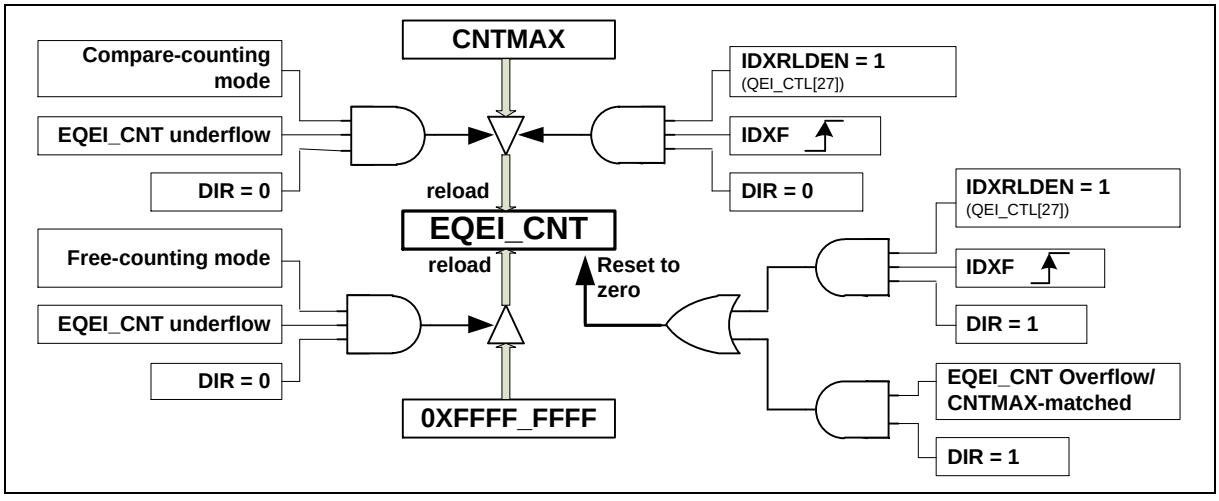


Figure 6.13-23 EQEI_CNT Reload/Reset Control

6.13.5.15 One Time Index Signal Reset Function

User can avoid CNT(EQEI_CNT[31:0]) being reset by selecting one time index signal reset function. To achieve this, user needs to set the register IDXRSTEN(EQEI_CTL[7]) and IDXRLDEN(EQEI_CTL[27]) to 1. Then CNT(EQEI_CNT[31:0]) will be reset when the first index signal coming. User should be aware that CNT(EQEI_CNT[31:0]) content will be latched into EQEI Counter Index Latch Register CNTLATCH (EQEI_CNTLATCH[31:0]) only at first rising edge of IDX signal, but UTCNT(EQEI_UTCNT[31:0]) is not subject to this restriction. User can select UTCNT(EQEI_UTCNT[31:0]) to be reset or not by setting IDXRSTUTS(EQEI_CTL[11]) register. If user wants to reset CNT(EQEI_CNT[31:0]) by next index signal when the first index signal resets has happened, user can set the register IDXRSTEV(EQEI_CTL[15]) to enable IDX to reset CNT(EQEI_CNT[31:0]) again, and monitor the flag FIDXEF(EQEI_STATUS[9]).

6.13.5.16 Capture EQEI Counter

If the bit HOLDCNT(EQEI_CTL[24]) or UTHOLDEN(EQEI_CTL2[9]) and UTIEF (EQEI_STATUS[17]) are set, the CNT(EQEI_CNT[31:0]) content will be captured into EQEI Counter Hold Register CNTHOLD(EQEI_CNTHOLD[31:0]), the data will be held until the next HOLDCNT (EQEI_CTL[24]) trigger comes. The bit HOLDCNT can be set by writing 1 to it or the rising edge of timers interrupt flags TIF (TIMERx_INTSTS[0]).

Note: The bit HOLDCNT is automatically cleared by hardware after CNTHOLD(EQEI_CNTHOLD[31:0]) captures the content of EQEI counter.

If the bit IDXLATEN (EQEI_CTL[25]) is set, the CNT(EQEI_CNT[31:0]) content will be latched into EQEI Counter Index Latch Register CNTLATCH (EQEI_CNTLATCH[31:0]) at every rising edge of CHX signal.

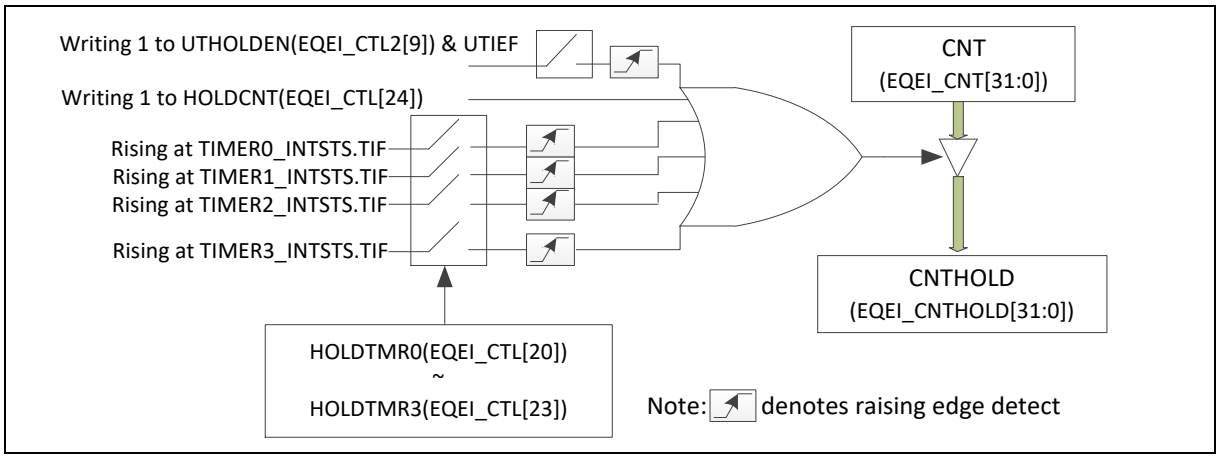


Figure 6.13-24 Trigger Control of Capturing EQEI Counter

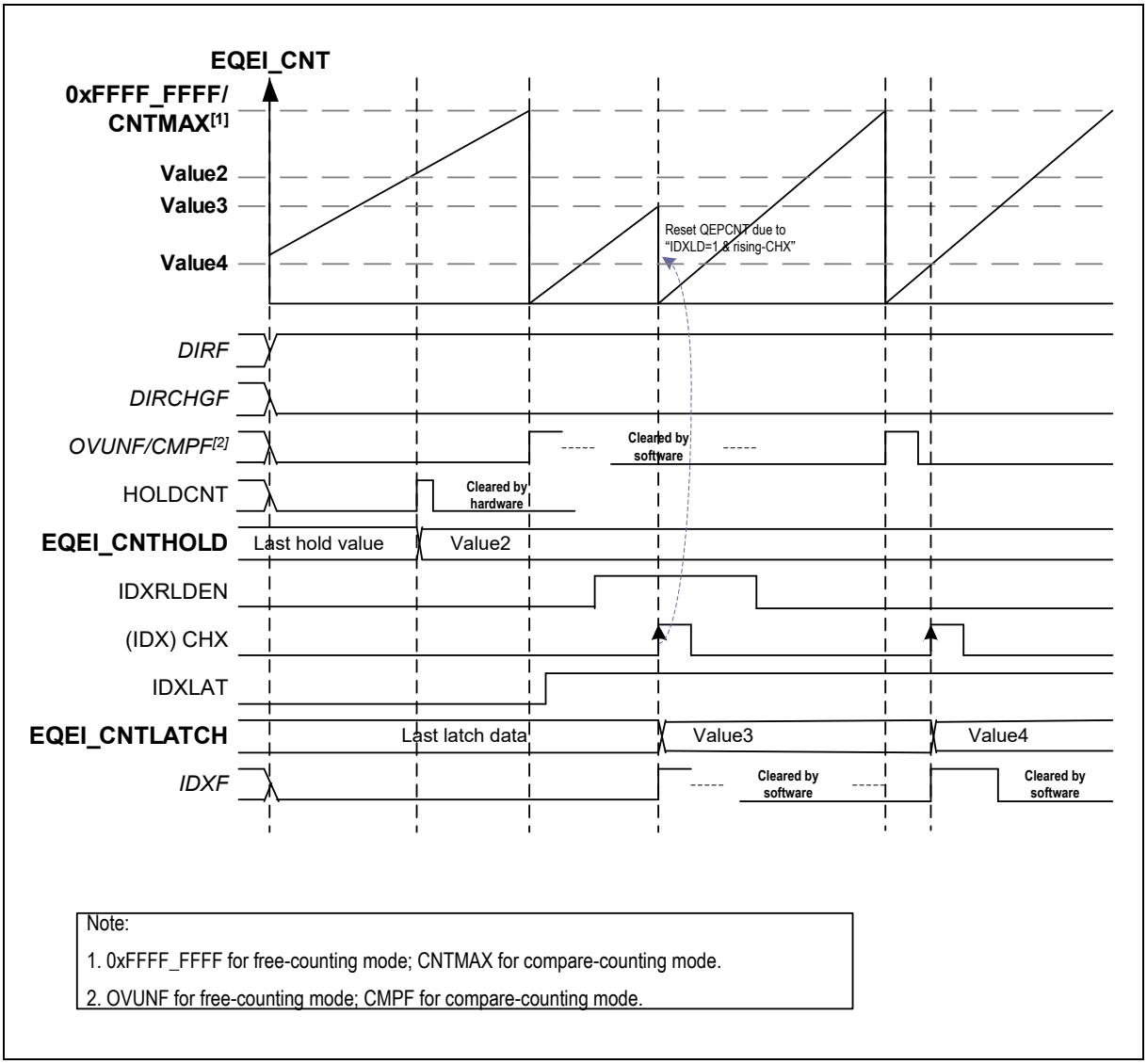


Figure 6.13-25 Capture and Latch EQEI Counter

6.13.5.17 EQEI Interrupt Architecture

There are six interrupt sources, each one of them has an interrupt flag and enable control bit, to trigger EQEI Interrupt. When the EQEI counter is up-counting and CNT(EQEI_CNT[31:0]) overflows or down-counting and underflows, the Overflow/Underflow flag OVUNF (EQEI_STATUS[2]) will be set by hardware and it will trigger EQEI Interrupt request if bit OVUNIEN (EQEI_CTL[16]) is high. When EQEI controller detects the encoder rotation change, it toggles the direction indication bit DIRF (EQEI_STATUS[8]) and the direction change flag DIRCHGF (EQEI_STATUS[3]) will be set by hardware that requests the EQEI interrupt if bit DIRIEN (EQEI_CTL[17]) is set. When the EQEI counter counting value is equal to the value of EQEI Counter Compare Register (CNTCMP(EQEI_CNTCMP[31:0])), the flag CMPF (EQEI_STATUS[1]) will be set by hardware and the EQEI Interrupt will be requested if bit CMPIEN (EQEI_CTL[18]) is high. When EQEI controller detects a rising edge at signal CHX (the filtered and polarity-set output of pin IDX), the flag IDXF(EQEI_STATUS[1]) will set by hardware and the EQEI interrupt will be requested if bit IDXIEN (EQEI_CTL[19]) is set. When the Unit Timer Counter's counting value equal to the value of Unit Timer Counter Compare Register (UTCMP(EQEI_UTCMP[31:0])), the flag UTIEF (EQEI_STATUS[10]) will be set by hardware and the EQEI Interrupt will be requested if bit TIEIEN (EQEI_CTL2[16]) is high. When the Phase error occurs in EQEI CHA and CHB, the flag PHEF (EQEI_STATUS[9]) will be set by hardware and the EQEI Interrupt will be requested if bit PHEIEN

(EQEI_CTL2[17]) is high. Note that the six flags, OVUNF(EQEI_STATUS[2]), DIRCHGF(EQEI_STATUS[3]), CMPF(EQEI_STATUS[1]), IDXF(EQEI_STATUS[0]), PHEF(EQEI_STATUS[9]), and UTIEF(EQEI_STATUS[10]) are set by hardware and must be cleared by software. Figure 6.13-26 demonstrates the architecture of Enhanced Quadrature Encoder Interface Controller interrupts.

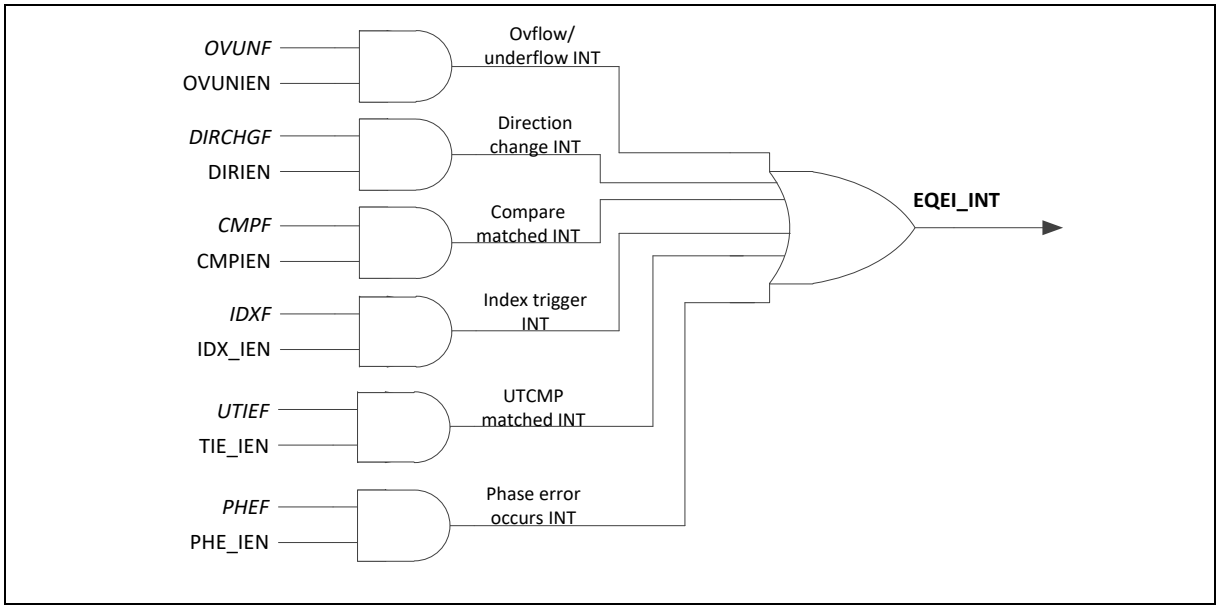


Figure 6.13-26 Enhanced Quadrature Encoder Interface Interrupt Architecture Diagram

6.13.5.18 EQEI phase error flag

Because input signals CHA and CHB have periodic changes, the mechanism of phase error monitor is used for detecting wrong phase. If the phase state is not correct, then the register PHEF (EQEI_STATUS[9]) will be set.

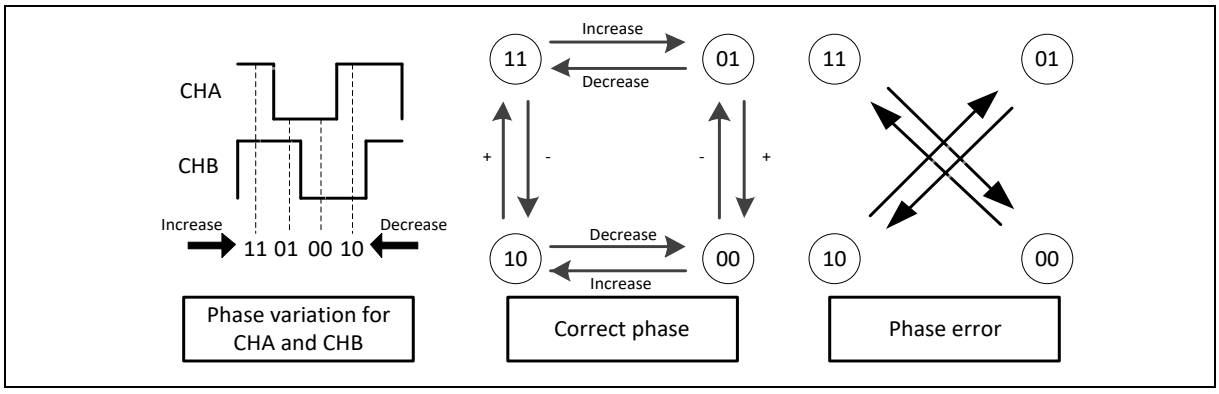


Figure 6.13-27 EQEI Phase Variation for CHA and CHB

6.13.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EQEI Base Address: **EQEI0_BA = 0x400B_0000** **EQEI1_BA = 0x400B_1000** **EQEI2_BA = 0x400B_2000** **EQEI3_BA = 0x400B_3000**				
EQEI_CNT **x=0, 1, 2, 3**	EQEIx_BA+0x00	R/W	EQEI Counter Register	0x0000_0000
EQEI_CNTHOLD **x=0, 1, 2, 3**	EQEIx_BA+0x04	R/W	EQEI Counter Hold Register	0x0000_0000
EQEI_CNTLATCH **x=0,1, 2, 3**	EQEIx_BA+0x08	R/W	EQEI Counter Index Latch Register	0x0000_0000
EQEI_CNTCMP **x=0, 1, 2, 3**	EQEIx_BA+0x0C	R/W	EQEI Counter Compare Register	0x0000_0000
EQEI_CNTMAX **x=0, 1, 2, 3**	EQEIx_BA+0x14	R/W	EQEI Pre-set Maximum Count Register	0x0000_0000
EQEI_CTL **x=0, 1, 2, 3**	EQEIx_BA+0x18	R/W	EQEI Controller Control Register	0x0000_0000
EQEI_CTL2 **x=0, 1, 2, 3**	EQEIx_BA+0x1C	R/W	EQEI Controller Control Register2	0x0000_0000
EQEI_UTCNT **x=0, 1, 2, 3**	EQEIx_BA+0x20	R/W	EQEI Unit Timer Counter Register	0x0000_0000
EQEI_UTCMP **x=0, 1, 2, 3**	EQEIx_BA+0x24	R/W	EQEI Unit Timer Compare Register	0xFFFF_FFFF
EQEI_STATUS **x=0, 1, 2, 3**	EQEIx_BA+0x2C	R/W	EQEI Controller Status Register	0x0000_0000

6.13.7 Register Description

EQEI Counter Register (EQEI_CNT)

Register	Offset	R/W	Description	Reset Value
EQEI_CNT	EQEIx_BA+0x00	R/W	EQEI Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
CNT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CNT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT							

Bits	Description	
[31:0]	**CNT**	**Enhanced Quadrature Encoder Interface Counter** A 32-bit up/down counter. When an effective phase pulse is detected, this counter is increased by one if the bit DIRF (EQEI_STATUS[8]) is one or decreased by one if the bit DIRF(EQEI_STATUS[8]) is 0. This register performs an integrator which count value is proportional to the encoder position. The pulse counter may be initialized to a predetermined value by one of three events occurs: 1. Software is written if EQEIEN (EQEI_CTL[29]) = 0. 2. Compare-match event if EQEIEN(EQEI_CTL[29])=1 and EQEI is in compare-counting mode. 3. Index signal change if EQEIEN(EQEI_CTL[29])=1 and IDXRLDEN (EQEI_CTL[27])=1.

EQEI Counter Hold Register (EQEI_CNTHOLD)

Register	Offset	R/W	Description	Reset Value
EQEI_CNTHOLD	EQEIx_BA+0x04	R/W	EQEI Counter Hold Register	0x0000_0000

31	30	29	28	27	26	25	24
CNTHOLD							
23	22	21	20	19	18	17	16
CNTHOLD							
15	14	13	12	11	10	9	8
CNTHOLD							
7	6	5	4	3	2	1	0
CNTHOLD							

Bits	Description	
[31:0]	CNTHOLD	**Enhanced Quadrature Encoder Interface Counter Hold** When the bit HOLDCNT (EQEI_CTL[24]) goes from low to high, the CNT(EQEI_CNT[31:0]) is copied into CNTHOLD (EQEI_CNTHOLD[31:0]) register.

EQEI Counter Index Latch Register (EQEI_CNTLATCH)

Register	Offset	R/W	Description	Reset Value
EQEI_CNTLATCH	EQEIx_BA+0x08	R/W	EQEI Counter Index Latch Register	0x0000_0000

31	30	29	28	27	26	25	24
CNTLATCH							
23	22	21	20	19	18	17	16
CNTLATCH							
15	14	13	12	11	10	9	8
CNTLATCH							
7	6	5	4	3	2	1	0
CNTLATCH							

Bits	Description	
[31:0]	CNTLATCH	**Enhanced Quadrature Encoder Interface Counter Index Latch** When the IDXF (EQEI_STATUS[0]) bit is set, the CNT(EQEI_CNT[31:0]) is copied into CNTLATCH (EQEI_CNTLATCH[31:0]) register.

EQEI Counter Compare Register (EQEI_CNTCMP)

Register	Offset	R/W	Description	Reset Value
EQEI_CNTCMP	EQEIx_BA+0x0C	R/W	EQEI Counter Compare Register	0x0000_0000

31	30	29	28	27	26	25	24
CNTCMP							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CNTCMP							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNTCMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNTCMP							

Bits	Description	
[31:0]	**CNTCMP**	**Enhanced Quadrature Encoder Interface Counter Compare** If the EQEI controller is in the compare-counting mode CMPEN (EQEI_CTL[28]) =1, when the value of CNT(EQEI_CNT[31:0]) matches CNTCMP(EQEI_CNTCMP[31:0]), CMPF will be set. This register is software writable.

EQEI Pre-set Maximum Count Register (EQEI_CNTMAX)

Register	Offset	R/W	Description	Reset Value
EQEI_CNTMAX	EQEIx_BA+0x14	R/W	EQEI Pre-set Maximum Count Register	0x0000_0000

31	30	29	28	27	26	25	24
CNTMAX							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CNTMAX							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNTMAX							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNTMAX							

Bits	Description	
[31:0]	**CNTMAX**	**Enhanced Quadrature Encoder Interface Preset Maximum Count** This register value determined by user stores the maximum value which may be the number of the EQEI counter for the EQEI controller compare-counting mode.

EQEI Controller Control Register (EQEI_CTL)

Register	Offset	R/W	Description	Reset Value
EQEI_CTL	EQEIx_BA+0x18	R/W	EQEI Controller Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		EQEIEN	CMPEN	IDXRLDEN	Reserved	IDXLATEN	HOLDCNT
23	22	21	20	19	18	17	16
HOLDTMR3	HOLDTMR2	HOLDTMR1	HOLDTMR0	IDXIEN	CMPIEN	DIRIEN	OVUNIEN
15	14	13	12	11	10	9	8
IDXRSTEV	IDXINV	CHBINV	CHAINV	Reserved	MODE		
7	6	5	4	3	2	1	0
IDXRSTEN	IDXEN	CHBEN	CHAEN	NFDIS	NFCLKSEL		

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**EQEIEN**	**Enhanced Quadrature Encoder Interface Controller Enable Bit** 0 = EQEI controller function Disabled. 1 = EQEI controller function Enabled.
[28]	**CMPEN**	**The Compare Function Enable Bit** The compare function in EQEI controller is to compare the dynamic counting EQEI_CNT with the compare register CNTCMP(EQEI_CNTCMP[31:0]), if CNT(EQEI_CNT[31:0]) reaches CNTCMP(EQEI_CNTCMP[31:0]), the flag CMPF will be set. 0 = Compare function Disabled. 1 = Compare function Enabled.
[27]	**IDXRLDEN**	**Index Trigger EQEI_CNT Reload Enable Bit** When this bit is high and a rising edge comes on signal CHX, the CNT(EQEI_CNT[31:0]) will be reset to 0 if the counter is in up-counting type (DIRF(EQEI_STATUS[8]) = 1); while the CNT(EQEI_CNT[31:0]) will be reloaded with CNTMAX (EQEI_CNTMAX[31:0]) content if the counter is in down-counting type (DIRF(EQEI_STATUS[8]) = 0). 0 = Reload function Disabled. 1 = EQEI_CNT re-initialized by Index signal Enabled.
[26]	**Reserved**	Reserved.
[25]	**IDXLATEN**	**Index Latch EQEI_CNT Enable Bit** If this bit is set to high, the CNT(EQEI_CNT[31:0]) content will be latched into CNTLATCH (EQEI_CNTLATCH[31:0]) at every rising on signal CHX. 0 = The index signal latch EQEI counter function Disabled. 1 = The index signal latch EQEI counter function Enabled.

[24]	**HOLDCNT**	**Hold EQEI_CNT Control** When this bit is set from low to high, the CNT(EQEI_CNT[31:0]) is copied into CNTHOLD(EQEI_CNTHOLD[31:0]). This bit may be set by writing 1 to it or Timer0~Timer3 interrupt flag TIF (TIMERx_INTSTS[0]). 0 = No operation. 1 = EQEI_CNT content is captured and stored in CNTHOLD(EQEI_CNTHOLD[31:0]). **Note:** This bit is automatically cleared after EQEI_CNTHOLD holds EQEI_CNT value.
[23]	**HOLDTMR3**	**Hold EQEI_CNT by Timer 3** 0 = TIF (TIMER3_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER3_INTSTS[0]) in timer 3 sets HOLDCNT to 1.
[22]	**HOLDTMR2**	**Hold EQEI_CNT by Timer 2** 0 = TIF(TIMER2_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER2_INTSTS[0]) in timer 2 sets HOLDCNT to 1.
[21]	**HOLDTMR1**	**Hold EQEI_CNT by Timer 1** 0 = TIF(TIMER1_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF (TIMER1_INTSTS[0]) in timer 1 sets HOLDCNT to 1.
[20]	**HOLDTMR0**	**Hold EQEI_CNT by Timer 0** 0 = TIF (TIMER0_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER0_INTSTS[0]) in timer 0 sets HOLDCNT to 1.
[19]	**IDXIEN**	**IDXF Trigger EQEI Interrupt Enable Bit** 0 = The IDXF can trigger EQEI interrupt Disabled. 1 = The IDXF can trigger EQEI interrupt Enabled.
[18]	**CMPIEN**	**CMPF Trigger EQEI Interrupt Enable Bit** 0 = CMPF can trigger EQEI controller interrupt Disabled. 1 = CMPF can trigger EQEI controller interrupt Enabled.
[17]	**DIRIEN**	**DIRCHGF Trigger EQEI Interrupt Enable Bit** 0 = DIRCHGF can trigger EQEI controller interrupt Disabled. 1 = DIRCHGF can trigger EQEI controller interrupt Enabled.
[16]	**OVUNIEN**	**OVUNF Trigger EQEI Interrupt Enable Bit** 0 = OVUNF can trigger EQEI controller interrupt Disabled. 1 = OVUNF can trigger EQEI controller interrupt Enabled.
[15]	**IDXRSTEV**	**IDX Signal Resets Enable Bit in First IDX Reset Event (Write Only)** 0 = The next IDX level high signal reset function Disabled. 1 = The next IDX level high signal reset function Eenabled. **Note:** This bit is only effective when IDXRSTEN (EQEI_CTL[7])=1 and IDXRLDEN (EQEI_CTL[27]) = 1.
[14]	**IDXINV**	**Inverse IDX Input Polarity** 0 = Not inverse IDX input polarity. 1 = IDX input polarity is inversed to EQEI controller.
[13]	**CHBINV**	**Inverse QEB Input Polarity** 0 = Not inverse QEB input polarity. 1 = QEB input polarity is inversed to EQEI controller.
[12]	**CHAINV**	**Inverse QEA Input Polarity** 0 = Not inverse QEA input polarity. 1 = QEA input polarity is inversed to EQEI controller.

[11]	**Reserved**	Reserved.
[10:8]	**MODE**	**EQEI Counting Mode Selection** There are seven quadrature encoder pulse counter operation modes. 000 = X4 Free-counting Mode. 001 = X2 Free-counting Mode. 010 = X4 Compare-counting Mode. 011 = X2 Compare-counting Mode. 100 = Phase Counting Mode Type 1. (PCMT1). 101 = Phase Counting Mode Type 2. (PCMT2). 110 = Directional Counting Mode. 111 = Reserved. **Note:** User needs to set DIRSRC(EQEI_CTL2[5:4]) when MODE(EQEI_CTL[10:8]) selects to directional counting mode.
[7]	**IDXRSTEN**	**IDX Reset EQEI Position Counter Enable Bit** 0 = Reset EQEI position counter in every time IDX signal. 1 = Reset EQEI position counter in first time IDX signal. **Note:** IDXRLDEN(EQEI_CTL[27]) should be set 1.
[6]	**IDXEN**	**IDX Input to EQEI Controller Enable Bit** 0 = IDX input to EQEI Controller Disabled. 1 = IDX input to EQEI Controller Enabled.
[5]	**CHBEN**	**QEB Input to EQEI Controller Enable Bit** 0 = QEB input to EQEI Controller Disabled. 1 = QEB input to EQEI Controller Enabled.
[4]	**CHAEN**	**QEA Input to EQEI Controller Enable Bit** 0 = QEA input to EQEI Controller Disabled. 1 = QEA input to EQEI Controller Enabled.
[3]	**NFDIS**	**EQEI Controller Input Noise Filter Disable Bit** 0 = The noise filter of EQEI controller Enabled. 1 = The noise filter of EQEI controller Disabled.
[2:0]	**NFCLKSEL**	**Noise Filter Clock Pre-divide Selection** To determine the sampling frequency of the Noise Filter clock. 000 = EQEI_CLK. 001 = EQEI_CLK/2. 010 = EQEI_CLK/4. 011 = EQEI_CLK/16. 100 = EQEI_CLK/32. 101 = EQEI_CLK/64. 110 = reserved. 111 = reserved.

EQEI Controller Control Register2 (EQEI_CTL2)

Register	Offset	R/W	Description	Reset Value
EQEI_CTL2	EQEIx_BA+0x1C	R/W	EQEI Controller Control Register2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						UTIEIEN	PHEIEN
15	14	13	12	11	10	9	8
Reserved				IDXRSTUTS	UTEVTRST	UTHOLDEN	UTEN
7	6	5	4	3	2	1	0
Reserved		DIRSRC		Reserved	CRS		SWAPEN

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**UTIEIEN**	**UTIEF Trigger EQEI Interrupt Enable Bit** 0 = UTIEF can trigger EQEI controller interrupt Disabled. 1 = UTIEF can trigger EQEI controller interrupt Enabled.
[16]	**PHEIEN**	**PHEF Trigger EQEI Interrupt Enable Bit** 0 = PHEF can trigger EQEI controller interrupt Disabled. 1 = PHEF can trigger EQEI controller interrupt Enabled.
[15:12]	**Reserved**	Reserved.
[11]	**IDXRSTUTS**	**IDX Resets Unit Timer Select Bit** 0 = Unit timer will not be reset when IDX reset event happens. 1 = Resets unit timer or not will follow EQEI_CNT when IDX reset event happens.
[10]	**UTEVTRST**	**Enable Bit to Reset EQEI Position Counter by Unit Timer Event** 0 = Reset EQEI position counter feature when unit timer counter event occurs Disabled. 1 = Reset EQEI position counter feature when unit timer counter event occurs Enabled.
[9]	**UTHOLDEN**	**Unit Timer Counter Hold Enable Bit** 0 = No operation. 1 = EQEI_CNT content is captured and stored in CNTHOLD(EQEI_CNTHOLD[31:0]) when UTCNT matches UTCMP(EQEI_UTCMP[31:0]).
[8]	**UTEN**	**Unit Timer Function Enable Bit** 0 = EQEI unit timer function Disabled. 1 = EQEI unit timer function Enabled.
[7:6]	**Reserved**	Reserved.

[5:4]	**DIRSRC**	**Direction Signal Source Select** 00 = Direction signal is determined from EQEI system calculation. 01 = reserved. 10 = Direction signal is tied 1 only for direction up count mode. 11 = Direction signal is tied 0 only for down count mode.
[3]	**Reserved**	Reserved.
[2:1]	**CRS**	**Clock Rate Setting without Quadrature Mode** 00 = EQEI counter only counts the falling edge. 01 = EQEI counter only counts the rising edge. 10 = EQEI counter counts the rising and falling edge. 11 = reserved.
[0]	**SWAPEN**	**Swap Function Enable Bit** 0 = EQEI swap function Disabled. 1 = EQEI swap function Enabled.

EQEI Unit Timer Counter Register (EQEI_UTCNT)

Register	Offset	R/W	Description	Reset Value
EQEI_UTCNT	EQEIx_BA+0x20	R/W	EQEI Unit Timer Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
UTCNT							
23	22	21	20	19	18	17	16
UTCNT							
15	14	13	12	11	10	9	8
UTCNT							
7	6	5	4	3	2	1	0
UTCNT							

Bits	Description	
[31:0]	**UTCNT**	**Unit Timer Counter** A 32-bit unit timer counter which may be reset to an initial value when any of the following events occur: 1. Software is written if UTEN (EQEI_CTL2[8]) = 0. 2. UT_EN (EQEI_CTL2[8]) =1, and the unit timer counter value matches UTCMP(EQEI_UTCMP[31:0]). 3. IDXRLDEN(EQEI_CTL[27]) =1 and IDXRSTUTS(EQEI_CTL2[11]=1, determine the unit timer to be reset or not will follow EQEI_CNT when IDX reset event happens.

EQEI Unit Timer Compare Register (EQEI_UTCMP)

Register	Offset	R/W	Description	Reset Value
EQEI_UTCMP	EQEIx_BA+0x24	R/W	EQEI Unit Timer Compare Register	0xFFFF_FFFF

31	30	29	28	27	26	25	24
UTCMP							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
UTCMP							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
UTCMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UTCMP							

Bits	Description	
[31:0]	**UTCMP**	**Unit Timer Counter Compare** If the EQEI unit timer is enabled (EQEI_CTL2[8]) =1, and the unit timer counter value also matches UTCMP(EQEI_UTCMP[31:0]), then UTIEF (EQEI_STATUS[10]) will be set. This register is software writable.

EQEI Controller Status Register (EQEI_STATUS)

Register	Offset	R/W	Description	Reset Value
EQEI_STATUS	EQEIx_BA+0x2C	R/W	EQEI Controller Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						UTIEF	PHEF
15	14	13	12	11	10	9	8
Reserved						FIDXEF	DIRF
7	6	5	4	3	2	1	0
Reserved				DIRCHGF	OVUNF	CMPF	IDXF

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**UTIEF**	**EQEI Unit Timer Event Flag** 0 = No timer event occurs in EQEI unit timer counter. 1 = Unit timer event occurs in EQEI unit timer counter. **Note:** This bit is only cleared by writing 1 to it.
[16]	**PHEF**	**EQEI Phase Error Flag** 0 = No Phase error occurs in EQEI CHA and CHB. 1 = Phase error occurs in EQEI CHA and CHB. **Note:** This bit is only cleared by writing 1 to it.
[15:10]	**Reserved**	Reserved.
[9]	**FIDXEF**	**First IDX Signal Reset Event Flag (Read Only)** 0 = The first IDX reset event has not happened yet. 1 = The first IDX reset event has happened. **Note:** This bit only effective when IDXRSTEN (EQEI_CTL[7])=1 and IDXRLDEN (EQEI_CTL[27])=1.
[8]	**DIRF**	**EQEI Counter Counting Direction Indication** 0 = EQEI Counter is in down-counting. 1 = EQEI Counter is in up-counting. **Note:** This bit is set/reset by hardware according to the phase detection between CHA and CHB.
[7:4]	**Reserved**	Reserved.
[3]	**DIRCHGF**	**Direction Change Flag** Flag is set by hardware while EQEI counter counting direction is changed. Software can clear this bit by writing 1 to it. 0 = No change in EQEI counter counting direction. 1 = EQEI counter counting direction is changed. **Note:** This bit is only cleared by writing 1 to it.

[2]	**OVUNF**	**EQEI Counter Overflow or Underflow Flag** Flag is set by hardware while CNT(EQEI_CNT[31:0]) overflows from 0xFFFF_FFFF to 0 in free-counting mode or from the CNTMAX (EQEI_CNTMAX[31:0]) to 0 in compare-counting mode. Similarly, the flag is set while EQEI counter underflows from 0 to 0xFFFF_FFFF or CNTMAX (EQEI_CNTMAX[31:0]). 0 = No overflow or underflow occurs in EQEI counter. 1 = EQEI counter occurs counting overflow or underflow. **Note:** This bit is only cleared by writing 1 to it.
[1]	**CMPF**	**Compare-match Flag** If the EQEI compare function is enabled, the flag is set by hardware while EQEI counter up or down counts and reach to the CNTCMP(EQEI_CNTCMP[31:0]). 0 = EQEI counter does not match with CNTCMP(EQEI_CNTCMP[31:0]). 1 = EQEI counter counts to the same as CNTCMP(EQEI_CNTCMP[31:0]). **Note:** This bit is only cleared by writing 1 to it.
[0]	**IDXF**	**IDX Detected Flag** When the EQEI controller detects a rising edge on signal CHX it will set flag IDXF to high. 0 = No rising edge detected on signal CHX. 1 = A rising edge occurs on signal CHX. **Note:** This bit is only cleared by writing 1 to it.

6.14 Watchdog Timer (WDT)

6.14.1 Overview

The Watchdog Timer (WDT) is used to perform a system reset when system runs into an unknown state. This prevents system from hanging for an infinite period of time. Besides, this Watchdog Timer supports the function to wake up system from Idle/Power-down mode.

6.14.2 Features

- 20-bit free running up counter for WDT time-out interval
- Selectable time-out interval (2^4 ~ 2^{20}) and the time-out interval is 1.6 ms ~ 104.8576 s if WDT_CLK = 10 kHz
- System kept in reset state for a period of (1 / WDT_CLK) * 63
- Supports selectable WDT reset delay period, including 1026, 130, 18 or 3 WDT_CLK reset delay period
- Supports to force WDT enabled after chip powered on or reset by setting CWDTEN[2:0] in Config0 register
- Supports WDT time-out wake-up function only if WDT clock source is selected as LIRC 10 kHz or LXT

6.14.3 Block Diagram

The WDT clock control is shown in Figure 6.14-1

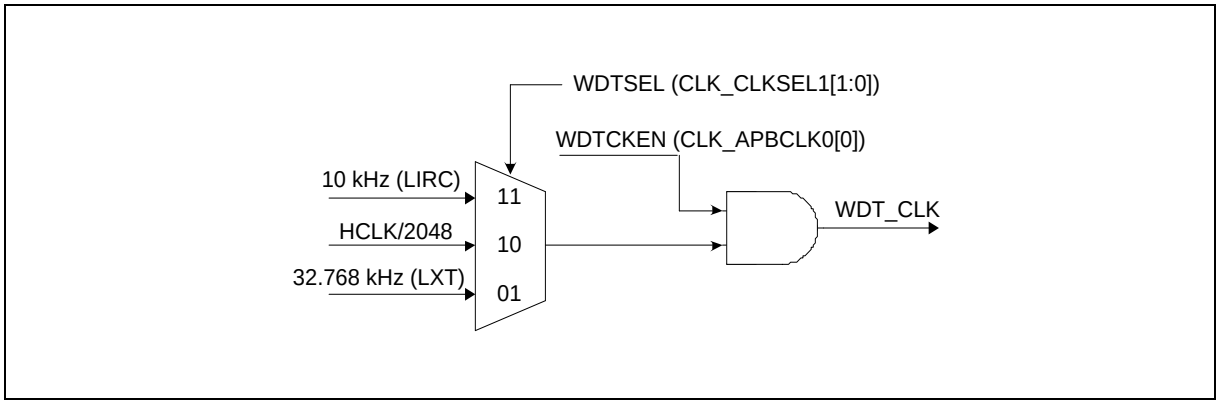
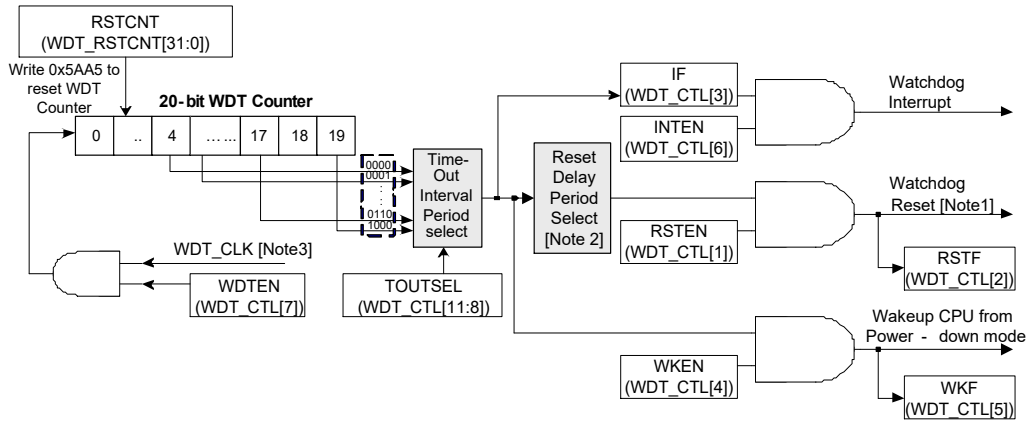


Figure 6.14-1 Watchdog Timer Clock Control



Note1: WDT resets CPU and lasts 63 WDT_CLK.

Note2: The WDT reset delay period can be selected as 3/18/130/1026 WDT_CLK.

Note3: Chip can be woken by WDT time-out interrupt signal generated only if WDT clock source selects LXT or LIRC

Figure 6.14-2 Watchdog Timer Block Diagram

6.14.4 Basic Configuration

- Clock Source Configuration
  - Select the source of WDT peripheral clock on WDTSEL (CLK_CLKSEL1[1:0])
  - Enable WDT peripheral clock in WDTCKEN (CLK_APBCLK0[0])
  - Force to enable WDT controller after chip powered on or reset in CWDTEN[2:0] (CWDTEN[2] is Config0[31], CWDTEN[1:0] is Config0[4:3]) while CWDTRN[2:0] is not configured to 111

6.14.5 Functional Description

The WDT includes a 20-bit free running up counter with programmable time-out intervals. Table 6.14-1 shows the WDT time-out interval period selection and Figure 6.14-3 shows the WDT time-out interval and reset period timing.

6.14.5.1 WDT Time-out Interrupt

Setting WDTEN (WDT_CTL[7]) to 1 will enable the WDT function and the WDT counter to start counting up. The SYNC (WDT_CTL[30]) can be indicated whether enable/disable WDTEN function is completed or not. There are nine time-out interval period can be selected by setting TOUTSEL (WDT_CTL[11:8]). When the WDT up counter reaches the TOUTSEL (WDT_CTL[11:8]) settings, WDT time-out interrupt will occur then WDT time-out interrupt flag IF (WDT_CTL[3]) will be set to 1 immediately. If INTEN (WDT_CTL[6]) is enabled, WDT time-out interrupt will inform CPU.

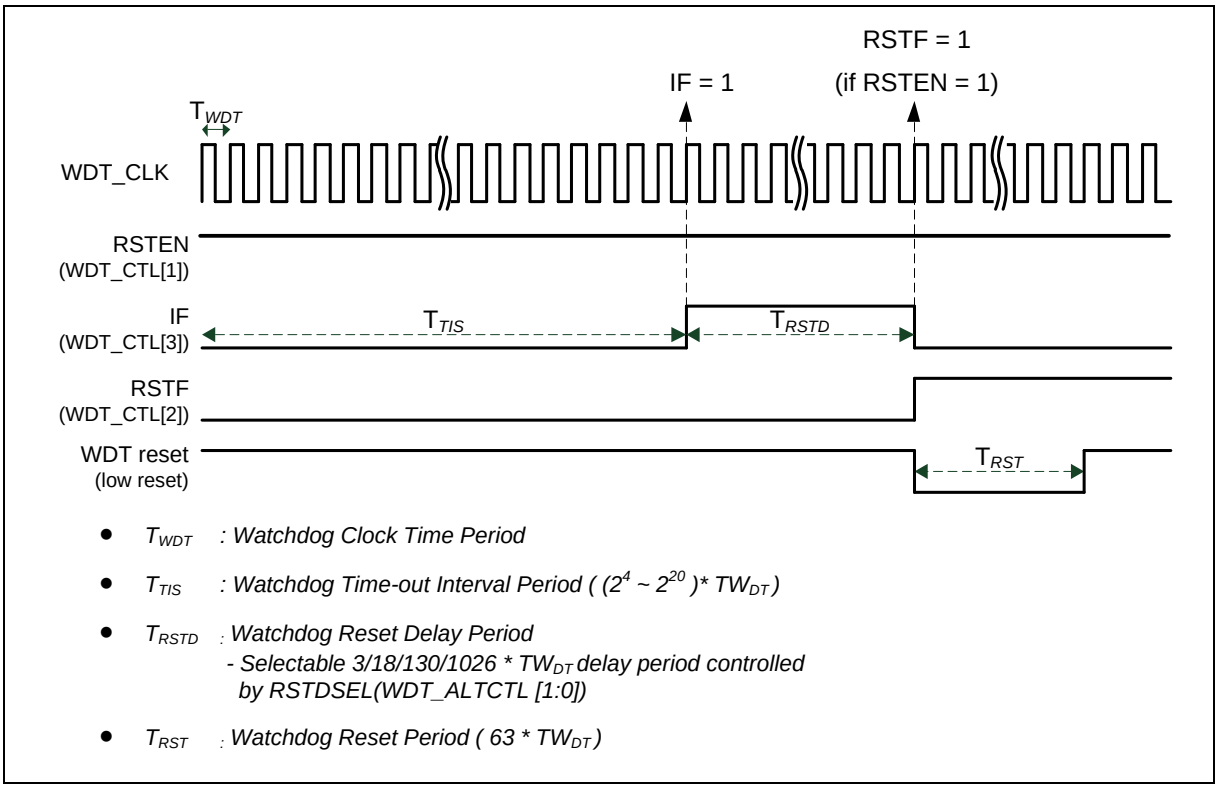
6.14.5.2 WDT Reset Delay Period and Reset System

There is a specified T_{RSTD} reset delay period follows the IF (WDT_CTL[3]) is setting to 1. User should program 0x00005AA5 to RSTCNT (WDT_RSTCNT[31:0]) to reset the 20-bit WDT up counter value to avoid generate WDT time-out reset signal before the T_{RSTD} reset delay period expires. Moreover, user should set RSTDSEL (WDT_ALTCTL[1:0]) to select reset delay period to clear WDT counter. If the WDT up counter value has not been cleared after the specific T_{RSTD} delay period expires, the WDT control will set RSTF (WDT_CTL[2]) to 1 if RSTEN (WDT_CTL[1]) bit is enabled, then chip enters to reset state

immediately. T_{RST} reset period will keep last 63 WDT clocks and then chip restarts executing program from reset vector (0x0000_0000). The RSTF (WDT_CTL[2]) will keep 1 after WDT time-out resets the chip, user can check RSTF (WDT_CTL[2]) by software to recognize the system has been reset by WDT time-out reset or not.

TOUTSEL	Time-Out Interval Period T_{TIS}	Reset Delay Period T_{RSTD}
0000	$2^4 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0001	$2^6 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0010	$2^8 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0011	$2^{10} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0100	$2^{12} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0101	$2^{14} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0110	$2^{16} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
0111	$2^{18} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
1000	$2^{20} * T_{WDT}$	(3/18/130/1026) * T_{WDT}

Table 6.14-1 Watchdog Timer Time-out Interval Period Selection



- T_{WDT} : *Watchdog Clock Time Period*
- T_{TIS} : *Watchdog Time-out Interval Period ((2^4 ~ 2^{20})* TW_{DT})*
- T_{RSTD} : *Watchdog Reset Delay Period*
  *- Selectable 3/18/130/1026 * TW_{DT} delay period controlled by RSTDSEL(WDT_ALTCTL [1:0])*
- T_{RST} : *Watchdog Reset Period (63 * TW_{DT})*

Figure 6.14-3 Watchdog Timer Time-out Interval and Reset Period Timing

6.14.5.3 WDT Wake-up

If WDT clock source is selected to 10 kHz or LXT, system can be woken up from Power-down mode while WDT time-out interrupt signal is generated and WKEN (WDT_CTL[4]) enabled. Note that user should set LXTEN (CLK_PWRCTL[1]) or LIRCEN (CLK_PWRCTL[3]) to select clock source before

system enters Power-down mode because the system peripheral clock are disabled when system is in Power-down mode. In the meanwhile, the WKF (WDT_CTL[5]) will be set to 1 automatically, and user can check WKF (WDT_CTL[5]) status by software to recognize the system has been woken up by WDT time-out interrupt or not.

6.14.5.4 WDT ICE Debug

When ICE is connected to MCU, WDT counter is counting or not by ICEDEBUG (WDT_CTL[31]). The default value of ICEDEBUG is 0, WDT counter will stop counting when CPU is held by ICE. If ICEDEBUG is set to 1, WDT counter will keep counting no matter CPU is held by ICE or not.

6.14.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
WDT Base Address: **WDT_BA = 0x4004_0000**				
WDT_CTL	WDT_BA+0x00	R/W	WDT Control Register	0x0000_0800
WDT_ALTCTL	WDT_BA+0x04	R/W	WDT Alternative Control Register	0x0000_0000
WDT_RSTCNT	WDT_BA+0x08	W	WDT Reset Counter Register	0x0000_0000

6.14.7 Register Description

WDT Control Register (WDT_CTL)

Register	Offset	R/W	Description	Reset Value
WDT_CTL	WDT_BA+0x00	R/W	WDT Control Register	0x0000_0800

31	30	29	28	27	26	25	24
ICEDEBUG	**SYNC**	**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**TOUTSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDTEN	**INTEN**	**WKF**	**WKEN**	**IF**	**RSTF**	**RSTEN**	**Reserved**

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Bit (Write Protect)** 0 = ICE debug mode acknowledgement affects WDT counting. WDT up counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. WDT up counter will keep going no matter CPU is held by ICE or not. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[30]	**SYNC**	**WDT Enable Control SYNC Flag Indicator (Read Only)** If user executes enable/disable WDTEN (WDT_CTL[7]), this flag can be indicated enable/disable WDTEN function is completed or not. 0 = Setting WDTEN bit is completed and WDT is ready. 1 = Setting WDTEN bit is synchronizing and not become active yet. **Note:** Performing enable or disable WDTEN bit needs 4 * WDT_CLK period to become active.
[29:12]	**Reserved**	Reserved.
[11:8]	**TOUTSEL**	**WDT Time-out Interval Selection (Write Protect)** These four bits select the time-out interval period for the WDT. 0000 = 2^4 * WDT_CLK. 0001 = 2^6 * WDT_CLK. 0010 = 2^8 * WDT_CLK. 0011 = 2^{10} * WDT_CLK. 0100 = 2^{12} * WDT_CLK. 0101 = 2^{14} * WDT_CLK. 0110 = 2^{16} * WDT_CLK. 0111 = 2^{18} * WDT_CLK. 1000 = 2^{20} * WDT_CLK. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	**WDTEN**	**WDT Enable Bit (Write Protect)**

		0 = WDT Disabled (This action will reset the internal up counter value). 1 = WDT Enabled. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** If CWDTEN[2:0] (combined by Config0[31] and Config0[4:3]) bits is not configured to 111, this bit is forced as 1 and user cannot change this bit to 0.
[6]	**INTEN**	**WDT Time-out Interrupt Enable Bit (Write Protect)** If this bit is enabled, the WDT time-out interrupt signal is generated and inform to CPU. 0 = WDT time-out interrupt Disabled. 1 = WDT time-out interrupt Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**WKF**	**WDT Time-out Wake-up Flag** This bit indicates the interrupt wake-up flag status of WDT 0 = WDT does not cause chip wake-up. 1 = Chip wake-up from Idle or Power-down mode if WDT time-out interrupt signal generated. **Note:** This bit is cleared by writing 1 to it.
[4]	**WKEN**	**WDT Time-out Wake-up Function Control (Write Protect)** If this bit is set to 1, while WDT time-out interrupt flag IF (WDT_CTL[3]) is generated to 1 and interrupt enable bit INTEN (WDT_CTL[6]) is enabled, the WDT time-out interrupt signal will generate a wake-up trigger event to chip. 0 = Wake-up trigger event Disabled if WDT time-out interrupt signal generated. 1 = Wake-up trigger event Enabled if WDT time-out interrupt signal generated. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** Chip can be woken up by WDT time-out interrupt signal generated only if WDT clock source is selected to 10 kHz internal low speed RC oscillator (LIRC) or LXT.
[3]	**IF**	**WDT Time-out Interrupt Flag** This bit will set to 1 while WDT up counter value reaches the selected WDT time-out interval. 0 = WDT time-out interrupt did not occur. 1 = WDT time-out interrupt occurred. **Note:** This bit is cleared by writing 1 to it.
[2]	**RSTF**	**WDT Time-out Reset Flag** This bit indicates the system has been reset by WDT time-out reset or not. 0 = WDT time-out reset did not occur. 1 = WDT time-out reset occurred. **Note:** This bit is cleared by writing 1 to it.
[1]	**RSTEN**	**WDT Time-out Reset Enable Bit (Write Protect)** Setting this bit will enable the WDT time-out reset function If the WDT up counter value has not been cleared after the specific WDT reset delay period expires. 0 = WDT time-out reset function Disabled. 1 = WDT time-out reset function Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**Reserved**	Reserved.

WDT Alternative Control Register (WDT_ALTCTL)

Register	Offset	R/W	Description	Reset Value
WDT_ALTCTL	WDT_BA+0x04	R/W	WDT Alternative Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**RSTDSEL**	

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1:0]	**RSTDSEL**	**WDT Reset Delay Selection (Write Protect)** When WDT time-out happened, user has a time named WDT Reset Delay Period to clear WDT counter by programming 0x5AA5 to prevent WDT time-out reset happened. User can select a suitable setting of RSTDSEL for different WDT Reset Delay Period. 00 = WDT Reset Delay Period is 1026 * WDT_CLK. 01 = WDT Reset Delay Period is 130 * WDT_CLK. 10 = WDT Reset Delay Period is 18 * WDT_CLK. 11 = WDT Reset Delay Period is 3 * WDT_CLK. **Note 1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note 2:** This register will be reset to 0 if WDT time-out reset happened.

WDT Reset Counter Register (WDT_RSTCNT)

Register	Offset	R/W	Description	Reset Value
WDT_RSTCNT	WDT_BA+0x08	W	WDT Reset Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
RSTCNT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RSTCNT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RSTCNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RSTCNT							

Bits	Description	
[31:0]	**RSTCNT**	**WDT Reset Counter Register** Writing 0x00005AA5 to this field will reset the internal 20-bit WDT up counter value to 0. **Note:** Performing RSTCNT to reset counter needs 2 * WDT_CLK period to become active.

6.15 Window Watchdog Timer (WWDT)

6.15.1 Overview

The Window Watchdog Timer (WWDT) is used to perform a system reset within a specified window period to prevent software running to uncontrollable status by any unpredictable condition.

6.15.2 Features

- 6-bit down counter value (CNTDAT, WWDT_CNT[5:0]) and 6-bit compare value (CMPDAT, WWDT_CTL[21:16]) to make the WWDT time-out window period flexible
- Supports 4-bit value (PSCSEL, WWDT_CTL[11:8]) to programmable maximum 11-bit prescale counter period of WWDT counter
- WWDT counter suspends in Idle/Power-down mode

6.15.3 Block Diagram

The WWDT clock control is shown in Figure 6.15-1.

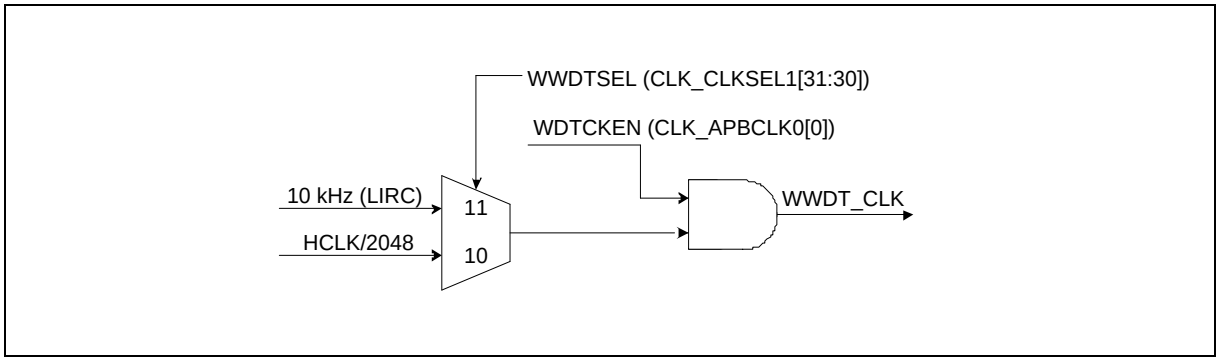


Figure 6.15-1 WWDT Clock Control

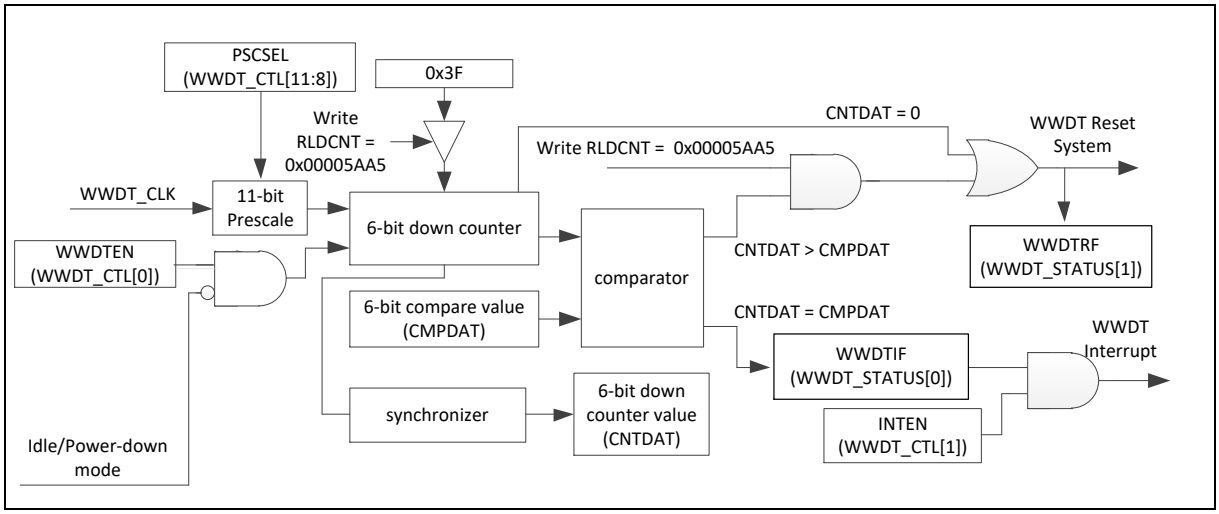


Figure 6.15-2 WWDT Block Diagram

6.15.4 Basic Configuration

- Clock Source Configuration
  - Select the source of WWDT peripheral clock on WWDTSEL (CLK_CLKSEL1[31:30])
  - Enable WWDT peripheral clock in WDTCKEN (CLK_APBCLK0[0]).

6.15.5 Functional Description

The WWDT includes a 6-bit down counter with programmable prescale value to define different WWDT time-out intervals. The clock source of 6-bit WWDT is based on system clock divide 2048 (HCLK/2048) or 10 kHz internal low speed RC oscillator (LIRC) with a programmable 11-bit prescale counter value which controlled by PSCSEL (WWDT_CTL[11:8]). Also, the correlate of PSCSEL (WWDT_CTL[11:8]) and prescale value are listed in Table 6.15-1.

PSCSEL	Prescale Value	Max. Time-Out Period	Max. Time-Out Interval (WWDT_CLK=10 kHz)
0000	1	1 * 64 * T_{WWDT}	6.4 ms
0001	2	2 * 64 * T_{WWDT}	12.8 ms
0010	4	4 * 64 * T_{WWDT}	25.6 ms
0011	8	8 * 64 * T_{WWDT}	51.2 ms
0100	16	16 * 64 * T_{WWDT}	102.4 ms
0101	32	32 * 64 * T_{WWDT}	204.8 ms
0110	64	64 * 64 * T_{WWDT}	409.6 ms
0111	128	128 * 64 * T_{WWDT}	819.2 ms
1000	192	192 * 64 * T_{WWDT}	1.2288 s
1001	256	256 * 64 * T_{WWDT}	1.6384 s
1010	384	384 * 64 * T_{WWDT}	2.4576 s
1011	512	512 * 64 * T_{WWDT}	3.2768 s
1100	768	768 * 64 * T_{WWDT}	4.9152 s
1101	1024	1024 * 64 * T_{WWDT}	6.5536 s
1110	1536	1536 * 64 * T_{WWDT}	9.8304 s
1111	2048	2048 * 64 * T_{WWDT}	13.1072 s

Table 6.15-1 WWDT Prescale Value Selection

6.15.5.1 WWDT Counting

When the WWDTEN (WWDT_CTL[0]) is set, WWDT down counter will start counting from 0x3F to 0. To prevent program runs to disable WWDT counter counting unexpected, the WWDT_CTL register can only be written once after chip is powered on or reset. User cannot disable WWDT counter counting (WWDTEN), change counter prescale period (PSCSEL) or change window compare value (CMPDAT) while WWDTEN (WWDT_CTL[0]) has been enabled by user unless chip is reset.

To avoid the system is reset while CPU clock is disabled, the WWDT counter will stop counting when CPU enters Idle/Power-down mode. After CPU enters normal mode, the WWDT counter will start down counting.

6.15.5.2 WWDT Compare Match Interrupt

During down counting by the WWDT counter, the WWDTIF (WWDT_STATUS[0]) is set to 1 while the WWDT counter value (CNTDAT) is equal to window compare value (CMPDAT) and WWDTIF can be cleared by user; if INTEN (WWDT_CTL[1]) is also set to 1 by user, the WWDT compare match interrupt signal is generated also while WWDTIF is set to 1 by hardware.

6.15.5.3 WWDT Reset System

Figure 6.15-3 shows three cases of WWDT reset and reload behavior.

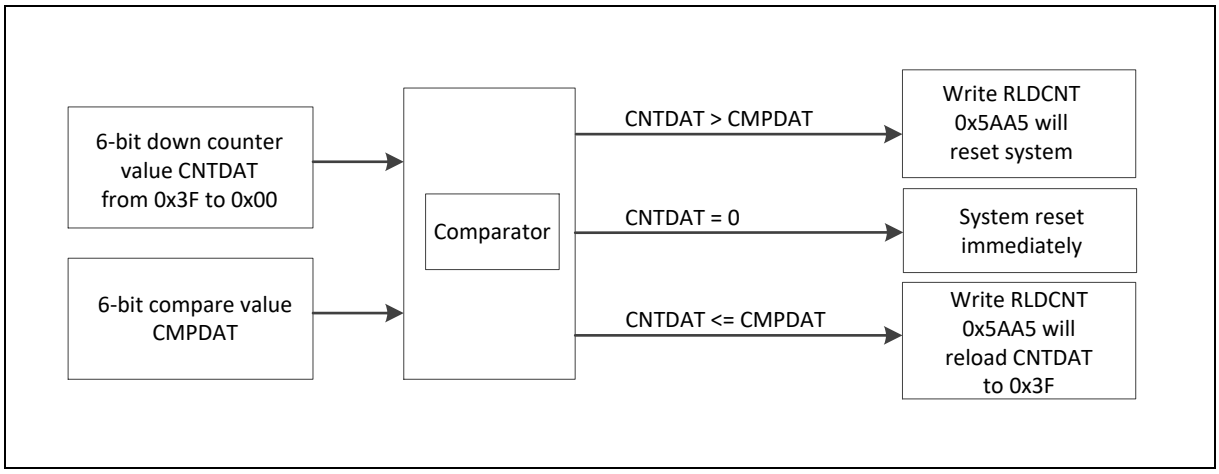


Figure 6.15-3 WWDT Reset and Reload Behavior

If the current CNTDAT (WWDT_CNT[5:0]) is larger than CMPDAT (WWDT_CTL[21:16]) and user writes 0x00005AA5 to the WWDT_RLDCNT register, the WWDT reset system signal will be generated immediately to cause chip reset also. The waveform of WWDT reload counter when CNTDAT > CMPDAT is shown in Figure 6.15-4.

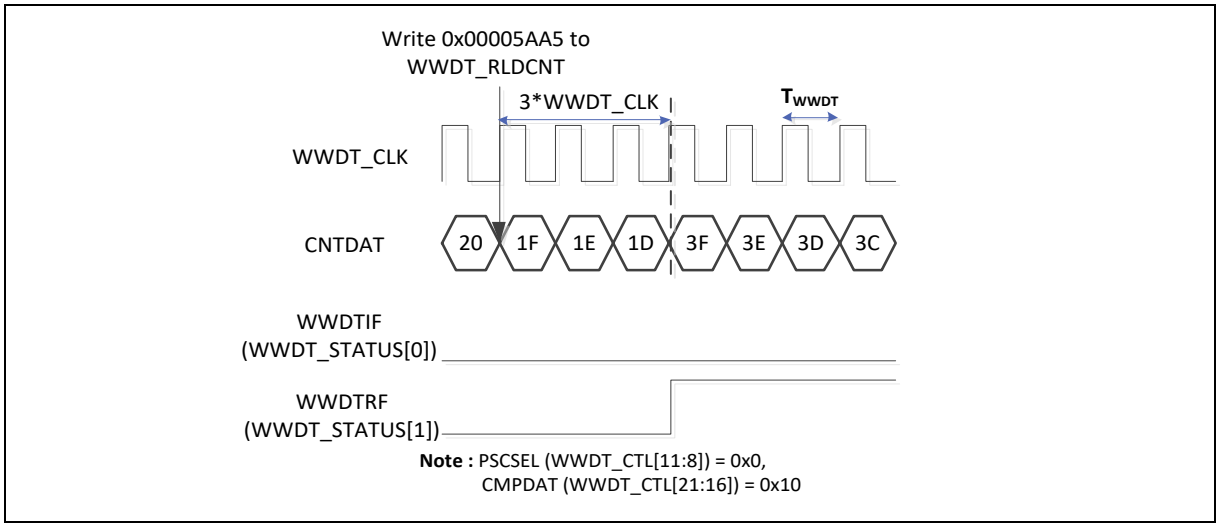


Figure 6.15-4 WWDT Reload Counter When CNTDAT > CMPDAT

When WWDTIF (WWDT_STATUS[0]) is generated, user must reload WWDT counter value to 0x3F by writing 0x00005AA5 to WWDT_RLDCNT register, and also to prevent the WWDT counter value from reaching 0 and generate WWDT reset system signal to inform system reset. Figure 6.15-5 shows the waveform of WWDT reload counter when CNTDAT is less than CMPDAT and Figure 6.15-6 shows the WWDT generates reset system signal (WWDTRF) if user does not write 0x00005AA5 to WWDT_RLD before WWDT counter value reaches 0.

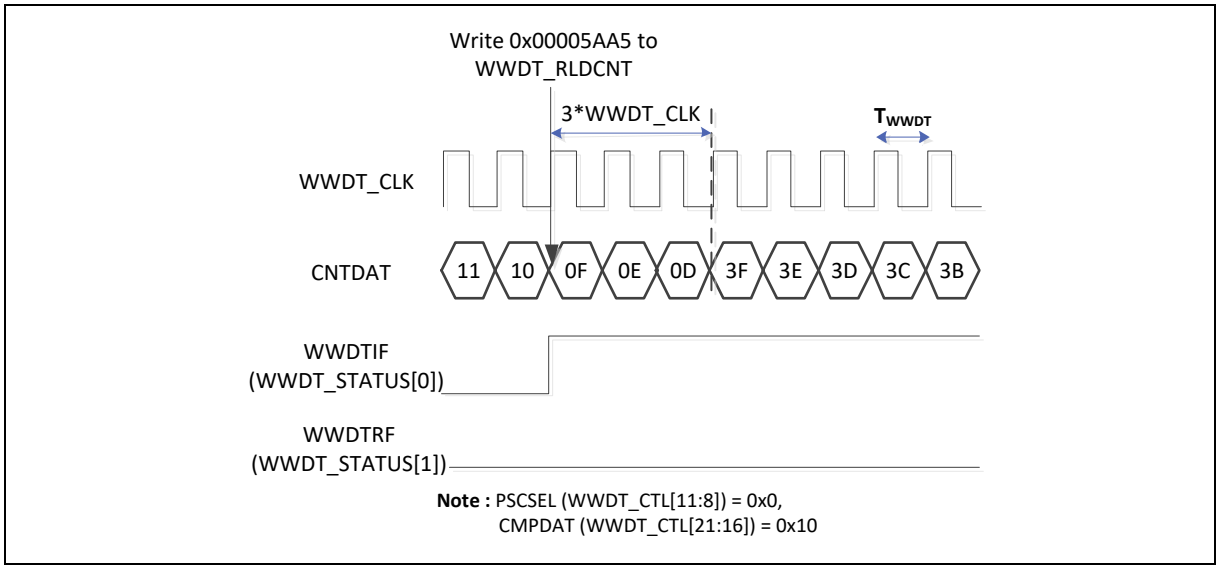


Figure 6.15-5 WWDT Reload Counter When WWDT_CNT < WINCMP

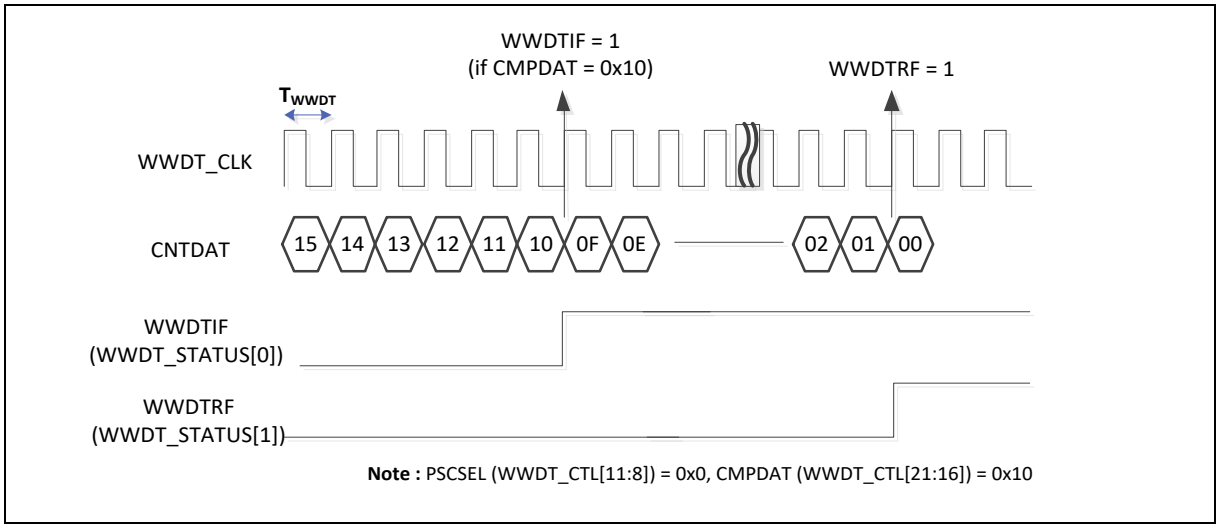


Figure 6.15-6 WWDT Interrupt and Reset Signals

6.15.5.4 WWDT Window Setting Limitation

When user writes 0x00005AA5 to WWDT_RLDCNT register to reload WWDT counter value to 0x3F, it needs 3 WWDT clocks to sync the reload command to actually perform reload action. Note that if user sets PSCSEL (WWDT_CTL[11:8]) to 0000, the counter prescale value should be as 1, and the CMPDAT (WWDT_CTL[21:16]) must be larger than 2. Otherwise, writing WWDT_RLDCNT register to reload WWDT counter value to 0x3F is unavailable, WWDTIF (WWDT_STATUS[0]) is generated, and WWDT reset system event always happened. The WWDT CMPDAT setting limitation is shown in Table 6.15-2.

If user sets CMPDATA as 0x3F and 0x0, the interrupt doesn't occur. The reset occurs when WWDT counts to 0x0, so the interrupt doesn't occur when CMPDATA is 0x0.

PSCSEL	Prescale Value	Valid CMPDAT Value
0000	1	0x3 ~ 0x3E
0001	2	0x2 ~ 0x3E

Others	Others	0x1 ~ 0x3E

Table 6.15-2 CMPDAT Setting Limitation

6.15.5.5 WWDT ICE Debug

When ICE is connected to MCU, the WWDT counter is counting (or not) by ICEDEBUG (WWDT_CTL[31]). The default value of ICEDEBUG is 0. The WWDT counter will stop counting when CPU is held by ICE. If ICEDEBUG is set to 1, WWDT counter will keep counting no matter CPU is held by ICE or not.

6.15.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
WWDT Base Address: **WWDT_BA = 0x4004_0100**				
WWDT_RLDCNT	WWDT_BA+0x00	W	WWDT Reload Counter Register	0x0000_0000
WWDT_CTL	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800
WWDT_STATUS	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000
WWDT_CNT	WWDT_BA+0x0C	R	WWDT Counter Value Register	0x0000_003F

6.15.7 Register Description

WWDT Reload Counter Register (WWDT_RLDCNT)

Register	Offset	R/W	Description	Reset Value
WWDT_RLDCNT	WWDT_BA+0x00	W	WWDT Reload Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
RLDCNT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RLDCNT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RLDCNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RLDCNT							

Bits	Description	
[31:0]	**RLDCNT**	**WWDT Reload Counter Register** Writing 0x00005AA5 to this register will reload the WWDT counter value to 0x3F. **Note:** User can only write WWDT_RLDCNT register to reload WWDT counter value when current WWDT counter value between 0 and CMPDAT (WWDT_CTL[21:16]). If user writes WWDT_RLDCNT when current WWDT counter value is larger than CMPDAT, WWDT reset signal will be generated immediately.

WWDT Control Register (WWDT_CTL)

Register	Offset	R/W	Description	Reset Value
WWDT_CTL	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800

Note: This register can be written only one time after chip is powered on or reset.

31	30	29	28	27	26	25	24
ICEDEBUG	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**CMPDAT**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**PSCSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**INTEN**	**WWDTEN**

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Bit** 0 = ICE debug mode acknowledgement effects WWDT counting. WWDT down counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. WWDT down counter will keep going no matter CPU is held by ICE or not.
[30:22]	**Reserved**	Reserved.
[21:16]	**CMPDAT**	**WWDT Window Compare** Set this register to adjust the valid reload window. **Note:** User can only write WWDT_RLDCNT register to reload WWDT counter value when current WWDT counter value between 0 and CMPDAT. If user writes WWDT_RLDCNT register when current WWDT counter value larger than CMPDAT, WWDT reset signal will generate immediately.
[15:12]	**Reserved**	Reserved.
[11:8]	**PSCSEL**	**WWDT Counter Prescale Period Selection** 0000 = Pre-scale is 1; Max time-out period is 1 * 64 * WWDT_CLK. 0001 = Pre-scale is 2; Max time-out period is 2 * 64 * WWDT_CLK. 0010 = Pre-scale is 4; Max time-out period is 4 * 64 * WWDT_CLK. 0011 = Pre-scale is 8; Max time-out period is 8 * 64 * WWDT_CLK. 0100 = Pre-scale is 16; Max time-out period is 16 * 64 * WWDT_CLK. 0101 = Pre-scale is 32; Max time-out period is 32 * 64 * WWDT_CLK. 0110 = Pre-scale is 64; Max time-out period is 64 * 64 * WWDT_CLK. 0111 = Pre-scale is 128; Max time-out period is 128 * 64 * WWDT_CLK. 1000 = Pre-scale is 192; Max time-out period is 192 * 64 * WWDT_CLK. 1001 = Pre-scale is 256; Max time-out period is 256 * 64 * WWDT_CLK. 1010 = Pre-scale is 384; Max time-out period is 384 * 64 * WWDT_CLK. 1011 = Pre-scale is 512; Max time-out period is 512 * 64 * WWDT_CLK. 1100 = Pre-scale is 768; Max time-out period is 768 * 64 * WWDT_CLK.

		1101 = Pre-scale is 1024; Max time-out period is 1024 * 64 * WWDT_CLK. 1110 = Pre-scale is 1536; Max time-out period is 1536 * 64 * WWDT_CLK. 1111 = Pre-scale is 2048; Max time-out period is 2048 * 64 * WWDT_CLK.
[7:2]	**Reserved**	Reserved.
[1]	**INTEN**	**WWDT Interrupt Enable Bit** If this bit is enabled, the WWDT counter compare match interrupt signal is generated and inform to CPU. 0 = WWDT counter compare match interrupt Disabled. 1 = WWDT counter compare match interrupt Enabled.
[0]	**WWDTEN**	**WWDT Enable Bit** 0 = WWDT counter is stopped. 1 = Enable WWDT counter starts counting.

WWDT Status Register (WWDT_STATUS)

Register	Offset	R/W	Description	Reset Value
WWDT_STATUS	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**WWDTRF**	**WWDTIF**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WWDTRF**	**WWDT Timer-out Reset Flag** This bit indicates the system has been reset by WWDT time-out reset or not. 0 = WWDT time-out reset did not occur. 1 = WWDT time-out reset occurred. **Note:** This bit is cleared by writing 1 to it.
[0]	**WWDTIF**	**WWDT Compare Match Interrupt Flag** This bit indicates the interrupt flag status of WWDT while WWDT counter value matches CMPDAT (WWDT_CTL[21:16]). 0 = No effect. 1 = WWDT counter value matches CMPDAT. **Note:** This bit is cleared by writing 1 to it.

WWDT Counter Value Register (WWDT_CNT)

Register	Offset	R/W	Description	Reset Value
WWDT_CNT	WWDT_BA+0x0C	R	WWDT Counter Value Register	0x0000_003F

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CNTDAT**					

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**CNTDAT**	**WWDT Counter Value** CNTDAT will be updated continuously to monitor 6-bit WWDT down counter value.

6.16 Real Time Clock (RTC)

6.16.1 Overview

The Real Time Clock (RTC) controller provides the real time and calendar message. The RTC offers programmable time tick and alarm match interrupts. The data format of time and calendar messages are expressed in BCD format. A digital frequency compensation feature is available to compensate external crystal oscillator frequency accuracy.

6.16.2 Features

- Supports external power pin V_{BAT}.
- Supports real time counter in RTC_TIME (hour, minute, second) and calendar counter in RTC_CAL (year, month, day) for RTC time and calendar check.
- Supports alarm time (hour, minute, second) and calendar (year, month, day) settings in RTC_TALM and RTC_CALM.
- Supports alarm time (hour, minute, second) and calendar (year, month, day) mask enable in RTC_TAMSK and RTC_CAMSK.
- Selectable 12-hour or 24-hour time scale in RTC_CLKFMT register.
- Optional support 1/128 second HZCNT in RTC_TIME and RTC_TALM.
- Supports Leap Year indication in RTC_LEAPYEAR register.
- Supports Day of the Week counter in RTC_WEEKDAY register.
- Frequency of RTC clock source compensate by RTC_FREQADJ register.
- All time and calendar message expressed in BCD format.
- Supports periodic RTC Time Tick interrupt with 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second.
- Supports RTC Time Tick and Alarm Match interrupt.
- Supports 1 Hz clock output.
- Supports chip wake-up from Idle or Power-down mode while a RTC interrupt signal is generated.
- Supports Daylight Saving Time software control in RTC_DSTCTL.
- Supports up to 3 pairs dynamic loop tamper pin or 6 individual tamper pins.
- Supports up to 80 bytes spare registers and tamper pins detection to clear the content of these spare registers.
- Supports Flash mass erase operate will also clear the spare registers content.

Section	Sub-Section	M463	M467
Functional Description	6.16.5.12 Spare Registers and Tamper Detector for M463 Series	●	-
	6.16.5.13 Spare Registers and Tamper Detector for M467 Series	-	●
Register	RTC Spare Register 5~19 (RTC_SPR5~19)	-	●

Description	RTC Tamper Pin Control Register (RTC_TAMPCTL) bit field [31], [23], [15], [7:3], [1:0]	-	●
	RTC Tamper Dynamic Seed Register (RTC_TAMPSEED)	-	●

Table 6.16-1 RTC Features Comparison Table at Different Chip

6.16.3 Block Diagram

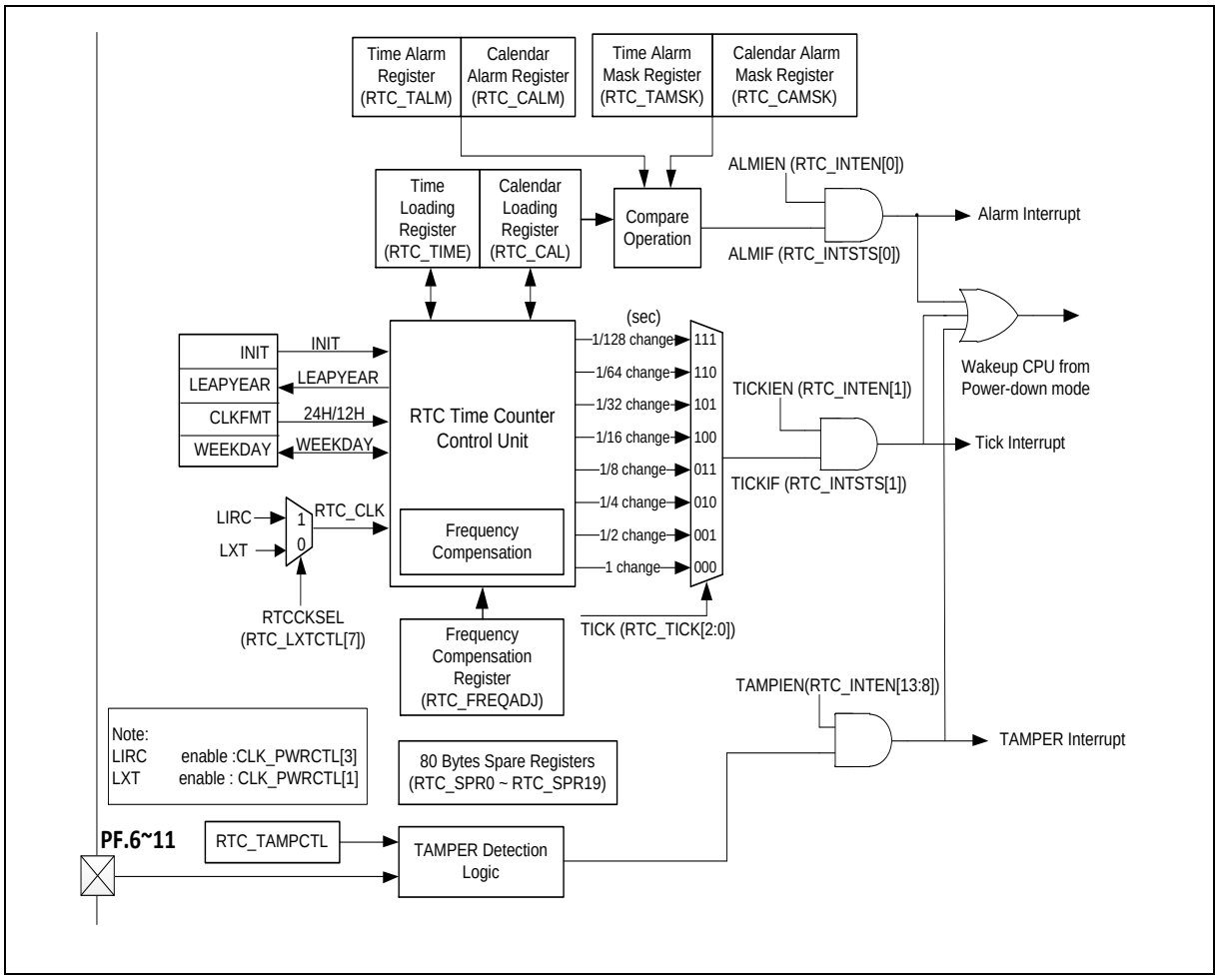


Figure 6.16-1 RTC Block Diagram

6.16.4 Basic Configuration

- Clock Source Configuration
  - The RTC controller clock source is enabled by RTCCKEN (CLK_APBCLK0[1])
  - The RTC Time Counter source can be selected as
    - Internal LIRC if RTCCKSEL (RTC_LXTCTL[7]) = 1
    - External LXT if RTCCKSEL (RTC_LXTCTL[7]) = 0

6.16.5 Functional Description

6.16.5.1 RTC Initiation

When a RTC block is powered on, RTC is at reset state. User has to write a number 0xa5eb1357 to RTC initial register INIT (RTC_INIT[31:0]) to make RTC leaving reset state. Once the INIT register is written as 0xa5eb1357, the RTC will be in normal active state permanently. User can read Active bit (RTC_INIT[0]) to check the RTC is at normal active state or reset state.

6.16.5.2 RTC Control Registers Access Attribute

The RTC control registers access attributes are shown in Table 6.16-2 .

Register	Active = 0	Active = 1
RTC_INIT	R/W	R/W
RTC_FREQADJ	R/W	R/W
RTC_TIME	Not available	R/W
RTC_CAL	Not available	R/W
RTC_CLKFMT	Not available	R/W
RTC_WEEKDAY	Not available	R/W
RTC_TALM	Not available	R/W
RTC_CALM	Not available	R/W
RTC_LEAPYEAR	Not available	R
RTC_INTEN	R/W	R/W
RTC_INTSTS	R/W	R/W
RTC_TICK	Not available	R/W
RTC_TAMSK	Not available	R/W
RTC_CAMSK	Not available	R/W
RTC_SPRCTL	R/W	R/W
RTC_SPRx	R/W	R/W
RTC_LXTCTL	R/W	R/W
RTC_GPIOCTL0	R/W	R/W
RTC_GPIOCTL1	R/W	R/W
RTC_DSTCTL	Not available	R/W
RTC_TAMPCTL	R/W	R/W

RTC_TAMPSEED	R/W	R/W

Table 6.16-2 RTC Read/Write Enable

6.16.5.3 Frequency Compensation

The frequency compensation circuit supports dynamic compensation to adjust compensation value without the compensation circuit reset. The enable bit for dynamic compensation is DCOMPEN (RTC_CLKFMT[16]).

If dynamic compensation is enabled, the minimal interval to continuous writing RTC_FREQADJ is 0.024 seconds, and FCRBUSY(RTC_FREQADJ[31]) flag is used to indicate that new register write operation is prohibited. And the compensate value will load into compensation circuit after 0.024~1.016 seconds when write to RTC_FREQADJ.

The RTC_FREQADJ register allows user to make digital compensation to a clock input. Please follow the example and formula below to write the actual frequency of 32k crystal to RTC_FREQADJ register. Following are the compensation examples for higher or lower than 32768 Hz.

Example 1:

Frequency counter measurement: 32773.65 Hz

Integer Part: 32773 => Setting INTEGER (RTC_FREQADJ[12:8]) = 0x15

Fraction Part: 0.65 X 64 = 41.6 (0x2A) => Setting FRACTION (RTC_FREQADJ[5:0]) = 0x2A

Example 2:

Frequency counter measurement: 32763.25 Hz

Integer part: 32763 => Setting INTEGER (RTC_FREQADJ[12:8]) = 0x0B

Fraction part: 0.25 X 64 = 16 (0x10) => Setting FRACTION (RTC_FREQADJ[5:0]) = 0x10

Note: The value of RTC_FREQADJ register will be the default value (0x0000_1000) while the compensation is not executed. User can utilize a frequency counter to measure RTC clock source via clock output function in manufacturing. Meanwhile, user can use clock output function to check the result of RTC frequency compensation.

6.16.5.4 Time and Calendar Counter

RTC_TIME and RTC_CAL are used to load the real time and calendar. RTC_TALM and RTC_CALM are used for setup alarm time and calendar.

6.16.5.5 12/24 Hour Time Scale Selection

The 12/24 hour time scale selection depends on 24HEN (RTC_CLKFMT[0]).

When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication, if RTC_TIME[21] is 1, it indicates PM time message and RTC_TIME[21] is 0 indicates AM time message.

24-Hour Time Scale (24HEN = 1)		**12-Hour Time Scale (PM Time + 0x20) (24HEN = 0)**	
0x00 (AM12)	0x12 (PM12)	0x12 (AM12)	0x32 (PM12)
0x01 (AM01)	0x13 (PM01)	0x01 (AM01)	0x21 (PM01)
0x02 (AM02)	0x14 (PM02)	0x02 (AM02)	0x22 (PM02)

0x03 (AM03)	0x15 (PM03)	0x03 (AM03)	0x23 (PM03)
0x04 (AM04)	0x16 (PM04)	0x04 (AM04)	0x24 (PM04)
0x05 (AM05)	0x17 (PM05)	0x05 (AM05)	0x25 (PM05)
0x06 (AM06)	0x18 (PM06)	0x06 (AM06)	0x26 (PM06)
0x07 (AM07)	0x19 (PM07)	0x07 (AM07)	0x27 (PM07)
0x08 (AM08)	0x20 (PM08)	0x08 (AM08)	0x28 (PM08)
0x09 (AM09)	0x21 (PM09)	0x09 (AM09)	0x29 (PM09)
0x10 (AM10)	0x22 (PM10)	0x10 (AM10)	0x30 (PM10)
0x11 (AM11)	0x23 (PM11)	0x11 (AM11)	0x31 (PM11)
Note: The hour value is written Into RTC_TIME[21:16]. Messages are expressed In BCD format.			

Table 6.16-312/24 Hour Time Scale Selection

6.16.5.6 Day of the Week Counter

The RTC controller provides day of week in WEEKDAY (RTC_WEEKDAY[2:0]) bits. The value is defined from 0 to 6 to represent Sunday to Saturday respectively.

6.16.5.7 Periodic Time Tick Interrupt

The Periodic Time Tick interrupt has 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second that are selected by TICK (RTC_TICK[2:0]) bits. When Periodic Time Tick interrupt is enabled by setting TICKIEN (RTC_INTEN[1]) to 1, the Periodic Time Tick interrupt is requested periodically in the period selected by RTC_TICK[2:0] settings.

6.16.5.8 Alarm Interrupt

When the real time and calendar message in RTC_TIME and RTC_CAL registers are equal to alarm time and calendar values in RTC_TALM and RTC_CALM registers, the RTC alarm interrupt flag ALMIF (RTC_INTSTS[0]) is set to 1 and the RTC alarm interrupt signal assert if the alarm interrupt enable ALMIEN (RTC_INTEN[0]) is enabled.

The RTC controller provides Time Alarm Mask Register (RTC_TAMSK) and Calendar Alarm Mask Register (RTC_CAMSK) to mask the specified digit and generate periodic interrupt without changing the alarm match condition in RTC_TALM and RTC_CALM registers in each alarm interrupt service routine.

6.16.5.9 Daylight Saving Time

The RTC controller also provides RTC_DSTCTL register to store the control settings of daylight saving time application. User can read RTC_DSTCTL value to check current RTC date/time counter runs in daylight saving time mode or normal mode.

6.16.5.10 1 Hz clock output

The RTC controller provides 1Hz clock output to CLKO function pin. User can set CLK1HZEN (CLK_CLKOCTL[6]) to 1 and enable RTC, 1Hz clock will output to CLKO function pin.

6.16.5.11 Application Note

1. All data in RTC_TALM, RTC_CALM, RTC_TIME and RTC_CAL registers are all expressed in BCD format.
2. User has to make sure that the loaded values are reasonable. For example, Load RTC_CAL as 201a (year), 13 (month), 00 (day), or RTC_CAL does not match with RTC_WEEKDAY, etc.
3. In RTC_CAL and RTC_CALM, only 2 BCD digits are used to express “year”. The 2 BCD digits of xy means 20xy, rather than 19xy or 21xy.
4. Example of 12-Hour Time Setting
   If current RTC time is PM12:59:30 in 12-Hour Time Scale mode, the RTC_TIME setting as:
   1) HOUR:
      RTC_TIME[21:16]: 0x32 (0x12+0x20) combined by TENHR (RTC_TIME[21:20]) is 0x3, HR (RTC_TIME[19:16]) is 0x2.
   2) MIN:
      RTC_TIME[14:8]: 0x59 combined by TENMIN (RTC_TIME[14:12]) is 0x5, MIN (RTC_TIME[11:8]) is 0x9.
   3) SEC:
      RTC_TIME[6:0]: 0x30 combined by TENSEC (RTC_TIME[6:4]) is 0x3, SEC (RTC_TIME[3:0]) is 0x0.
5. Table 6.16-4 shows registers value after both core power and battery power are first powered on.

Register	Reset State
RTC_INIT	0
RTC_FREQADJ	0x00001000 (no compensation)
RTC_CAL	15/8/8 (year/month/day)
RTC_TIME	00:00:00 (hour : minute : second)
RTC_CALM	00/00/00 (year/month/day)
RTC_TALM	00:00:00 (hour : minute : second)
RTC_CLKFMT	1 (24-hour mode)
RTC_WEEKDAY	6 (Saturday)
RTC_INTEN	0
RTC_INTSTS	0
RTC_LEAPYEAR	0
RTC_TICK	0
RTC_DSTCTL	0

Table 6.16-4 Register Value After Powered On

6.16.5.12 Spare Registers and Tamper Detector for M463 Series

The RTC module is equipped with 20 bytes spare registers to store user’s important information. These spare registers are located in RTC domain. User needs to enable SPRRWEN (RTC_SPRCTL[2]) before writing one of 5 spare registers (RTC_SPR0 ~ RTC_SPR4).

When the transition condition defined in RTC_TAMPCTL is detected, tamper detected interrupt flag TAMPxIF (RTC_INTSTS[13:8]) will be generated. Meanwhile, the 20 bytes spare registers (RTC_SPR0 ~ RTC_SPR4) content will be cleared automatically by hardware to prevent the security data be disclosure and current RTC time and calendar will be loaded to RTC_TAMPTIME and RTC_TAMPCAL registers. These values can only be updated again when all TAMPxIF are cleared to 0. And if TAMPxIF is set to 1, the interrupt is generated to NVIC if the tamper detect interrupt enable RTC_TAMPxIEN (RTC_INTEN[13:8]) is actived.

After Flash mass erase operation, these 20 bytes spare registers content will be cleared automatically.

RTC supports 6 individual tamper pin. The TAMPxLV (RTC_TAMPCTL[4x+9], x=0, 1, ...,5) depends on level attribute of TAMPERx pin for individual tamper detection.

Static Tamper Programming Sequence Example:

Enable Tamper Programming Sequence:

(1) Clean the TAMPxIF (RTC_INTSTS[x+8], x=0, 1, …, 5)

(2) Set TAMPxLV (RTC_TAMPCTL[4x+9], x=0, 1, ...,5) and TAMPxDBEN (RTC_TAMPCTL[4x+10], x=0, 1, ...,5).

(3) Enable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5).

Disable or Re-configure Tamper Programming Sequence:

(1) Disable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5).

(2) Wait for 3 RTC CLK.

(3) Re-configure TAMPxLV (RTC_TAMPCTL[4x+9], x=0, 1, ...,5) and TAMPxDBEN (RTC_TAMPCTL[4x+10], x=0, 1, ...,5)..

(4) Enable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5)

6.16.5.13 Spare Registers and Tamper Detector for M467 Series

The RTC module is equipped with 80 bytes spare registers to store user's important information. These spare registers are located in RTC domain, user needs to enable SPRRWEN (RTC_SPRCTL[2]) before writing one of 20 spare registers (RTC_SPR0 ~ RTC_SPR19).

When the transition condition defined in RTC_TAMPCTL is detected, tamper detected interrupt flag TAMPxIF (RTC_INTSTS[13:8]) will be generated. Meanwhile, the 80 bytes spare registers (RTC_SPR0 ~ RTC_SPR19) content will be cleared automatically by hardware to prevent the security data be disclosure and current RTC time and calendar will be loaded to RTC_TAMPTIME and RTC_TAMPCAL registers, these values only can be update again when all TAMPxIF are cleared to 0. And if TAMPxIF is set to 1, the interrupt is generated to NVIC if the tamper detect interrupt enable RTC_TAMPxIEN (RTC_INTEN[13:8]) is actived.

After Flash mass erase operation, these 80 bytes spare registers content will be cleared automatically.

RTC support 3 pair dynamic loop tamper pin or 6 individual tamper pin. The DYNRATE (RTC_TAMPCTL[7:5]) determine a reference pattern bit duration and it is shown in Figure 6.16-2. In Table 6.16-5, setting DYNSRC (RTC_TAMPCTL[3]) can select detect signal is new value which generated by hardware random value generator, previous random value or user defined SEED value (RTC_TAMPSEED[31:0]) for dynamic loop tamper pin. Set DYN1ISS (RTC_TAMPCTL[0]) 1 can be select Pair1 detection source is from tamper 0, and DYN12SS (RTC_TAMPCTL[1]) 1 can be select Pair2 detection source is also from tamper 0. The TAMPxLV (RTC_TAMPCTL[4x+9], x=0, 1, ...,5) depend on level attribute of TAMPERx pin for individual tamper detection. The tamper control effection is shown in Table 6.16-6 and Table 6.16-7.

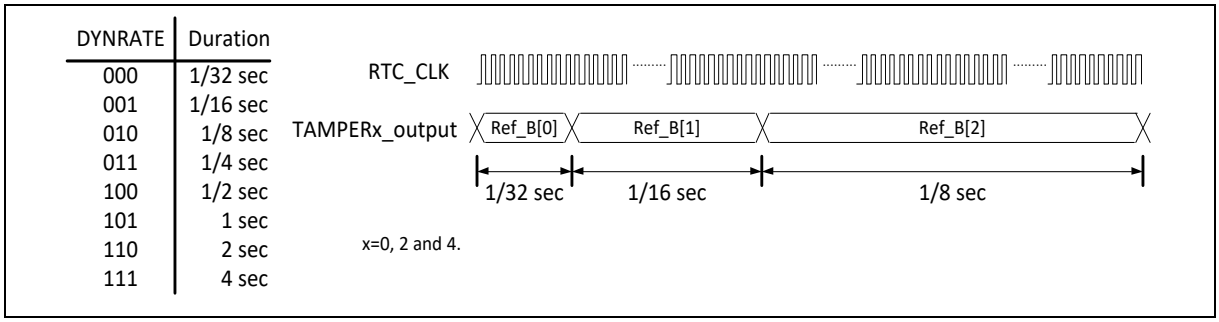


Figure 6.16-2 Dynamic Rate Definition

DYNPRxEN	DYNSRC	The Detect Signal Description
0	-	The TAMPERn is static tamper detection (n=0 ~ 5)
1	0	Generating new random value as reference pattern when the value runs out
1	1	Repeated user defined SEED value as reference pattern when the value runs out

Table 6.16-5 Dynamic Pattern Source Selection

Tamper Configuration			Tamper Control Bit Effect					
DYNPR0EN	TAMPER0EN	TAMPER1EN	TAMP0LV	TAMP1LV	TAMP0DBEN	TAMP1DBEN	TAMPER 0	TAMPER 1
0	0	0	-	-	-	-	-	-
0	0	1	-	V	-	V	-	Static
0	1	0	V	-	V	-	Static	-
0	1	1	V	V	V	V	Static	Static
1	-	0	-	-	-	-	Dynamic Out	-
1	-	1	-	-	-	-	Dynamic Out	Dynamic In

Table 6.16-6 Tamper Control Bit Effection for Pair 0

Tamper Configuration				Tamper Control Bit Effect					
DYNPRxEN	DYNxISS	TAMPERnEN	TAMPEmREN	TAMPnLV	TAMPmLV	TAMPnDBEN	TAMPmDBEN	TAMPER n	TAMPER m
0	-	0	0	-	-	-	-	-	-
0	-	0	1	-	V	-	V	-	Static
0	-	1	0	V	-	V	-	Static	-
0	-	1	1	V	V	V	V	Static	Static
1	0	-	0	-	-	-	-	Dynamic Out	-
1	0	-	1	-	-	-	-	Dynamic Out	Dynamic In
1	1	0	1	-	-	-	-	-	Dynamic Input form

									TAMPER 0
1	1	1	1	V	V	-	-	Static	Dynamic Input form TAMPER 0
x= 1 and 2		n= 2 and 4, m= 3 and 5							

Table 6.16-7 Tamper Control Bit Effection for Pair 1 and 2

Static Tamper Programming Sequence Example:

Enable Tamper Programming Sequence:

(1) Clean the TAMPxIF (RTC_INTSTS[x+8], x=0, 1, …, 5)

(2) Set TAMPxLV (RTC_TAMPCTL[4x+9], x=0, 1, ...,5) and TAMPxDBEN (RTC_TAMPCTL[4x+10], x=0, 1, ...,5).

(3) Enable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5).

Disable or Re-configure Tamper Programming Sequence:

(1) Disable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5).

(2) Wait for 3 RTC CLK.

(3) Re-configure TAMPxLV (RTC_TAMPCTL[4x+9], x=0, 1, ...,5) and TAMPxDBEN (RTC_TAMPCTL[4x+10], x=0, 1, ...,5)..

(4) Enable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5)

Dynamic Tamper Programming Sequence Example:

Enable Tamper Programming Sequence:

(1) Clean the TAMPxIF (RTC_INTSTS[x+8], x=0, 1, …, 5)

(2) Fill the SEED (RTC_TAMPSEED[31:0]).

(3) Set DYNSRC (RTC_TAMPCTL[3]), DYNxISS (RTC_TAMPCTL[x], x=0, 1) and DYNRATE (RTC_TAMPCTL[7:5]).

(4) Enable DYNPRxEN (RTC_TAMPCTL[8x+16], x=0, 1, 2).

(5) Enable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5).

(6) Set SEEDRLD (RTC_TAMPCTL[4]).

Disable or Re-confingure Tamper Programming Sequence:

(1) Disable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5).

(2) Wait for 3 RTC CLK.

(3) Reconfigure the tamper setting

(4) Enable TAMPxEN (RTC_TAMPCTL[4x+8], x=0, 1, ...,5).

6.16.5.14 Backup Domain GPIO Function

When PF.4/X32_OUT and PF.5/X32_IN pins are not used as low speed 32K oscillator function or PF.6~11 are not used for TAPMER function, they can be used as GPIO pin function. The IOCTLSEL (RTC_LXTCTL[8]) is used to select the PF.4~11 pins are controlled by RTC_GPIOCTL0/1 registers by V_{BAT} power domain or relative GPIO control registers by GPIO module. Figure 6.16-3 shows backup I/O control diagram.

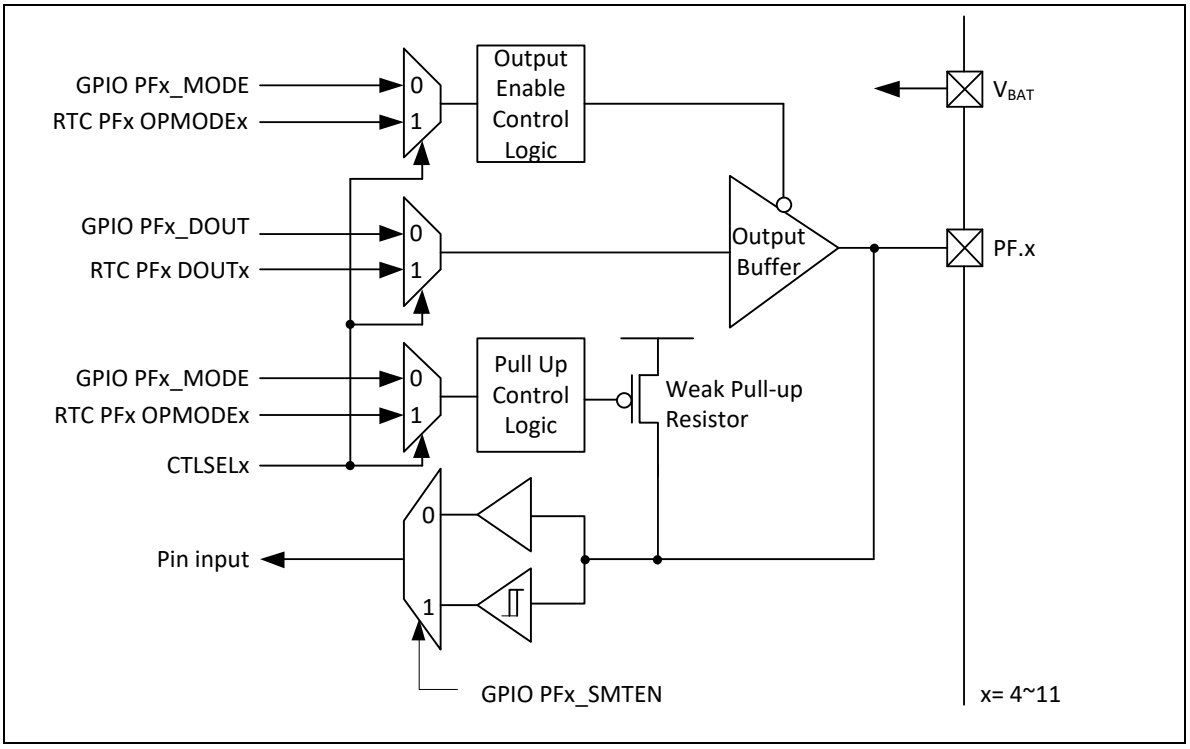


Figure 6.16-3 Backup I/O Control Diagram

6.16.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
RTC Base Address: **RTC_BA = 0x4004_1000**				
RTC_INIT	RTC_BA+0x00	R/W	RTC Initiation Register	0x0000_0000
RTC_FREQADJ	RTC_BA+0x08	R/W	RTC Frequency Compensation Register	0x0000_1000
RTC_TIME	RTC_BA+0x0C	R/W	RTC Time Loading Register	0x0000_0000
RTC_CAL	RTC_BA+0x10	R/W	RTC Calendar Loading Register	0x0015_0808
RTC_CLKFMT	RTC_BA+0x14	R/W	RTC Time Scale Selection Register	0x0000_0001
RTC_WEEKDAY	RTC_BA+0x18	R/W	RTC Day of the Week Register	0x0000_0006
RTC_TALM	RTC_BA+0x1C	R/W	RTC Time Alarm Register	0x0000_0000
RTC_CALM	RTC_BA+0x20	R/W	RTC Calendar Alarm Register	0x0000_0000
RTC_LEAPYEAR	RTC_BA+0x24	R	RTC Leap Year Indicator Register	0x0000_0000
RTC_INTEN	RTC_BA+0x28	R/W	RTC Interrupt Enable Register	0x0000_0000
RTC_INTSTS	RTC_BA+0x2C	R/W	RTC Interrupt Status Register	0x0000_0000
RTC_TICK	RTC_BA+0x30	R/W	RTC Time Tick Register	0x0000_0000
RTC_TAMSK	RTC_BA+0x34	R/W	RTC Time Alarm Mask Register	0x0000_0000
RTC_CAMSK	RTC_BA+0x38	R/W	RTC Calendar Alarm Mask Register	0x0000_0000
RTC_SPRCTL	RTC_BA+0x3C	R/W	RTC Spare Functional Control Register	0x0000_0000
RTC_SPR0	RTC_BA+0x40	R/W	RTC Spare Register 0	0x0000_0000
RTC_SPR1	RTC_BA+0x44	R/W	RTC Spare Register 1	0x0000_0000
RTC_SPR2	RTC_BA+0x48	R/W	RTC Spare Register 2	0x0000_0000
RTC_SPR3	RTC_BA+0x4C	R/W	RTC Spare Register 3	0x0000_0000
RTC_SPR4	RTC_BA+0x50	R/W	RTC Spare Register 4	0x0000_0000
RTC_SPR5	RTC_BA+0x54	R/W	RTC Spare Register 5	0x0000_0000
RTC_SPR6	RTC_BA+0x58	R/W	RTC Spare Register 6	0x0000_0000
RTC_SPR7	RTC_BA+0x5C	R/W	RTC Spare Register 7	0x0000_0000
RTC_SPR8	RTC_BA+0x60	R/W	RTC Spare Register 8	0x0000_0000
RTC_SPR9	RTC_BA+0x64	R/W	RTC Spare Register 9	0x0000_0000
RTC_SPR10	RTC_BA+0x68	R/W	RTC Spare Register 10	0x0000_0000
RTC_SPR11	RTC_BA+0x6C	R/W	RTC Spare Register 11	0x0000_0000

RTC_SPR12	RTC_BA+0x70	R/W	RTC Spare Register 12	0x0000_0000
RTC_SPR13	RTC_BA+0x74	R/W	RTC Spare Register 13	0x0000_0000
RTC_SPR14	RTC_BA+0x78	R/W	RTC Spare Register 14	0x0000_0000
RTC_SPR15	RTC_BA+0x7C	R/W	RTC Spare Register 15	0x0000_0000
RTC_SPR16	RTC_BA+0x80	R/W	RTC Spare Register 16	0x0000_0000
RTC_SPR17	RTC_BA+0x84	R/W	RTC Spare Register 17	0x0000_0000
RTC_SPR18	RTC_BA+0x88	R/W	RTC Spare Register 18	0x0000_0000
RTC_SPR19	RTC_BA+0x8C	R/W	RTC Spare Register 19	0x0000_0000
RTC_LXTCTL	RTC_BA+0x100	R/W	RTC 32.768 kHz Oscillator Control Register	0xXXXX_XX0E
RTC_GPIOCTL0	RTC_BA+0x104	R/W	RTC GPIO Control 0 Register	0x0000_0000
RTC_GPIOCTL1	RTC_BA+0x108	R/W	RTC GPIO Control 1 Register	0x0000_0000
RTC_DSTCTL	RTC_BA+0x110	R/W	RTC Daylight Saving Time Control Register	0x0000_0000
RTC_TAMPCTL	RTC_BA+0x120	R/W	RTC Tamper Pin Control Register	0x0000_0000
RTC_TAMPSEED	RTC_BA+0x128	R/W	RTC Tamper Dynamic Seed Register	0x0000_0000
RTC_TAMPTIME	RTC_BA+0x130	R	RTC Tamper Time Register	0x0000_0000
RTC_TAMPCAL	RTC_BA+0x134	R	RTC Tamper Calendar Register	0x0000_0000

6.16.7 Register Description

RTC Initiation Register (RTC_INIT)

Register	Offset	R/W	Description	Reset Value
RTC_INIT	RTC_BA+0x00	R/W	RTC Initiation Register	0x0000_0000

31	30	29	28	27	26	25	24
INIT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
INIT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
INIT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INIT							**ACTIVE**

Bits	Description	
[31:1]	**INIT**	**RTC Initiation (Write Only)** When RTC block is powered on, RTC is at reset state. User has to write a number (0x a5eb1357) to INIT to make RTC leave reset state. Once the INIT is written as 0xa5eb1357, the RTC will be in un-reset state permanently. The INIT is a write-only field and read value will be always 0.
[0]	**ACTIVE**	**RTC Active Status (Read Only)** 0 = RTC is at reset state. 1 = RTC is at normal active state.

RTC Frequency Compensation Register (RTC_FREQADJ)

Register	Offset	R/W	Description	Reset Value
RTC_FREQADJ	RTC_BA+0x08	R/W	RTC Frequency Compensation Register	0x0000_1000

31	30	29	28	27	26	25	24
FCRBUSY	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			INTEGER				
7	6	5	4	3	2	1	0
Reserved		FRACTION					

Bits	Description	
[31]	**FCRBUSY**	**Frequency Compensation Register Write Operation Busy (Read Only)** 0 = The new register write operation is acceptable. 1 = The last write operation is in progress and new register write operation prohibited. **Note:** This bit is only used when DCOMPEN (RTC_CLKFMT[16]) enabled.
[30:13]	**Reserved**	Reserved.
[12:8]	**INTEGER**	**Integer Part** 00000 = Integer part of detected value is 32752. 00001 = Integer part of detected value is 32753. 00010 = Integer part of detected value is 32754. 00011 = Integer part of detected value is 32755. 00100 = Integer part of detected value is 32756. 00101 = Integer part of detected value is 32757. 00110 = Integer part of detected value is 32758. 00111 = Integer part of detected value is 32759. 01000 = Integer part of detected value is 32760. 01001 = Integer part of detected value is 32761. 01010 = Integer part of detected value is 32762. 01011 = Integer part of detected value is 32763. 01100 = Integer part of detected value is 32764. 01101 = Integer part of detected value is 32765. 01110 = Integer part of detected value is 32766. 01111 = Integer part of detected value is 32767. 10000 = Integer part of detected value is 32768. 10001 = Integer part of detected value is 32769. 10010 = Integer part of detected value is 32770. 10011 = Integer part of detected value is 32771. 10100 = Integer part of detected value is 32772. 10101 = Integer part of detected value is 32773.

		10110 = Integer part of detected value is 32774. 10111 = Integer part of detected value is 32775. 11000 = Integer part of detected value is 32776. 11001 = Integer part of detected value is 32777. 11010 = Integer part of detected value is 32778. 11011 = Integer part of detected value is 32779. 11100 = Integer part of detected value is 32780. 11101 = Integer part of detected value is 32781. 11110 = Integer part of detected value is 32782. 11111 = Integer part of detected value is 32783.
[7:6]	**Reserved**	Reserved.
[5:0]	**FRACTION**	**Fraction Part** Formula: FRACTION = (fraction part of detected value) X 64. **Note:** Digit in FCR must be expressed as hexadecimal number.

Note: If the dynamic compensation is not enabled, i.e. DCOMPEN (RTC_CLKFMT[16]) = 0, the FREQADJ counter will be reset for starting to compensate when writing RTC_FREQADJ. It will be impacted if the RTC counter is restarted.

RTC Time Loading Register (RTC_TIME)

Register	Offset	R/W	Description	Reset Value
RTC_TIME	RTC_BA+0x0C	R/W	RTC Time Loading Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	HZCNT						
23	22	21	20	19	18	17	16
Reserved		TENHR		HR			
15	14	13	12	11	10	9	8
Reserved	TENMIN			MIN			
7	6	5	4	3	2	1	0
Reserved	TENSEC			SEC			

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**HZCNT**	Index of sub-second counter (0x00~0x7F)
[23:22]	**Reserved**	Reserved.
[21:20]	**TENHR**	10-Hour Time Digit (0~2) When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication (If RTC_TIME[21] is 1, it indicates PM time message.)
[19:16]	**HR**	1-Hour Time Digit (0~9)
[15]	**Reserved**	Reserved.
[14:12]	**TENMIN**	10-Min Time Digit (0~5)
[11:8]	**MIN**	1-Min Time Digit (0~9)
[7]	**Reserved**	Reserved.
[6:4]	**TENSEC**	10-Sec Time Digit (0~5)
[3:0]	**SEC**	1-Sec Time Digit (0~9)

Note:

1. RTC_TIME is a BCD digit counter (except HZCNT) and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.
3. The FREQADJ counter will be reset for starting to compensate when writing RTC_TIME, RTC_CAL, RTC_WEEKDAY. It will impact if the RTC counter will be restarted.
4. HZCNT will set to “0” when writing RTC_TIME in RTC_CLKFMT[8] = 0. User can write value 0x00 ~ 0x7F in RTC_CLKFMT[8] = 1 .

RTC Calendar Loading Register (RTC_CAL)

Register	Offset	R/W	Description	Reset Value
RTC_CAL	RTC_BA+0x10	R/W	RTC Calendar Loading Register	0x0015_0808

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TENYEAR				**YEAR**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**TENMON**	**MON**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**TENDAY**		**DAY**			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**TENYEAR**	10-Year Calendar Digit (0~9)
[19:16]	**YEAR**	1-Year Calendar Digit (0~9)
[15:13]	**Reserved**	Reserved.
[12]	**TENMON**	10-Month Calendar Digit (0~1)
[11:8]	**MON**	1-Month Calendar Digit (0~9)
[7:6]	**Reserved**	Reserved.
[5:4]	**TENDAY**	10-Day Calendar Digit (0~3)
[3:0]	**DAY**	1-Day Calendar Digit (0~9)

Note:

1. RTC_CAL is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.
3. The FREQADJ counter will be reset for starting to compensate when writing RTC_TIME, RTC_CAL, RTC_WEEKDAY. It will impact if the RTC counter will be restarted.

RTC Time Scale Selection Register (RTC_CLKFMT)

Register	Offset	R/W	Description	Reset Value
RTC_CLKFMT	RTC_BA+0x14	R/W	RTC Time Scale Selection Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**DCOMPEN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**HZCNTEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**24HEN**

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DCOMPEN**	**Dynamic Compensation Enable Bit** 0 = Dynamic Compensation Disabled. 1 = Dynamic Compensation Enabled.
[15:9]	**Reserved**	Reserved.
[8]	**HZCNTEN**	**Sub-second Counter Enable Bit** 0 = HZCNT disabled in RTC_TIME and RTC_TALM. 1 = HZCNT enabled in RTC_TIME and RTC_TALM.
[7:1]	**Reserved**	Reserved.
[0]	**24HEN**	**24-hour / 12-hour Time Scale Selection** Indicates that RTC_TIME and RTC_TALM are in 24-hour time scale or 12-hour time scale 0 = 12-hour time scale with AM and PM indication selected. 1 = 24-hour time scale selected.

RTC Day of the Week Register (RTC_WEEKDAY)

Register	Offset	R/W	Description	Reset Value
RTC_WEEKDAY	RTC_BA+0x18	R/W	RTC Day of the Week Register	0x0000_0006

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**WEEKDAY**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**WEEKDAY**	**Day of the Week Register** 000 = Sunday. 001 = Monday. 010 = Tuesday. 011 = Wednesday. 100 = Thursday. 101 = Friday. 110 = Saturday. 111 = Reserved.

Note: The FREQADJ counter will be reset for starting to compensate when writing RTC_TIME, RTC_CAL, RTC_WEEKDAY. It will impact if the RTC counter will be restarted.

RTC Time Alarm Register (RTC_TALM)

Register	Offset	R/W	Description	Reset Value
RTC_TALM	RTC_BA+0x1C	R/W	RTC Time Alarm Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	HZCNT						
23	22	21	20	19	18	17	16
Reserved		TENHR		HR			
15	14	13	12	11	10	9	8
Reserved	TENMIN			MIN			
7	6	5	4	3	2	1	0
Reserved	TENSEC			SEC			

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**HZCNT**	Index of sub-second counter (0x00~0x7F)
[21:20]	**TENHR**	10-Hour Time Digit of Alarm Setting (0~2) When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication (If RTC_TIME[21] is 1, it indicates PM time message.)
[19:16]	**HR**	1-Hour Time Digit of Alarm Setting (0~9)
[15]	**Reserved**	Reserved.
[14:12]	**TENMIN**	10-Min Time Digit of Alarm Setting (0~5)
[11:8]	**MIN**	1-Min Time Digit of Alarm Setting (0~9)
[7]	**Reserved**	Reserved.
[6:4]	**TENSEC**	10-Sec Time Digit of Alarm Setting (0~5)
[3:0]	**SEC**	1-Sec Time Digit of Alarm Setting (0~9)

Note:

1. RTC_TALM is a BCD digit counter, except HZCNT.
2. The reasonable value range is listed in the parenthesis.

RTC Calendar Alarm Register (RTC_CALM)

Register	Offset	R/W	Description	Reset Value
RTC_CALM	RTC_BA+0x20	R/W	RTC Calendar Alarm Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TENYEAR				**YEAR**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**TENMON**	**MON**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**TENDAY**		**DAY**			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**TENYEAR**	10-Year Calendar Digit of Alarm Setting (0~9)
[19:16]	**YEAR**	1-Year Calendar Digit of Alarm Setting (0~9)
[15:13]	**Reserved**	Reserved.
[12]	**TENMON**	10-Month Calendar Digit of Alarm Setting (0~1)
[11:8]	**MON**	1-Month Calendar Digit of Alarm Setting (0~9)
[7:6]	**Reserved**	Reserved.
[5:4]	**TENDAY**	10-Day Calendar Digit of Alarm Setting (0~3)
[3:0]	**DAY**	1-Day Calendar Digit of Alarm Setting (0~9)

Note:

1. RTC_CALM is a BCD digit counter.
2. The reasonable value range is listed in the parenthesis.

RTC Leap Year Indication Register (RTC_LEAPYEAR)

Register	Offset	R/W	Description	Reset Value
RTC_LEAPYEAR	RTC_BA+0x24	R	RTC Leap Year Indicator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							LEAPYEAR

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**LEAPYEAR**	**Leap Year Indication (Read Only)** 0 = This year is not a leap year. 1 = This year is leap year.

RTC Interrupt Enable Register (RTC_INTEN)

Register	Offset	R/W	Description	Reset Value
RTC_INTEN	RTC_BA+0x28	R/W	RTC Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		TAMP5IEN	TAMP4IEN	TAMP3IEN	TAMP2IEN	TAMP1IEN	TAMP0IEN
7	6	5	4	3	2	1	0
Reserved						TICKIEN	ALMIEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**TAMP5IEN**	**Tamper 5 or Pair 2 Interrupt Enable Bit** Set TAMP5IEN to 1 can also enable chip wake-up function when tamper 5 interrupt event is generated. 0 = Tamper 5 or Pair 2 interrupt Disabled. 1 = Tamper 5 or Pair 2 interrupt Enabled.
[12]	**TAMP4IEN**	**Tamper 4 Interrupt Enable Bit** Set TAMP4IEN to 1 can also enable chip wake-up function when tamper 4 interrupt event is generated. 0 = Tamper 4 interrupt Disabled. 1 = Tamper 4 interrupt Enabled.
[11]	**TAMP3IEN**	**Tamper 3 or Pair 1 Interrupt Enable Bit** Set TAMP3IEN to 1 can also enable chip wake-up function when tamper 3 interrupt event is generated. 0 = Tamper 3 or Pair 1 interrupt Disabled. 1 = Tamper 3 or Pair 1 interrupt Enabled.
[10]	**TAMP2IEN**	**Tamper 2 Interrupt Enable Bit** Set TAMP2IEN to 1 can also enable chip wake-up function when tamper 2 interrupt event is generated. 0 = Tamper 2 interrupt Disabled. 1 = Tamper 2 interrupt Enabled.
[9]	**TAMP1IEN**	**Tamper 1 or Pair 0 Interrupt Enable Bit** Set TAMP1IEN to 1 can also enable chip wake-up function when tamper 1 interrupt event is generated. 0 = Tamper 1 or Pair 0 interrupt Disabled. 1 = Tamper 1 or Pair 0 interrupt Enabled.
[8]	**TAMP0IEN**	**Tamper 0 Interrupt Enable Bit** Set TAMP0IEN to 1 can also enable chip wake-up function when tamper 0 interrupt event

		is generated. 0 = Tamper 0 interrupt Disabled. 1 = Tamper 0 interrupt Enabled.
[7:2]	**Reserved**	Reserved.
[1]	**TICKIEN**	**Time Tick Interrupt Enable Bit** Set TICKIEN to 1 can also enable chip wake-up function when RTC tick interrupt event is generated. 0 = RTC Time Tick interrupt Disabled. 1 = RTC Time Tick interrupt Enabled.
[0]	**ALMIEN**	**Alarm Interrupt Enable Bit** Set ALMIEN to 1 can also enable chip wake-up function when RTC alarm interrupt event is generated. 0 = RTC Alarm interrupt Disabled. 1 = RTC Alarm interrupt Enabled.

RTC Interrupt Status Register (RTC_INTSTS)

Register	Offset	R/W	Description	Reset Value
RTC_INTSTS	RTC_BA+0x2C	R/W	RTC Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		TAMP5IF	TAMP4IF	TAMP3IF	TAMP2IF	TAMP1IF	TAMP0IF
7	6	5	4	3	2	1	0
Reserved						TICKIF	ALMIF

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**TAMP5IF**	**Tamper 5 or Pair 2 Interrupt Flag** This bit is set when TAMPER5 detected level non-equal TAMP5LV (RTC_TAMPCTL[29]) or TAMPER4 and TAMPER5 disconnected during DYNPR2EN (RTC_TAMPCTL[31]) is activated or TAMPER0 and TAMPER5 disconnected during DYNPR2EN (RTC_TAMPCTL[31]) and DYN2ISS (RTC_TAMPCTL[1]) are activated. 0 = No Tamper 5 or Pair 2 interrupt flag is generated. 1 = Tamper 5 or Pair 2 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will generate again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values can loaded while a tamper event occurred.
[12]	**TAMP4IF**	**Tamper 4 Interrupt Flag** This bit is set when TAMPER4 detected level non-equal TAMP4LV (RTC_TAMPCTL[25]). 0 = No Tamper 4 interrupt flag is generated. 1 = Tamper 4 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will generate again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values can loaded while a tamper event occurred.
[11]	**TAMP3IF**	**Tamper 3 or Pair 1 Interrupt Flag** This bit is set when TAMPER3 detected level non-equal TAMP3LV (RTC_TAMPCTL[21]) or TAMPER2 and TAMPER3 disconnected during DYNPR1EN (RTC_TAMPCTL[23]) is activated or TAMPER0 and TAMPER3 disconnected during DYNPR1EN (RTC_TAMPCTL[23]) and DYN1ISS (RTC_TAMPCTL[0]) are activated. 0 = No Tamper 3 or Pair 1 interrupt flag is generated. 1 = Tamper 3 or Pair 1 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will generate again when Tamper setting condition is not

		restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values can loaded while a tamper event occurred.
[10]	**TAMP2IF**	**Tamper 2 Interrupt Flag** This bit is set when TAMPER2 detected level non-equal TAMP2LV (RTC_TAMPCTL[17]). 0 = No Tamper 2 interrupt flag is generated. 1 = Tamper 2 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will generate again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values can loaded while a tamper event occurred.
[9]	**TAMP1IF**	**Tamper 1 or Pair 0 Interrupt Flag** This bit is set when TAMPER1 detected level non-equal TAMP1LV (RTC_TAMPCTL[13]) or TAMPER0 and TAMPER1 disconnected during DYNPR0EN (RTC_TAMPCTL[15]) is activated. 0 = No Tamper 1 or Pair 0 interrupt flag is generated. 1 = Tamper 1 or Pair 0 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will generate again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values can loaded while a tamper event occurred.
[8]	**TAMP0IF**	**Tamper 0 Interrupt Flag** This bit is set when TAMPER0 detected level non-equal TAMP0LV (RTC_TAMPCTL[9]). 0 = No Tamper 0 interrupt flag is generated. 1 = Tamper 0 interrupt flag is generated. **Note 1:** Write 1 to clear this bit. **Note 2:** This interrupt flag will generate again when Tamper setting condition is not restoration. **Note 3:** Need to clear all TAMPxIF, after that, new RTC_TAMPTIME and RTC_TAMPCAL values can loaded while a tamper event occurred.
[7:2]	**Reserved**	Reserved.
[1]	**TICKIF**	**RTC Time Tick Interrupt Flag** 0 = Tick condition did not occur. 1 = Tick condition occurred. **Note:** Write 1 to clear this bit.
[0]	**ALMIF**	**RTC Alarm Interrupt Flag** 0 = Alarm condition is not matched. 1 = Alarm condition is matched. **Note:** Write 1 to clear this bit.

RTC Time Tick Register (RTC_TICK)

Register	Offset	R/W	Description	Reset Value
RTC_TICK	RTC_BA+0x30	R/W	RTC Time Tick Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					TICK		

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	TICK	**Time Tick Register** These bits are used to select RTC time tick period for Periodic Time Tick Interrupt request. 000 = Time tick is 1 second. 001 = Time tick is 1/2 second. 010 = Time tick is 1/4 second. 011 = Time tick is 1/8 second. 100 = Time tick is 1/16 second. 101 = Time tick is 1/32 second. 110 = Time tick is 1/64 second. 111 = Time tick is 1/128 second.

RTC Time Alarm MASK Register (RTC_TAMSK)

Register	Offset	R/W	Description	Reset Value
RTC_TAMSK	RTC_BA+0x34	R/W	RTC Time Alarm Mask Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		MTENHR	MHR	MTENMIN	MMIN	MTENSEC	MSEC

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**MTENHR**	Mask 10-Hour Time Digit of Alarm Setting (0~2)
[4]	**MHR**	Mask 1-Hour Time Digit of Alarm Setting (0~9)
[3]	**MTENMIN**	Mask 10-Min Time Digit of Alarm Setting (0~5)
[2]	**MMIN**	Mask 1-Min Time Digit of Alarm Setting (0~9)
[1]	**MTENSEC**	Mask 10-Sec Time Digit of Alarm Setting (0~5)
[0]	**MSEC**	Mask 1-Sec Time Digit of Alarm Setting (0~9)

Note:

1. RTC_TALM is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.
3. MTENHR/MHR base on 24 hour Time Scale.

RTC Calendar Alarm MASK Register (RTC_CAMSK)

Register	Offset	R/W	Description	Reset Value
RTC_CAMSK	RTC_BA+0x38	R/W	RTC Calendar Alarm Mask Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MTENYEAR**	**MYEAR**	**MTENMON**	**MMON**	**MTENDAY**	**MDAY**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**MTENYEAR**	Mask 10-Year Calendar Digit of Alarm Setting (0~9)
[4]	**MYEAR**	Mask 1-Year Calendar Digit of Alarm Setting (0~9)
[3]	**MTENMON**	Mask 10-Month Calendar Digit of Alarm Setting (0~1)
[2]	**MMON**	Mask 1-Month Calendar Digit of Alarm Setting (0~9)
[1]	**MTENDAY**	Mask 10-Day Calendar Digit of Alarm Setting (0~3)
[0]	**MDAY**	Mask 1-Day Calendar Digit of Alarm Setting (0~9)

Note:

1. RTC_CALM is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.

RTC Spare Functional Control Register (RTC_SPRCTL)

Register	Offset	R/W	Description	Reset Value
RTC_SPRCTL	RTC_BA+0x3C	R/W	RTC Spare Functional Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							KSTRIGEN
7	6	5	4	3	2	1	0
Reserved		SPRCSTS	Reserved		SPRRWEN	Reserved	

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**KSTRIGEN**	**Key Store Trigger Enable Bit** 0 = Tamper event is detected and to trigger Key Store Disabled. 1 = Tamper event is detected and to trigger Key Store Enabled.
[7:6]	**Reserved**	Reserved.
[5]	**SPRCSTS**	**SPR Clear Flag** This bit indicates if the RTC_SPR0 ~ RTC_SPR19 content is cleared when specify tamper event is detected. 0 = Spare register content is not cleared. 1 = Spare register content is cleared. **Note 1:** Write 1 to clear this bit. **Note 2:** This bit keeps 1 when RTC_INTSTS[13:8] is not equal to 0.
[4:3]	**Reserved**	Reserved.
[2]	**SPRRWEN**	**Spare Register Enable Bit** 0 = Spare register Disabled. 1 = Spare register Enabled. **Note:** When spare register is disabled, RTC_SPR0 ~ RTC_SPR19 cannot be accessed.
[1:0]	**Reserved**	Reserved.

RTC Spare Register (RTC_SPRx)

Register	Offset	R/W	Description	Reset Value
RTC_SPR0	RTC_BA+0x40	R/W	RTC Spare Register 0	0x0000_0000
RTC_SPR1	RTC_BA+0x44	R/W	RTC Spare Register 1	0x0000_0000
RTC_SPR2	RTC_BA+0x48	R/W	RTC Spare Register 2	0x0000_0000
RTC_SPR3	RTC_BA+0x4C	R/W	RTC Spare Register 3	0x0000_0000
RTC_SPR4	RTC_BA+0x50	R/W	RTC Spare Register 4	0x0000_0000
RTC_SPR5	RTC_BA+0x54	R/W	RTC Spare Register 5	0x0000_0000
RTC_SPR6	RTC_BA+0x58	R/W	RTC Spare Register 6	0x0000_0000
RTC_SPR7	RTC_BA+0x5C	R/W	RTC Spare Register 7	0x0000_0000
RTC_SPR8	RTC_BA+0x60	R/W	RTC Spare Register 8	0x0000_0000
RTC_SPR9	RTC_BA+0x64	R/W	RTC Spare Register 9	0x0000_0000
RTC_SPR10	RTC_BA+0x68	R/W	RTC Spare Register 10	0x0000_0000
RTC_SPR11	RTC_BA+0x6C	R/W	RTC Spare Register 11	0x0000_0000
RTC_SPR12	RTC_BA+0x70	R/W	RTC Spare Register 12	0x0000_0000
RTC_SPR13	RTC_BA+0x74	R/W	RTC Spare Register 13	0x0000_0000
RTC_SPR14	RTC_BA+0x78	R/W	RTC Spare Register 14	0x0000_0000
RTC_SPR15	RTC_BA+0x7C	R/W	RTC Spare Register 15	0x0000_0000
RTC_SPR16	RTC_BA+0x80	R/W	RTC Spare Register 16	0x0000_0000
RTC_SPR17	RTC_BA+0x84	R/W	RTC Spare Register 17	0x0000_0000
RTC_SPR18	RTC_BA+0x88	R/W	RTC Spare Register 18	0x0000_0000
RTC_SPR19	RTC_BA+0x8C	R/W	RTC Spare Register 19	0x0000_0000

31	30	29	28	27	26	25	24
SPARE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SPARE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPARE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPARE							

Bits	Description	
[31:0]	**SPARE**	**Spare Register** This field is used to store back-up information defined by user. This field will be cleared by hardware automatically in the following conditions, a tamper pin event is detected, or after Flash mass operation.

RTC 32K Oscillator Control Register (RTC_LXTCTL)

Register	Offset	R/W	Description	Reset Value
RTC_LXTCTL	RTC_BA+0x100	R/W	RTC 32.768 kHz Oscillator Control Register	0xXXXX_XX0E

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							IOCTLSEL
7	6	5	4	3	2	1	0
RTCCKSEL	Reserved			GAIN			Reserved

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**IOCTLSEL**	**I/O Pin Backup Control Selection** When low speed 32 kHz oscillator is disabled or TAMPxEN is disabled, PF.4 pin (X32_OUT pin), PF.5 pin (X32_IN pin) or PF.6~11 pin (TAMPERx pin) can be used as GPIO function. User can program IOCTLSEL to decide PF.4~11 I/O function is controlled by system power domain GPIO module or V_{BAT} power domain RTC_GPIOCTL0/1 control register. 0 = PF.4~11 pin I/O function is controlled by GPIO module. 1 = PF.4~11 pin I/O function is controlled by V_{BAT} power domain. **Note:** This bit field will be set to 1 by hardware automatically while the system entering Power-down mode if any writable RTC register has been written when RTCCKEN (CLK_APBCLK0[1]) enabled. And this bit field will keep 1 after the system wakes up, need to program it to 0 if want PF.4~11 I/O function is controlled by system power domain GPIO module.
[7]	**RTCCKSEL**	**RTC Clock Source Selection** 0 = Clock source from external low speed crystal oscillator (LXT). 1 = Clock source from internal low speed RC oscillator (LIRC).
[6:4]	**Reserved**	Reserved.
[3:1]	**GAIN**	**Oscillator Gain Option** User can select oscillator gain according to crystal external loading and operating temperature range. The larger gain value corresponding to stronger driving capability and higher power consumption. 001 = L1 mode. 010 = L2 mode. 011 = L3 mode. 100 = L4 mode. 101 = L5 mode. 110 = L6 mode. 111 = L7 mode (Default).

[0]	**Reserved**	Reserved.

RTC GPIO Control Register0 (RTC_GPIOCTL0)

Register	Offset	R/W	Description	Reset Value
RTC_GPIOCTL0	RTC_BA+0x104	R/W	RTC GPIO Control 0 Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		PUSEL3		DINOFF3	DOUT3	OPMODE3	
23	22	21	20	19	18	17	16
Reserved		PUSEL2		DINOFF2	DOUT2	OPMODE2	
15	14	13	12	11	10	9	8
Reserved		PUSEL1		DINOFF1	DOUT1	OPMODE1	
7	6	5	4	3	2	1	0
Reserved		PUSEL0		DINOFF0	DOUT0	OPMODE0	

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**PUSEL3**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.7 I/O pull-up or pull-down. 00 = PF.7 pull-up and pull-down Disabled. 01 = PF.7 pull-up Enabled. 10 = PF.7 pull-down Enabled. 11 = PF.7 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE3 is set as input tri-state and open-drain mode.
[27]	**DINOFF3**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.7 digital input path Enabled. 1 = PF.7 digital input path Disabled (digital input tied to low).
[26]	**DOUT3**	**I/O Output Data** 0 = PF.7 output low. 1 = PF.7 output high.
[25:24]	**OPMODE3**	**I/O Operation Mode** 00 = PF.7 is input only mode. 01 = PF.7 is output push pull mode. 10 = PF.7 is open drain mode. 11 = PF.7 is quasi-bidirectional mode.
[23:22]	**Reserved**	Reserved.
[21:20]	**PUSEL2**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.6 I/O Pull-up or Pull-down. 00 = PF.6 pull-up and pull-down Disabled. 01 = PF.6 pull-up Enabled. 10 = PF.6 pull-down Enabled.

		11 = PF.6 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE2 is set as input tri-state and open-drain mode.
[19]	**DINOFF2**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.6 digital input path Enabled. 1 = PF.6 digital input path Disabled (digital input tied to low).
[18]	**DOUT2**	**I/O Output Data** 0 = PF.6 output low. 1 = PF.6 output high.
[17:16]	**OPMODE2**	**I/O Operation Mode** 00 = PF.6 is input only mode. 01 = PF.6 is output push pull mode. 10 = PF.6 is open drain mode. 11 = PF.6 is quasi-bidirectional mode.
[15:14]	**Reserved**	Reserved.
[13:12]	**PUSEL1**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.5 I/O pull-up or pull-down. 00 = PF.5 pull-up and pull-down Disabled. 01 = PF.5 pull-up Enabled. 10 = PF.5 pull-down Enabled. 11 = PF.5 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE1 is set as input tri-state and open-drain mode.
[11]	**DINOFF1**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.5 digital input path Enabled. 1 = PF.5 digital input path Disabled (digital input tied to low).
[10]	**DOUT1**	**I/O Output Data** 0 = PF.5 output low. 1 = PF.5 output high.
[9:8]	**OPMODE1**	**I/O Operation Mode** 00 = PF.5 is input only mode. 01 = PF.5 is output push pull mode. 10 = PF.5 is open drain mode. 11 = PF.5 is quasi-bidirectional mode.
[7:6]	**Reserved**	Reserved.
[5:4]	**PUSEL0**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.4 I/O pull-up or pull-down. 00 = PF.4 pull-up and pull-down Disabled. 01 = PF.4 pull-up Enabled. 10 = PF.4 pull-down Enabled. 11 = PF.4 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE0 is set as input tri-state and open-drain mode.

[3]	**DINOFF0**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.4 digital input path Enabled. 1 = PF.4 digital input path Disabled (digital input tied to low).
[2]	**DOUT0**	**I/O Output Data** 0 = PF.4 output low. 1 = PF.4 output high.
[1:0]	**OPMODE0**	**I/O Operation Mode** 00 = PF.4 is input only mode. 01 = PF.4 is output push pull mode. 10 = PF.4 is open drain mode. 11 = PF.4 is quasi-bidirectional mode.

Note: RTC GPIO will be input mode when not programmed, and user needs to avoid the pin floating problem.

RTC GPIO Control Register1 (RTC_GPIOCTL1)

Register	Offset	R/W	Description	Reset Value
RTC_GPIOCTL1	RTC_BA+0x108	R/W	RTC GPIO Control 1 Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		PUSEL7		DINOFF7	DOUT7	OPMODE7	
23	22	21	20	19	18	17	16
Reserved		PUSEL6		DINOFF6	DOUT6	OPMODE6	
15	14	13	12	11	10	9	8
Reserved		PUSEL5		DINOFF5	DOUT5	OPMODE5	
7	6	5	4	3	2	1	0
Reserved		PUSEL4		DINOFF4	DOUT4	OPMODE4	

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**PUSEL7**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.11 I/O pull-up or pull-down. 00 = PF.11 pull-up and pull-down Disabled. 01 = PF.11 pull-up Enabled. 10 = PF.11 pull-down Enabled. 11 = PF.11 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE7 is set as input tri-state and open-drain mode.
[27]	**DINOFF7**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.11 digital input path Enabled. 1 = PF.11 digital input path Disabled (digital input tied to low).
[26]	**DOUT7**	**I/O Output Data** 0 = PF.11 output low. 1 = PF.11 output high.
[25:24]	**OPMODE7**	**I/O Operation Mode** 00 = PF.11 is input only mode. 01 = PF.11 is output push pull mode. 10 = PF.11 is open drain mode. 11 = PF.11 is quasi-bidirectional mode.
[23:22]	**Reserved**	Reserved.
[21:20]	**PUSEL6**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.10 I/O pull-up or pull-down. 00 = PF.10 pull-up and pull-down Disabled. 01 = PF.10 pull-up Enabled. 10 = PF.10 pull-down Enabled.

		11 = PF.10 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE6 is set as input tri-state and open-drain mode.
[19]	**DINOFF6**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.10 digital input path Enabled. 1 = PF.10 digital input path Disabled (digital input tied to low).
[18]	**DOUT6**	**I/O Output Data** 0 = PF.10 output low. 1 = PF.10 output high.
[17:16]	**OPMODE6**	**I/O Operation Mode** 00 = PF.10 is input only mode. 01 = PF.10 is output push pull mode. 10 = PF.10 is open drain mode. 11 = PF.10 is quasi-bidirectional mode .
[15:14]	**Reserved**	Reserved.
[13:12]	**PUSEL5**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.9 I/O pull-up or pull-down. 00 = PF.9 pull-up and pull-down Disabled. 01 = PF.9 pull-up Enabled. 10 = PF.9 pull-down Enabled. 11 = PF.9 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE5 is set as input tri-state and open-drain mode.
[11]	**DINOFF5**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.9 digital input path Enabled. 1 = PF.9 digital input path Disabled (digital input tied to low).
[10]	**DOUT5**	**I/O Output Data** 0 = PF.9 output low. 1 = PF.9 output high.
[9:8]	**OPMODE5**	**I/O Operation Mode** 00 = PF.9 is input only mode. 01 = PF.9 is output push pull mode. 10 = PF.9 is open drain mode. 11 = PF.9 is quasi-bidirectional mode .
[7:6]	**Reserved**	Reserved.
[5:4]	**PUSEL4**	**I/O Pull-up and Pull-down Enable Bits** Determine PF.8 I/O pull-up or pull-down. 00 = PF.8 pull-up and pull-down Disabled. 01 = PF.8 pull-up Enabled. 10 = PF.8 pull-down Enabled. 11 = PF.8 pull-up and pull-down Disabled. **Note:** Basically, the pull-up control and pull-down control has following behavior limitation. The independent pull-up / pull-down control register is only valid when OPMODE4 is set as input tri-state and open-drain mode.

[3]	**DINOFF4**	**I/O Pin Digital Input Path Disable Bit** 0 = PF.8 digital input path Enabled. 1 = PF.8 digital input path Disabled (digital input tied to low).
[2]	**DOUT4**	**I/O Output Data** 0 = PF.8 output low. 1 = PF.8 output high.
[1:0]	**OPMODE4**	**I/O Operation Mode** 00 = PF.8 is input only mode. 01 = PF.8 is output push pull mode. 10 = PF.8 is open drain mode. 11 = PF.8 is quasi-bidirectional mode.

Note: RTC GPIO will be input mode when not programmed, and user needs to avoid the pin floating problem.

RTC Daylight Saving Time Control Register (RTC_DSTCTL)

Register	Offset	R/W	Description	Reset Value
RTC_DSTCTL	RTC_BA+0x110	R/W	RTC Daylight Saving Time Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					DSBAK	SUBHR	ADDHR

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**DSBAK**	**Daylight Saving Back** 0 = Daylight Saving Change is not performed. 1 = Daylight Saving Change is performed.
[1]	**SUBHR**	**Subtract 1 Hour** 0 = No effect. 1 = Indicates RTC hour digit has been subtracted one hour for winter time change.
[0]	**ADDHR**	**Add 1 Hour** 0 = No effect. 1 = Indicates RTC hour digit has been added one hour for summer time change.

RTC Tamper Pin Control Register (RTC_TAMPCTL)

Register	Offset	R/W	Description	Reset Value
RTC_TAMPCTL	RTC_BA+0x120	R/W	RTC Tamper Pin Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DYNPR2EN	TAMP5DBEN	TAMP5LV	TAMP5EN	Reserved	TAMP4DBEN	TAMP4LV	TAMP4EN
23	22	21	20	19	18	17	16
DYNPR1EN	TAMP3DBEN	TAMP3LV	TAMP3EN	Reserved	TAMP2DBEN	TAMP2LV	TAMP2EN
15	14	13	12	11	10	9	8
DYNPR0EN	TAMP1DBEN	TAMP1LV	TAMP1EN	Reserved	TAMP0DBEN	TAMP0LV	TAMP0EN
7	6	5	4	3	2	1	0
DYNRATE			SEEDRLD	DYNSRC	Reserved	DYN2ISS	DYN1ISS

Bits	Description	
[31]	**DYNPR2EN**	**Dynamic Pair 2 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[30]	**TAMP5DEN**	**Tamper 5 De-bounce Enable Bit** 0 = Tamper 5 de-bounce Disabled. 1 = Tamper 5 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[29]	**TAMP5LV**	**Tamper 5 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[28]	**TAMP5EN**	**Tamper 5 Detect Enable Bit** 0 = Tamper 5 detect Disabled. 1 = Tamper 5 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector need to sync 2 ~ 3 RTC counter clock.
[27]	**Reserved**	Reserved.
[26]	**TAMP4DEN**	**Tamper 4 De-bounce Enable Bit** 0 = Tamper 4 de-bounce Disabled. 1 = Tamper 4 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[25]	**TAMP4LV**	**Tamper 4 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[24]	**TAMP4EN**	**Tamper4 Detect Enable Bit** 0 = Tamper 4 detect Disabled. 1 = Tamper 4 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector need to sync 2 ~ 3

		RTC counter clock.
[23]	**DYNPR1EN**	**Dynamic Pair 1 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[22]	**TAMP3DEN**	**Tamper 3 De-bounce Enable Bit** 0 = Tamper 3 de-bounce Disabled. 1 = Tamper 3 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[21]	**TAMP3LV**	**Tamper 3 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[20]	**TAMP3EN**	**Tamper 3 Detect Enable Bit** 0 = Tamper 3 detect Disabled. 1 = Tamper 3 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector need to sync 2 ~ 3 RTC counter clock.
[19]	**Reserved**	Reserved.
[18]	**TAMP2DEN**	**Tamper 2 De-bounce Enable Bit** 0 = Tamper 2 de-bounce Disabled. 1 = Tamper 2 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[17]	**TAMP2LV**	**Tamper 2 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[16]	**TAMP2EN**	**Tamper 2 Detect Enable Bit** 0 = Tamper 2 detect Disabled. 1 = Tamper 2 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector need to sync 2 ~ 3 RTC counter clock.
[15]	**DYNPR0EN**	**Dynamic Pair 0 Enable Bit** 0 = Static detect. 1 = Dynamic detect.
[14]	**TAMP1DEN**	**Tamper 1 De-bounce Enable Bit** 0 = Tamper 1 de-bounce Disabled. 1 = Tamper 1 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[13]	**TAMP1LV**	**Tamper 1 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[12]	**TAMP1EN**	**Tamper 1 Detect Enable Bit** 0 = Tamper 1 detect Disabled. 1 = Tamper 1 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector needs to sync 2 ~ 3 RTC counter clock.
[11]	**Reserved**	Reserved.

[10]	**TAMP0DEN**	**Tamper 0 De-bounce Enable Bit** 0 = Tamper 0 de-bounce Disabled. 1 = Tamper 0 de-bounce Enabled, tamper detection pin will sync 1 RTC counter clock.
[9]	**TAMP0LV**	**Tamper 0 Level** This bit depends on level attribute of tamper pin for static tamper detection. 0 = Detect voltage level is low. 1 = Detect voltage level is high.
[8]	**TAMP0EN**	**Tamper0 Detect Enable Bit** 0 = Tamper 0 detect Disabled. 1 = Tamper 0 detect Enabled. **Note:** The detection reference clock is RTC counter clock. Tamper detector needs to sync 2 ~ 3 RTC counter clock.
[7:5]	**DYNRATE**	**Dynamic Change Rate** This item is choice the dynamic tamper output change rate. 000 = 2^{10} * RTC_CLK. 001 = 2^{11} * RTC_CLK. 010 = 2^{12} * RTC_CLK. 011 = 2^{13} * RTC_CLK. 100 = 2^{14} * RTC_CLK. 101 = 2^{15} * RTC_CLK. 110 = 2^{16} * RTC_CLK. 111 = 2^{17} * RTC_CLK. **Note:** After revising this field, set SEEDRLD (RTC_TAMPCTL[4]) can reload change rate immediately.
[4]	**SEEDRLD**	**Reload New Seed for PRNG Engine** Setting this bit, the tamper configuration will be reloaded. 0 = Generating key based on the current seed. 1 = Reload new seed. **Note 1:** Before this bit is set, the tamper configuration should be set to complete and this bit will be auto clear to 0 after reload new seed completed. **Note 2:** The detection reference clock is RTC counter clock. Tamper detector needs to sync 2 ~ 3 RTC counter clock.
[3]	**DYNSRC**	**Dynamic Reference Pattern** This fields determine the new reference pattern when current pattern run out in dynamic pair mode. 0 = The new reference pattern is generated by random number generator when the reference pattern run out. 1 = The new reference pattern is repeated from SEED (RTC_TAMPSEED[31:0]) when the reference pattern run out. **Note:** After this bit is modified, the SEEDRLD (RTC_TAMPCTL[4]) should be set.
[2]	**Reserved**	Reserved.
[1]	**DYN2ISS**	**Dynamic Pair 2 Input Source Select** This bit determine Tamper 5 input is from Tamper 4 or Tamper 0 in dynamic mode. 0 = Tamper input is from Tamper 4. 1 = Tamper input is from Tamper 0. **Note:** This bit has effect only when DYNPR2EN (RTC_TAMPCTL[24]) and DYNPR0EN (RTC_TAMPCTL[15]) are set.
[0]	**DYN1ISS**	**Dynamic Pair 1 Input Source Select**

		This bit determine Tamper 3 input is from Tamper 2 or Tamper 0 in dynamic mode. 0 = Tamper input is from Tamper 2. 1 = Tamper input is from Tamper 0. **Note:** This bit is effective only when DYNPR1EN (RTC_TAMPCTL[16]) and DYNPR0EN (RTC_TAMPCTL[15]) are set.

RTC Tamper Dynamic Seed Register (RTC_TAMPSEED)

Register	6.16.7.1 Offset	R/W	Description	Reset Value
RTC_TAMPSEED	RTC_BA+0x128	R/W	RTC Tamper Dynamic Seed Register	0x0000_0000

31	30	29	28	27	26	25	24
SEED							
23	22	21	20	19	18	17	16
SEED							
15	14	13	12	11	10	9	8
SEED							
7	6	5	4	3	2	1	0
SEED							

Bits	6.16.7.2 Description	
[31:0]	SEED	Seed Value

RTC Tamper Time Register (RTC_TAMPTIME)

Register	Offset	R/W	Description	Reset Value
RTC_TAMPTIME	RTC_BA+0x130	R	RTC Tamper Time Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	**HZCNT**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**TENHR**		**HR**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**TENMIN**			**MIN**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**TENSEC**			**SEC**			

Bits	Description	
[31]	**Reserved**	Reserved.
[30:24]	**HZCNT**	Index of sub-second counter (0x00~0x7F)
[21:20]	**TENHR**	10-Hour Time Digit of TAMPER Time (0~2) **Note:** 24-hour time scale only .
[19:16]	**HR**	1-Hour Time Digit of TAMPER Time (0~9)
[15]	**Reserved**	Reserved.
[14:12]	**TENMIN**	10-Min Time Digit of TAMPER Time (0~5)
[11:8]	**MIN**	1-Min Time Digit of TAMPER Time (0~9)
[7]	**Reserved**	Reserved.
[6:4]	**TENSEC**	10-Sec Time Digit of TAMPER Time (0~5)
[3:0]	**SEC**	1-Sec Time Digit of TAMPER Time (0~9)

Note:

1. RTC_TAMPTIME is a BCD digit counter, except HZCNT.
2. The reasonable value range is listed in the parenthesis.
3. This fields cannot be updated until all TAMPxIF are cleared.

RTC Tamper Calendar Register (RTC_TAMPCAL)

Register	Offset	R/W	Description	Reset Value
RTC_TAMPCAL	RTC_BA+0x134	R	RTC Tamper Calendar Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
TENYEAR				YEAR			
15	14	13	12	11	10	9	8
Reserved			TENMON	MON			
7	6	5	4	3	2	1	0
Reserved		TENDAY		DAY			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**TENYEAR**	10-Year Calendar Digit of TAMPER Calendar (0~9)
[19:16]	**YEAR**	1-Year Calendar Digit of TAMPER Calendar (0~9)
[15:13]	**Reserved**	Reserved.
[12]	**TENMON**	10-Month Calendar Digit of TAMPER Calendar (0~1)
[11:8]	**MON**	1-Month Calendar Digit of TAMPER Calendar (0~9)
[7:6]	**Reserved**	Reserved.
[5:4]	**TENDAY**	10-Day Calendar Digit of TAMPER Calendar (0~3)
[3:0]	**DAY**	1-Day Calendar Digit of TAMPER Calendar (0~9)

Note:

1. RTC_TAMPCAL is a BCD digit counter.
2. The reasonable value range is listed in the parenthesis.
3. This fields cannot be updated until all TAMPxIF are cleared.

6.17 Camera Capture Interface Controller (CCAP)

6.17.1 Overview

The camera capture interface controller (CCAP) is designed to capture image data from a sensor. After capturing or fetching image data, it processes the image data. Then, the embedded DMA controller will move the data from the internal FIFO to system memory with AHB bus.

6.17.2 Features

- CCIR601 & CCIR656 interfaces supported for connection to CMOS image sensor
- 4-bit and 8-bit camera data width for CCIR656 interface
- YUV422 and RGB565 color format supported for data-in from CMOS sensor
- YUV422, RGB565, RGB555 and Y-only color supported for packet data output.
- Single interrupt source to interrupt controller from maskable interrupt source: Address Match, Bus Master Transfer Error, Video Frame End
- Embedded DMA controller supported to transfer data from internal FIFO to system memory through AHB bus
- CROP function supported to crop input image to the required size for digital application.
- Frame rate scaling-down supported
- Image scaling-down supported
- Bit luma output with 8-bit threshold setting supported.

6.17.3 Block Diagram

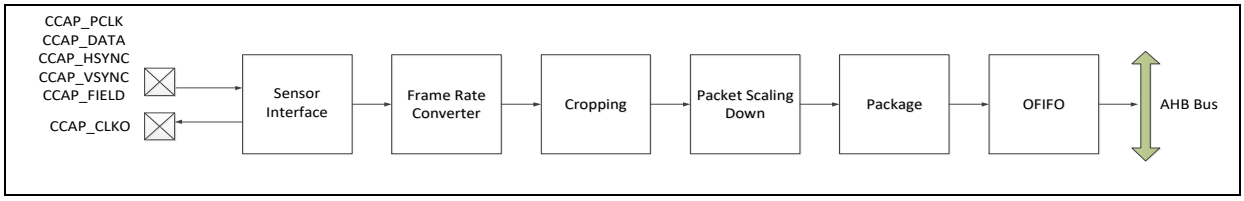


Figure 6.17-1 Camera Capture Interface Controller Functional Block Diagram

6.17.4 Basic Configuration

- Clock Source Configuration
  - Select the source of CCAP sensor pixel clock (to sensor) on CCAPSEL (CLK_CLKSEL0[17:16]).
  - Select the divider number from CCAP sensor pixel clock (to sensor) on VSENSEDIV (CLK_CLKDIV3[15:8]).
  - Enable CCAP controller clock in CCAPCKEN (CLK_AHBCLK0[8]).
  - Enable CCAP sensor pixel clock (to sensor) in SENCKEN (CLK_AHBCLK0[9]).
- Reset Configuration
  - Reset CCAP controller in CCAPRST (SYS_IPRST0[7]).

6.17.4.1 CCAP Pin Description

Name	BITs	I/O Type	Pin Description
CCAP_DATA	8	Input	Sensor pixel data

CCAP_HSYNC	1	Input	Sensor horizontal synchronization
CCAP_VSYNC	1	Input	Sensor vertical synchronization
CCAP_PCLK	1	Input	Sensor pixel clock(from sensor)
CCAP_FIELD	1	Input	Even/Odd Field Flag
CCAP_CLKO	1	Output	Sensor pixel clock(to sensor)

- The CCAP_DATA is synchronous with CCAP_PCLK. You can set the register PCLKP (CCAP_PAR[8]) to change the polarity of the rising/falling edge.
- The CCAP_HSYNC signal indicates the start/end of a line. You can set the HSP (CCAP_PAR[9]) to change its polarity.
- The CCAP_VSYNC signal indicates the start/end of a frame. You can set the VSP (CCAP_PAR[10]) to change its polarity.
- The CCAP_FIELD signal indicates the odd field or the even field.
- The CCAP_CLKO signal indicates the pixel clock to the sensor.

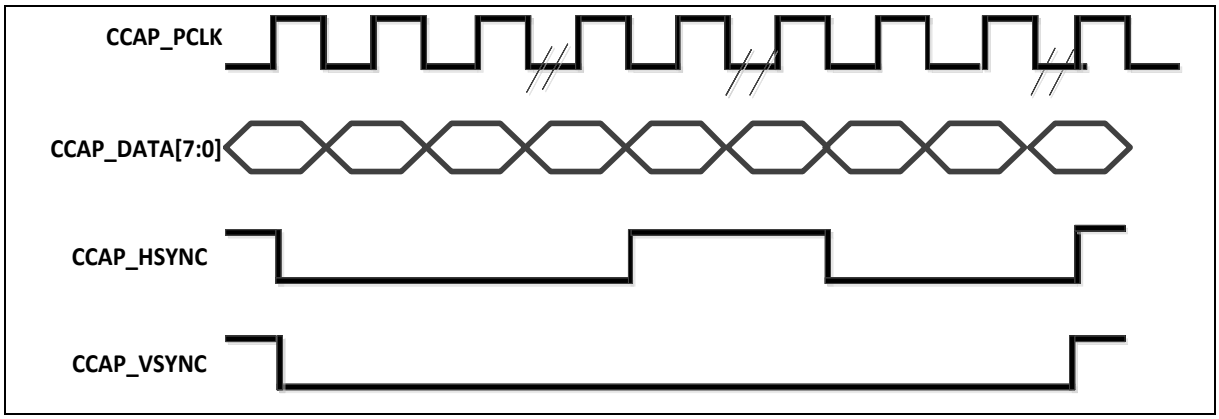


Figure 6.17-2 CCAP Basic Pin Diagram

6.17.5 Functional Description

6.17.5.1 Image Capture Flow

When user wants to use the CCAP function to capture image data from a sensor to system memory, it is necessary to do these settings.

- Configure the GPIO to set the MFP.
- Configure the clocks to enable the clock of the CCAP.
- Configure the related control register according to input CMOS sensor spec
- Configure the interrupt related control register to enable or mask the interrupt.
- Configure the embedded DMA related control register to let DMA move the data to the system memory
- Configure the control register about CCAP feature (Cropping, One shutter mode, Luminance 8-bit to 1-bit conversion, frame rate scaling down, packet vertical/horizontal scaling down)
- Configure the control register to enable the capture function and packet output

6.17.5.2 Sensor Type and Output Format

The CCAP supports two different sensors: Monochrome CMOS sensor and Color CMOS sensor. User

can set the MONO(CCAP_CTL[7]) according to the input sensor.

The CCAP supports two different sensor input format (YCbCr422,RGB565). User can set the INFMT(CCAP_PAR[0]) according to the input sensor format.

The CCAP supports two different sensor input types (CCIR601,CCIR656). User can set the SENTYPE(CCAP_PAR[1]) according to the input sensor type.

The CCAP supports different sensor input data order. User can set the INDATORD(CCAP_PAR[3:2]) according to the input sensor order.

INFMT(CCAP_PAR[0])	INDATORD(CCAP_PAR[3:2])	Sensor Input Data Order
0 (YCbCr)	0	Byte 0 1 2 3 : Y0 U0 Y1 V0
0 (YCbCr)	1	Byte 0 1 2 3 : Y0 V0 Y1 U0
0 (YCbCr)	2	Byte 0 1 2 3 : U0 Y0 V0 Y1
0 (YCbCr)	3	Byte 0 1 2 3 : V0 Y0 U0 Y1
1 (RGB565)	0	Byte 0 1 :{R[4:0],G[5:3]} {G[2:0],B[4:0]}
1 (RGB565)	1	Byte 0 1 :{B[4:0],G[5:3]} {G[2:0],R[4:0]}
1 (RGB565)	2	Byte 0 1 :{G[2:0],B[4:0]} {R[4:0],G[5:3]}
1 (RGB565)	3	Byte 0 1 :{G[2:0],R[4:0]} {B[4:0],G[5:3]}

The CCAP supports different sensor signal polarity (VSYNC,HSYNC,PCLKP). User can set the VSP(CCAP_PAR[10]), HSP(CCAP_PAR[9]), PLKP(CCAP_PAR[8]) according to the input sensor polarity.

For Monochrome CMOS sensor, the CCAP supports two different sensor data I/O interface and data sequence. User can set the MY8_MY4(CCAP_CTL[18]) and MY4_SWAP(CCAP_CTL[17]) according to the input sensor.

The CCAP supports four different output format to system memory (YCbCr422, only output Y,RGB555,RGB565). User can set the OUTFMT(CCAP_PAR[5:4]) according to the user need.

User can set the UPDATE(CCAP_CTL[24]) register. If this value is set to 1, it will update these register when new frame starts.

6.17.5.3 CCAP Interrupt Source Enable and Mask

The CCAP supports three interrupt sources:

- Address Match Interrupt: When Current Packet System Memory Address Register (CCAP_CURADDRP) is equal to the Compare Memory Address Register (CCAP_CMPADDR), it will trigger this interrupt.
- Bus Master Transfer Error Interrupt: When CCAP issues a request to the AHB bus, and get an error response from bus. It will trigger this interrupt.
- Video Frame End Interrupt: When the end of frame capture, it will trigger this interrupt.

User can set the Camera Capture Interface Interrupt Register (CCAP_INT) to enable or mask these interrupts.

6.17.5.4 Data Transfer with the Embedded DMA Controller

The CCAP has the embedded DMA controller to transfer the data from CCAP internal FIFO to outside system memory. User can set the CCAP_PKTBA0 control register to decide the system memory base address, and set the PKTSTRIDE to decide the pixel stride width between two lines. User also can read the CCAP_CURADDRP to know the status about which address will be written by CCAP.

User can set the threshold value by PKTFTH (CCAP_FIFOTH[28:24]).When the number of the words internal FIFO received are bigger than PKTFTH value, it will trigger the embedded DMA to move data

to system memory. And user can read OVF(CCAP_FIFOTH[31]) to know whether the internal FIFO is overflow or not.

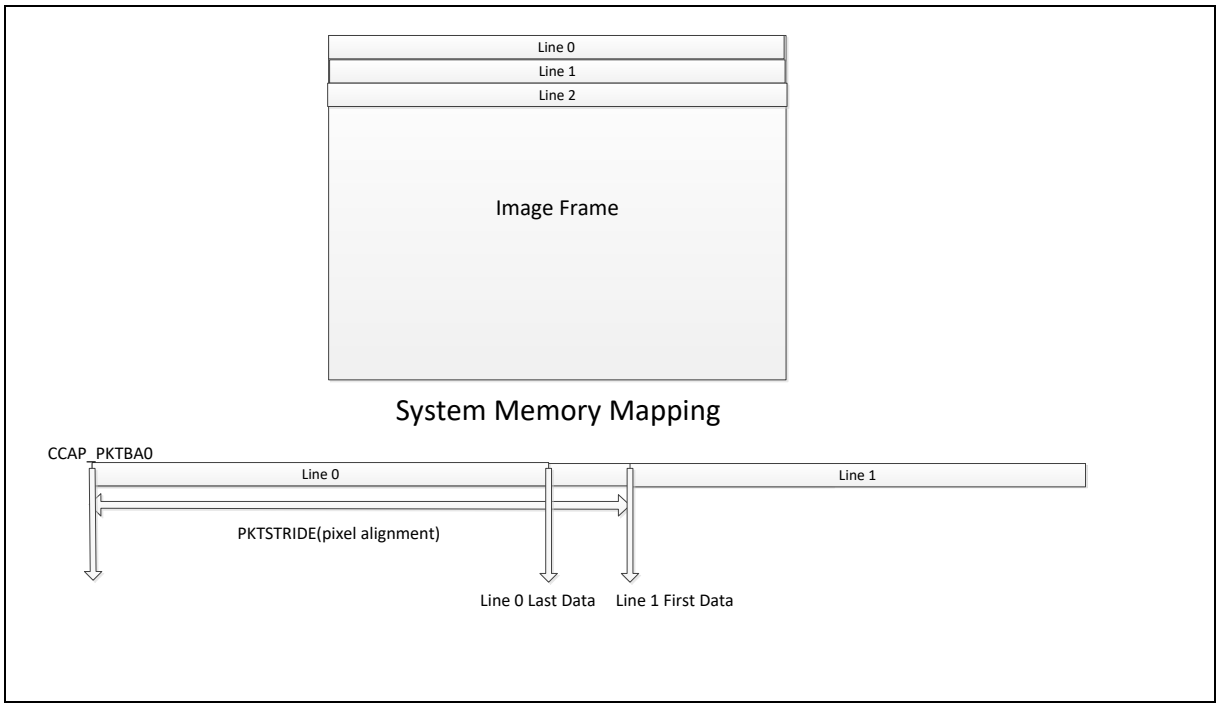


Figure 6.17-3 System Memory Mapping Block Cropping Feature

The capture interface can select a window from the received image. The size of the window is specified by the number of pixel clocks (horizontal dimension) and the number of lines (vertical dimension). The start (left upper corner) coordinates can be specified by the CCAP_CWSP register. The size (vertical dimension in number of lines and horizontal dimension in number of pixel clocks) can be specified by the CCAP_CWS register.

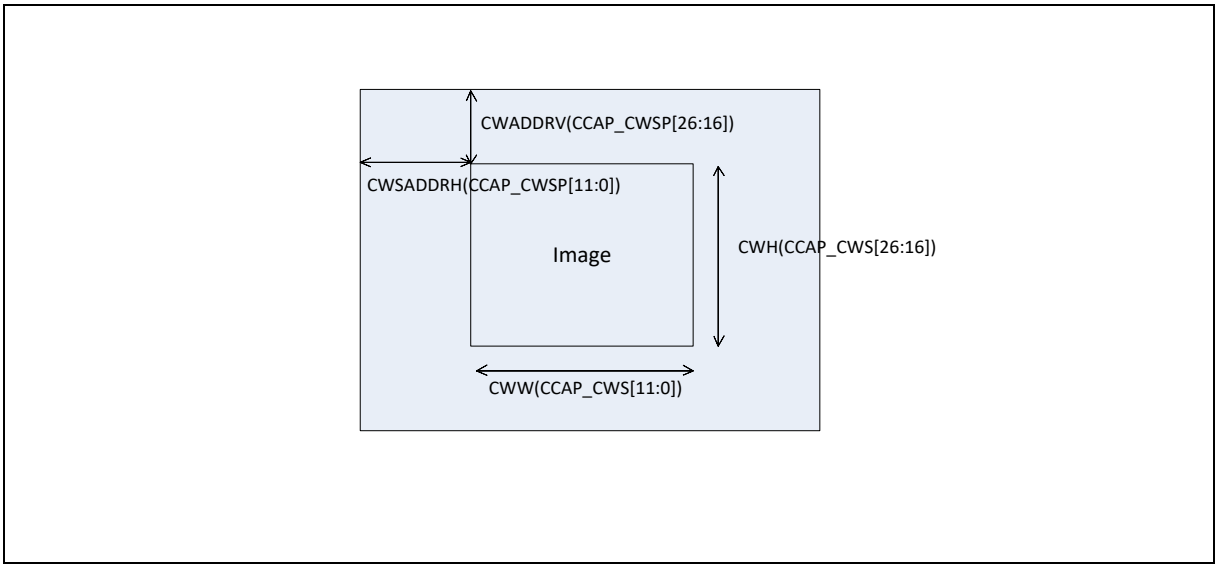


Figure 6.17-4 Image Start and Size of the Window after Cropping Block

6.17.5.5 One Shutter Mode (Single Frame)

In this mode, a single frame is captured. After the SHUTTER (CCAP_CTL[16]) bit is set, the Image

Capture interface automatically disables the capture interface after a frame is captured.

6.17.5.6 Luminance 8-bit to 1-bit Conversion Feature

When Luma_Y_one (CCAP_CTL[19]) bit is set, it will convert Y-8-bit to Y-1-bit according to the LUMA_Y1_THRESH (CCAP_LUMA_Y1_THD[7:0]).When the Y-8-bit value is bigger than LUMA_Y1_THRESH,it will output 1.Otherwise,it will output 0.

6.17.5.7 Frame Rate Scaling Down Feature

The CCAP can set the FRM(CCAP_FRCTL[5:0]) and FRN(CCAP_FRCTL[13:8]) to support frame rate scaling down. The output image frame rate will be equal to input image format *(FRN/FRM)

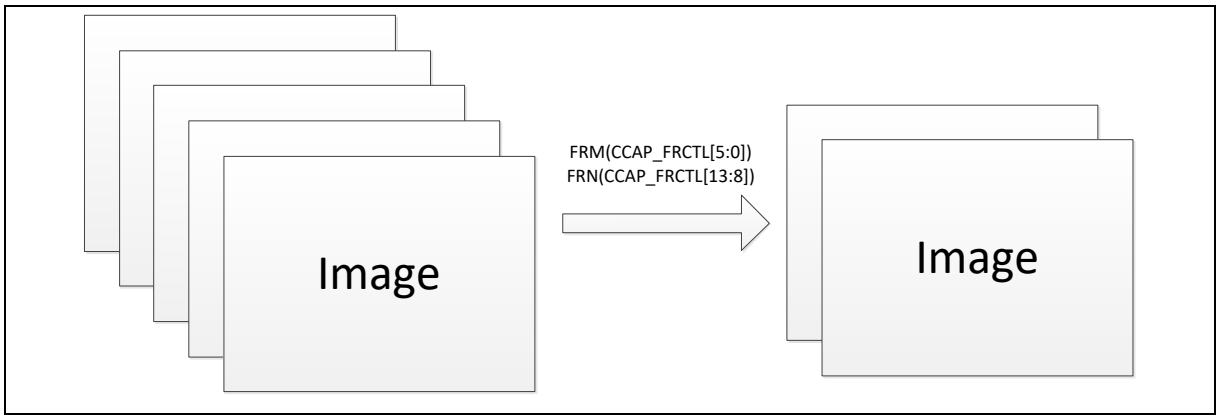


Figure 6.17-5 Frame Rate Scaling Down Block

6.17.5.8 Packet Vertical/Horizontal Scaling Down Feature

The CCAP can set the CCAP_PKTSM and CCAP_PKTSL control register to scale down the vertical/horizontal of input image frame.

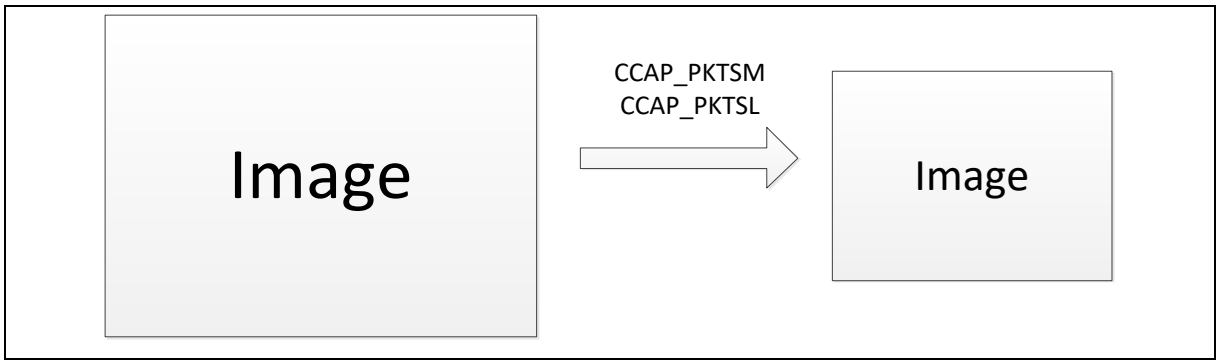


Figure 6.17-6 Frame Packet Scaling Down Block

6.17.5.9 Enable the Capture Interface

User can set the CCAPEN(CCAP_CTL[0]) to enable the capture interface, and set the PKTEN (CCAP_CTL[6]) to enable the packet output.

6.17.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write.

Register	Offset	R/W	Description	Reset Value
CCAP Base Address: **CCAP_BA = 0x4003_0000**				
CCAP_CTL	CCAP_BA+0x00	R/W	Camera Capture Interface Control Register	0x0000_0050
CCAP_PAR	CCAP_BA+0x04	R/W	Camera Capture Interface Parameter Register	0x0000_0000
CCAP_INT	CCAP_BA+0x08	R/W	Camera Capture Interface Interrupt Register	0x0000_0000
CCAP_CWSP	CCAP_BA+0x20	R/W	Cropping Window Starting Address Register	0x0000_0000
CCAP_CWS	CCAP_BA+0x24	R/W	Cropping Window Size Register	0x0000_0000
CCAP_PKTSL	CCAP_BA+0x28	R/W	Packet Scaling Vertical/Horizontal Factor Register (LSB)	0x0000_0000
CCAP_FRCTL	CCAP_BA+0x30	R/W	Scaling Frame Rate Factor Register	0x0000_0000
CCAP_STRIDE	CCAP_BA+0x34	R/W	Frame Output Pixel Stride Width Register	0x0000_0000
CCAP_FIFOTH	CCAP_BA+0x3C	R/W	FIFO Threshold Register	0x070D_0507
CCAP_CMPADDR	CCAP_BA+0x40	R/W	Compare Memory Base Address Register	0xFFFF_FFFC
CCAP_LUMA_Y1_THD	CCAP_BA+0x44	R/W	Luminance Y8 to Y1 Threshold Value Register	0x0000_0080
CCAP_PKTSM	CCAP_BA+0x48	R/W	Packet Scaling Vertical/Horizontal Factor Register (MSB)	0x0000_0000
CCAP_CURADDRP	CCAP_BA+0x50	R	Current Packet System Memory Address Register	0x0000_0000
CCAP_PKTBA0	CCAP_BA+0x60	R/W	System Memory Packet Base Address 0 Register	0x0000_0000

6.17.7 Register Description

Camera Capture Interface Control Register (CCAP_CTL)

Register	Offset	R/W	Description	Reset Value
CCAP_CTL	CCAP_BA+0x00	R/W	Camera Capture Interface Control Register	0x0000_0050

31	30	29	28	27	26	25	24
Reserved							**VPRST**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved			**UPDATE**	**Luma_Y_One**	**MY8_MY4**	**MY4_SWAP**	**SHUTTER**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MONO	**PKTEN**	**Reserved**					**CCAPEN**

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**VPRST**	**Capture Interface Reset** 0 = Capture interface reset Disabled. 1 = Capture interface reset Enabled.
[23:21]	**Reserved**	Reserved.
[20]	**UPDATE**	**Update Register at New Frame** 0 = Update register at new frame Disabled. 1 = Update register at new frame Enabled (auto cleared to 0 when register updated).
[19]	**Luma_Y_One**	**Color/Monochrome CMOS Sensor Luminance 8-bit Y to 1-bit Y Conversion** 0 = Color/Monochrome CMOS sensor Luma-Y-One bit Disabled. 1 = Color/Monochrome CMOS sensor Luma-Y-One bit Enabled. **Note:** Color CMOS sensor U/V components are ignored when the Luma_Y_One is enabled.
[18]	**MY8_MY4**	**Monochrome CMOS Sensor Data I/O Interface** 0 = Monochrome CMOS sensor is by the 4-bit data I/O interface. 1 = Monochrome CMOS sensor is by the 8-bit data I/O interface.
[17]	**MY4_SWAP**	**Monochrome CMOS Sensor 4-bit Data Nibble Swap** 0 = The 4-bit data input sequence: 1st Pixel is for 1st Nibble (1st pixel @MSB). 1 = The 4-bit data input sequence: 1st Pixel is for 2nd Nibble (1st pixel @LSB).
[16]	**SHUTTER**	**Camera Capture Interface Automatically Disable the Capture Interface After a Frame Had Been Captured** 0 = Shutter Disabled. 1 = Shutter Enabled.
[15:8]	**Reserved**	Reserved.

[7]	**MONO**	**Monochrome CMOS Sensor Select** 0 = Color CMOS Sensor. 1 = Monochrome CMOS Sensor. The U/V components are ignored when the MONO is enabled.
[6]	**PKTEN**	**Packet Output Enable Bit** 0 = Packet output Disabled. 1 = Packet output Enabled.
[5:1]	**Reserved**	Reserved.
[0]	**CCAPEN**	**Camera Capture Interface Enable Bit** 0 = Camera Capture Interface Disabled. 1 = Camera Capture Interface Enabled.

Camera Capture Interface Parameter Register (CCAP_PAR)

Register	Offset	R/W	Description	Reset Value
CCAP_PAR	CCAP_BA+0x04	R/W	Camera Capture Interface Parameter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					FBB	Reserved	
15	14	13	12	11	10	9	8
Reserved					VSP	HSP	PCLKP
7	6	5	4	3	2	1	0
Reserved	RANGE	OUTFMT		INDATORD		SENTYPE	INFMT

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**FBB**	**Field by Blank** Field by Blank (only in ccir-656 mode) means blanking pixel data(0x80108010) have to transfer to system memory or not. 0 = Field by blank Disabled. (blank pixel data will transfer to system memory). 1 = Field by blank Enabled. (only active data will transfer to system memory).
[10]	**VSP**	**Sensor Vsync Polarity** 0 = Sync Low. 1 = Sync High.
[9]	**HSP**	**Sensor Hsync Polarity** 0 = Sync Low. 1 = Sync High.
[8]	**PCLKP**	**Sensor Pixel Clock Polarity** 0 = Input video data and signals are latched by falling edge of Pixel Clock. 1 = Input video data and signals are latched by rising edge of Pixel Clock.
[7]	**Reserved**	Reserved.
[6]	**RANGE**	**Scale Input YUV CCIR601 Color Range to Full Range** 0 = Default. 1 = Scale to full range.
[5:4]	**OUTFMT**	**Image Data Format Output to System Memory** 00 = YCbCr422. 01 = Only output Y. (Select this format when CCAP_CTL “Luma_Y_One” or “MONO” enabled). 10 = RGB555. 11 = RGB565.
[3:2]	**INDATORD**	**Sensor Input Data Order**

		If INFMT (CCAP_PAR[0]) = 0 (YCbCr). 00 = Sensor input data (Byte 0 1 2 3) is Y0 U0 Y1 V0. 01 = Sensor input data (Byte 0 1 2 3) is Y0 V0 Y1 U0. 10 = Sensor input data (Byte 0 1 2 3) is U0 Y0 V0 Y1. 11 = Sensor input data (Byte 0 1 2 3) is V0 Y0 U0 Y1. If INFMT (CCAP_PAR[0]) = 1 (RGB565). 00 = Sensor input data (Byte 0) is {R[4:0],G[5:3]}. Sensor input data (Byte 1) is {G[2:0],B[4:0]}. 01 = Sensor input data (Byte 0) is {B[4:0],G[5:3]}. Sensor input data (Byte 1) is {G[2:0], R[4:0]}. 10 = Sensor input data (Byte 0) is {G[2:0],B[4:0]}. Sensor input data (Byte 1) is {R[4:0], G[5:3]}. 11 = Sensor input data (Byte 0) is {G[2:0],R[4:0]}. Sensor input data (Byte 1) is {B[4:0], G[5:3]}.
[1]	**SENTYPE**	**Sensor Input Type** 0 = CCIR601. 1 = CCIR656, Vsync & Hsync embedded in the data signal.
[0]	**INFMT**	**Sensor Input Data Format** 0 = YCbCr422. 1 = RGB565.

Camera Capture Interface Interrupt Register (CCAP_INT)

Register	Offset	R/W	Description	Reset Value
CCAP_INT	CCAP_BA+0x08	R/W	Camera Capture Interface Interrupt Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**ADDRMIEN**	**Reserved**	**MEIEN**	**VIEN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**ADDRMINTF**	**Reserved**	**MEINTF**	**VINTF**

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19]	**ADDRMIEN**	**Address Match Interrupt Enable Bit** 0 = Address match interrupt Disabled. 1 = Address match interrupt Enabled.
[18]	**Reserved**	Reserved.
[17]	**MEIEN**	**Bus Master Transfer Error Interrupt Enable Bit** 0 = Bus Master Transfer error interrupt Disabled. 1 = Bus Master Transfer error interrupt Enabled.
[16]	**VIEN**	**Video Frame End Interrupt Enable Bit** 0 = Video frame end interrupt Disabled. 1 = Video frame end interrupt Enabled.
[15:4]	**Reserved**	Reserved.
[3]	**ADDRMINTF**	**Memory Address Match Interrupt** 0 = Memory Address Match Interrupt did not occur. 1 = Memory Address Match Interrupt occurred. **Note:** This bit is cleared by writing 1 to it.
[2]	**Reserved**	Reserved.
[1]	**MEINTF**	**Bus Master Transfer Error Interrupt** 0 = Transfer Error did not occur. 1 = Transfer Error occurred. **Note:** This bit is cleared by writing 1 to it.
[0]	**VINTF**	**Video Frame End Interrupt** 0 = Did not receive a frame completely. 1 = Received a frame completely. **Note:** This bit is cleared by writing 1 to it.

Cropping Window Starting Address Register (CCAP_CWSP)

Register	Offset	R/W	Description	Reset Value
CCAP_CWSP	CCAP_BA+0x20	R/W	Cropping Window Starting Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					**CWSADDRV**		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CWSADDRV							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CWSADDRH**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CWSADDRH							

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:16]	**CWSADDRV**	**Cropping Window Vertical Starting Address** Specify the value of the cropping window vertical start address**.**
[15:12]	**Reserved**	Reserved.
[11:0]	**CWSADDRH**	**Cropping Window Horizontal Starting Address** Specify the value of the cropping window horizontal start address**.**

Cropping Window Size Register (CCAP_CWS)

Register	Offset	R/W	Description	Reset Value
CCAP_CWS	CCAP_BA+0x24	R/W	Cropping Window Size Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					**CWH**		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CWH							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CWW**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CWW							

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:16]	**CWH**	**Cropping Window Height** Specify the size of the cropping window height.
[15:12]	**Reserved**	Reserved.
[11:0]	**CWW**	**Cropping Window Width** Specify the size of the cropping window width.

Packet Scaling Vertical/Horizontal Factor Register (LSB) (CCAP_PKTSL)

Register	Offset	R/W	Description	Reset Value
CCAP_PKTSL	CCAP_BA+0x28	R/W	Packet Scaling Vertical/Horizontal Factor Register (LSB)	0x0000_0000

31	30	29	28	27	26	25	24
PKTSVNL							
23	22	21	20	19	18	17	16
PKTSVML							
15	14	13	12	11	10	9	8
PKTSHNL							
7	6	5	4	3	2	1	0
PKTSHML							

Bits	Description	
[31:24]	**PKTSVNL**	**Packet Scaling Vertical Factor N** Specify the lower 8-bit of numerator part (N) of the vertical scaling factor. The lower 8-bit will be cascaded with higher 8-bit (PKDSVNH) to form a 16-bit numerator of vertical factor.
[23:16]	**PKTSVML**	**Packet Scaling Vertical Factor M** Specify the lower 8-bit of denominator part (M) of the vertical scaling factor. The lower 8-bit will be cascaded with higher 8-bit (PKDSVMH) to form a 16-bit denominator (M) of vertical factor. The output image width will be equal to the image height * N/M. **Note:** The value of N must be equal to or less than M.
[15:8]	**PKTSHNL**	**Packet Scaling Horizontal Factor N** Specify the lower 8-bit of numerator part (N) of the horizontal scaling factor. The lower 8-bit will be cascaded with higher 8-bit (PKDSHNH) to form a 16-bit numerator of horizontal factor.
[7:0]	**PKTSHML**	**Packet Scaling Horizontal Factor M** Specify the lower 8-bit of denominator part (M) of the horizontal scaling factor. The lower 8-bit will be cascaded with higher 8-bit (PKDSHMH) to form a 16-bit denominator (M) of vertical factor. The output image width will be equal to the image width * N/M. **Note:** The value of N must be equal to or less than M.

Scaling Frame Rate Factor Register (CCAP_FRCTL)

Register	Offset	R/W	Description	Reset Value
CCAP_FRCTL	CCAP_BA+0x30	R/W	Scaling Frame Rate Factor Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**28**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**FRN**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**FRM**					

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:8]	**FRN**	**Scaling Frame Rate Factor N** Specify the numerator part (N) of the frame rate scaling factor.
[7:6]	**Reserved**	Reserved.
[5:0]	**FRM**	**Scaling Frame Rate Factor M** Specify the denominator part (M) of the frame rate scaling factor. The output image frame rate will be equal to input image frame rate * (N/M). **Note:** The value of N must be equal to or less than M.

Output Frame Pixel Stride Width (CCAP_STRIDE)

Register	Offset	R/W	Description	Reset Value
CCAP_STRIDE	CCAP_BA+0x34	R/W	Frame Output Pixel Stride Width Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**28**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**PKTSTRIDE**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PKTSTRIDE							

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:0]	**PKTSTRIDE**	**Packet Frame Output Pixel Stride Width** The output pixel stride size of the packet pipe. It is a 32-pixel aligned stride width for the Luma-Y-One bit format or a 4-pixel aligned stride with for the Luma-Y-Eight bit format when color or monochrome CMOS sensors used. This means that every new captured line is by word alignment address when color or monochrome CMOS sensors used.

FIFO Threshold Register (CCAP_FIFOTH)

Register	Offset	R/W	Description	Reset Value
CCAP_FIFOTH	CCAP_BA+0x3C	R/W	FIFO Threshold Register	0x070D_0507

31	30	29	28	27	26	25	24
OVF	**Reserved**		**PKTFTH**				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31]	**OVF**	**FIFO Overflow Flag** Indicate the FIFO overflow flag.
[30:29]	**Reserved**	Reserved.
[28:24]	**PKTFTH**	**Packet FIFO Threshold** Specify the 5-bit value of the packet FIFO threshold.
[23:0]	**Reserved**	Reserved.

Compare Memory Address Register (CCAP_CMPADDR)

Register	Offset	R/W	Description	Reset Value
CCAP_CMPADDR	CCAP_BA+0x40	R/W	Compare Memory Base Address Register	0xFFFF_FFFC

31	30	29	28	27	26	25	24
CMPADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPADDR							

Bits	Description	
[31:0]	**CMPADDR**	**Compare Memory Base Address** It is a word alignment address, that is, the address is aligned by ignoring the 2 LSB bits [1:0].

Luminance Y8 to Y1 Threshold Value Register (CCAP_LUMA_Y1_THD)

Register	Offset	R/W	Description	Reset Value
CCAP_LUMA_Y1_THD	CCAP_BA+0x44	R/W	Luminance Y8 to Y1 Threshold Value Register	0x0000_0080

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
LUMA_Y1_THRESH							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**LUMA_Y1_THRESH**	**Luminance Y8 to Y1 Threshold Value** Specify the 8-bit threshold value for the luminance Y bit-8 to the luminance Y 1-bit conversion.

Packet Scaling Vertical/Horizontal Factor Register (MSB) (CCAP_PKTSM)

Register	Offset	R/W	Description	Reset Value
CCAP_PKTSM	CCAP_BA+0x48	R/W	Packet Scaling Vertical/Horizontal Factor Register (MSB)	0x0000_0000

31	30	29	28	27	26	25	24
PKTSVNH							
23	22	21	20	19	18	17	16
PKTSVMH							
15	14	13	12	11	10	9	8
PKTSHNH							
7	6	5	4	3	2	1	0
PKTSHMH							

Bits	Description	
[31:24]	PKTSVNH	**Packet Scaling Vertical Factor N** Specify the higher 8-bit of numerator part (N) of the vertical scaling factor. Please refer to the register "CCAP_PKTSL" to check the cooperation between these two registers.
[23:16]	PKTSVMH	**Packet Scaling Vertical Factor M** Specify the higher 8-bit of denominator part (M) of the vertical scaling factor. Please refer to the register "CCAP_PKTSL" to check the cooperation between these two registers.
[15:8]	PKTSHNH	**Packet Scaling Horizontal Factor N** Specify the higher 8-bit of numerator part (N) of the horizontal scaling factor. Please refer to the register "CCAP_PKTSL" for the detailed operation.
[7:0]	PKTSHMH	**Packet Scaling Horizontal Factor M** Specify the higher 8-bit of denominator part (M) of the horizontal scaling factor. Please refer to the register "CCAP_PKTSL" for the detailed operation.

Current Packet System Memory Address Register (CCAP_CURADDRP)

Register	Offset	R/W	Description	Reset Value
CCAP_CURADDRP	CCAP_BA+0x50	R	Current Packet System Memory Address Register	0x0000_0000

31	30	29	28	27	26	25	24
CURADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CURADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CURADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURADDR							

Bits	Description	
[31:0]	**CURADDR**	**Current Packet Output Memory Address** Specify the 32-bit value of the current packet output memory address.

System Memory Packet Base Address 0 Register (CCAP_PKTBA0)

Register	Offset	R/W	Description	Reset Value
CCAP_PKTBA0	CCAP_BA+0x60	R/W	System Memory Packet Base Address 0 Register	0x0000_0000

31	30	29	28	27	26	25	24
BASEADDR							
23	22	21	20	19	18	17	16
BASEADDR							
15	14	13	12	11	10	9	8
BASEADDR							
7	6	5	4	3	2	1	0
BASEADDR							

Bits	Description	
[31:0]	BASEADDR	**System Memory Packet Base Address 0** It is a word alignment address, that is, the address is aligned by ignoring the 2 LSB bits [1:0].

6.18 Enhanced Input Capture Timer (ECAP)

6.18.1 Overview

The chip provides up to four units of Input Capture Timer/Counter whose capture function can detect the digital edge-changed signal at channel inputs. Each unit has three input capture channels. The timer/counter is equipped with up counting, reload and compare-match capabilities.

6.18.2 Features

- Up to four Input Capture Timer/Counter units, CAP0, CAP1, CAP2 and CAP3.
- Each unit has 3 input channels.
- Each unit has its own interrupt vector.
- Each input channel has its own capture counter hold register.
- 24-bit Input Capture up-counting timer/counter.
- With noise filter in front end of input ports.
- Edge detector with three options:
  - Rising edge detection
  - Falling edge detection
  - Both edge detection
- Captured events reset and/or reload capture counter.
- Supports compare-match function.

6.18.3 Block Diagram

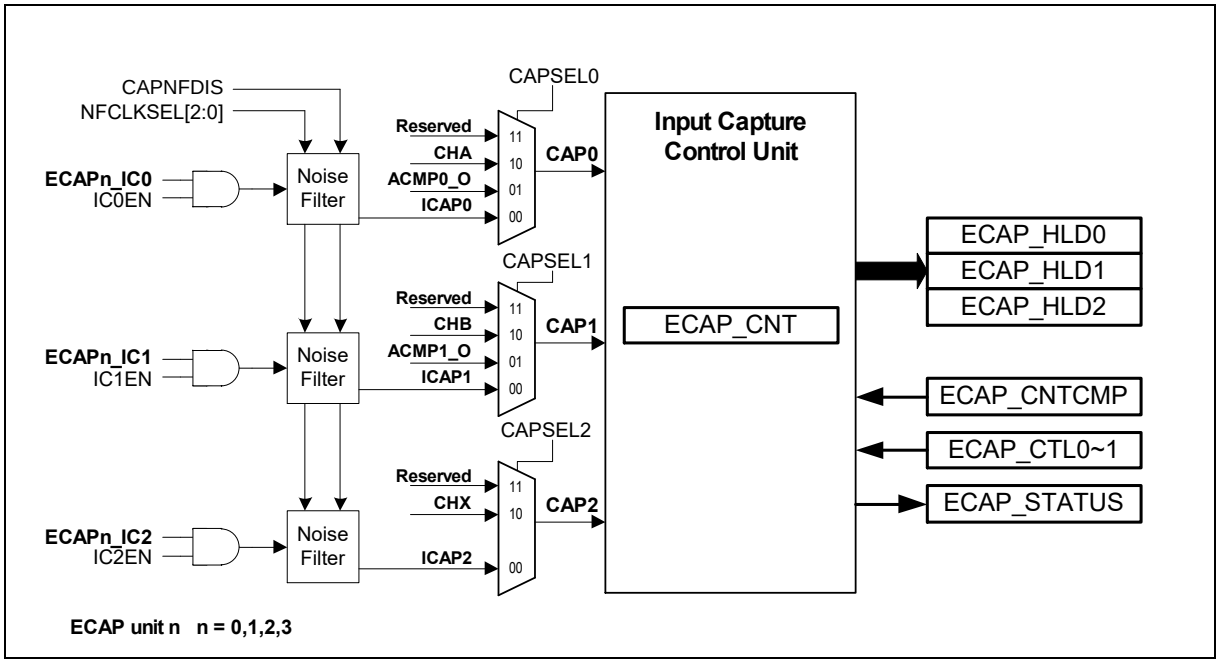


Figure 6.18-1 Input Capture Timer/Counter Architecture

6.18.4 Basic Configuration

6.18.4.1 ECAP0 Basic Configuration

- Clock Source Configuration
  - Enable ECAP0 peripheral clock in ECAP0CKEN(CLK_APBCLK1[26]).
- Reset Configuration
  - Reset ECAP0 peripheral in ECAP0RST (SYS_IPRST2[26]).

6.18.4.2 ECAP1 Basic Configuration

- Clock Source Configuration
  - Enable ECAP1 peripheral clock in ECAP1CKEN (CLK_APBCLK1[27]).
- Reset Configuration
  - Reset ECAP1 peripheral in ECAP1RST (SYS_IPRST2[27]).

6.18.4.3 ECAP2 Basic Configuration

- Clock Source Configuration
  - Enable ECAP2 peripheral clock in ECAP2CKEN (CLK_APBCLK1[13]).
- Reset Configuration
  - Reset ECAP2 peripheral in ECAP2RST (SYS_IPRST2[13]).

6.18.4.4 ECAP3 Basic Configuration

- Clock Source Configuration
  - Enable ECAP3 peripheral clock in ECAP3CKEN (CLK_APBCLK1[14]).
- Reset Configuration
  - Reset ECAP3 peripheral in ECAP3RST (SYS_IPRST2[14]).

6.18.5 Functional Description

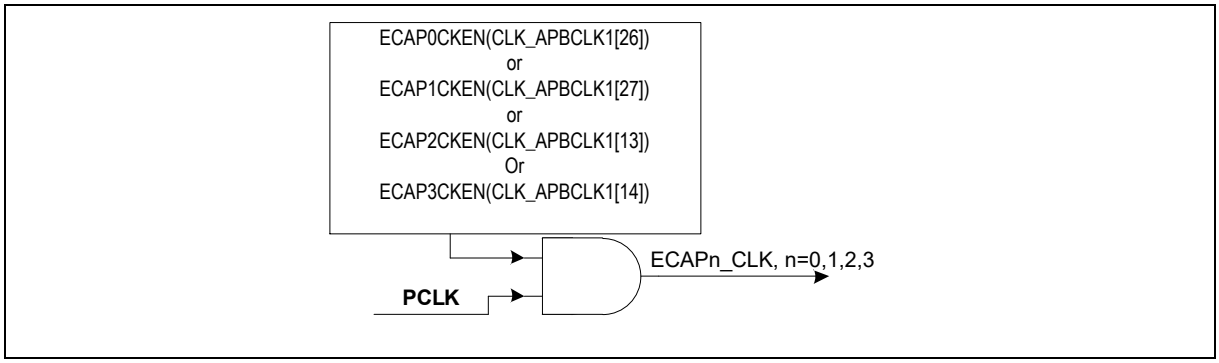


Figure 6.18-2 Input Capture Timer/Counter Clock Source Control

Figure 6.18-1 illustrates the architecture of the Input Capture. Each input capture timer/counter unit supports 3 input channels with programmable input signal sources. The port pins ECAP_IC0 to ECAP_IC2 can be fed to the inputs of capture unit through noise filter or bypass it (CAPNFDIS = 1), and the QEI controller input signals (CHA, CHB and CHX) also can be internally routed to the capture inputs by setting the register ECAP_CTL0(CAPSEL0~ CAPSEL 2).

6.18.5.1 Input Noise Filter

The architecture of input noise filter is similar to that one used for QEI (refer to Figure 6.13-4). With 6 sampling-rate options, it supports a wide range of filtering noise whose duration is from 33 ns (NFCLKSEL = 000) to 2.1 us (NFCLKSEL = 101) while PCLK is running at the frequency of 90 MHz. Table 6.18-1 lists the relation between the setting of NFCLKSEL, the duration of filtered noise and the duration of the signal that is guaranteed to be sampled.

The maximum duration of the noise that is filtered out should not be more than 3 clock cycle, while the minimum duration of the signal that is guaranteed to be sampled should be more than 4 clock cycle.

NFCLKSEL	Maximum Duration Of Noise	Minimum Duration Of Signal
000	33 ns	44 ns
001	67 ns	89 ns
010	133 ns	178 ns
011	533 ns	711 ns
100	1067 ns	1422 ns

101	2133 ns	2844 ns
		PCLK= 90 MHz

Table 6.18-1 Typical Case of Noise Filter Settings

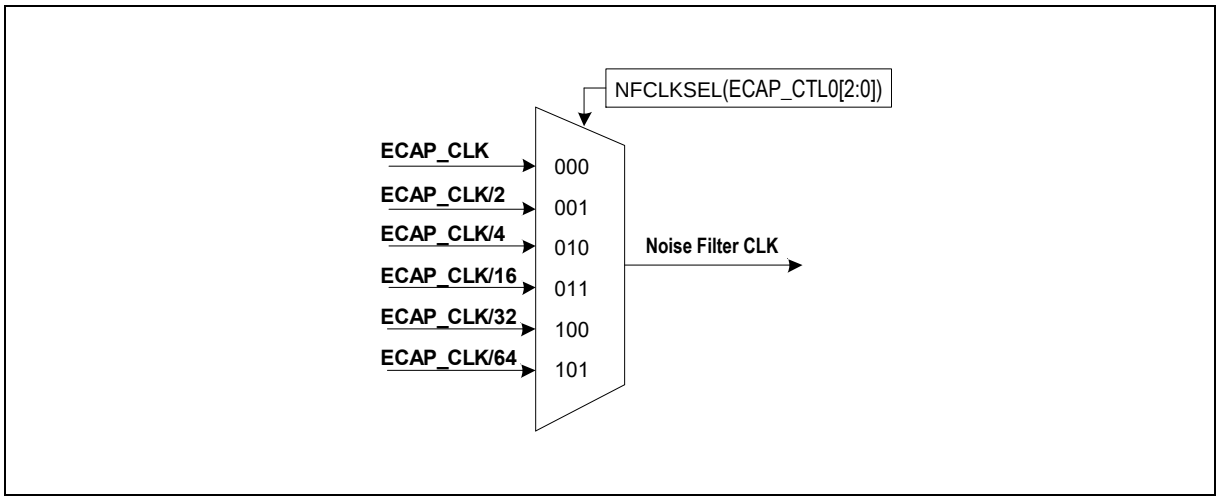


Figure 6.18-3 Noise Filter Sampling Clock Selection

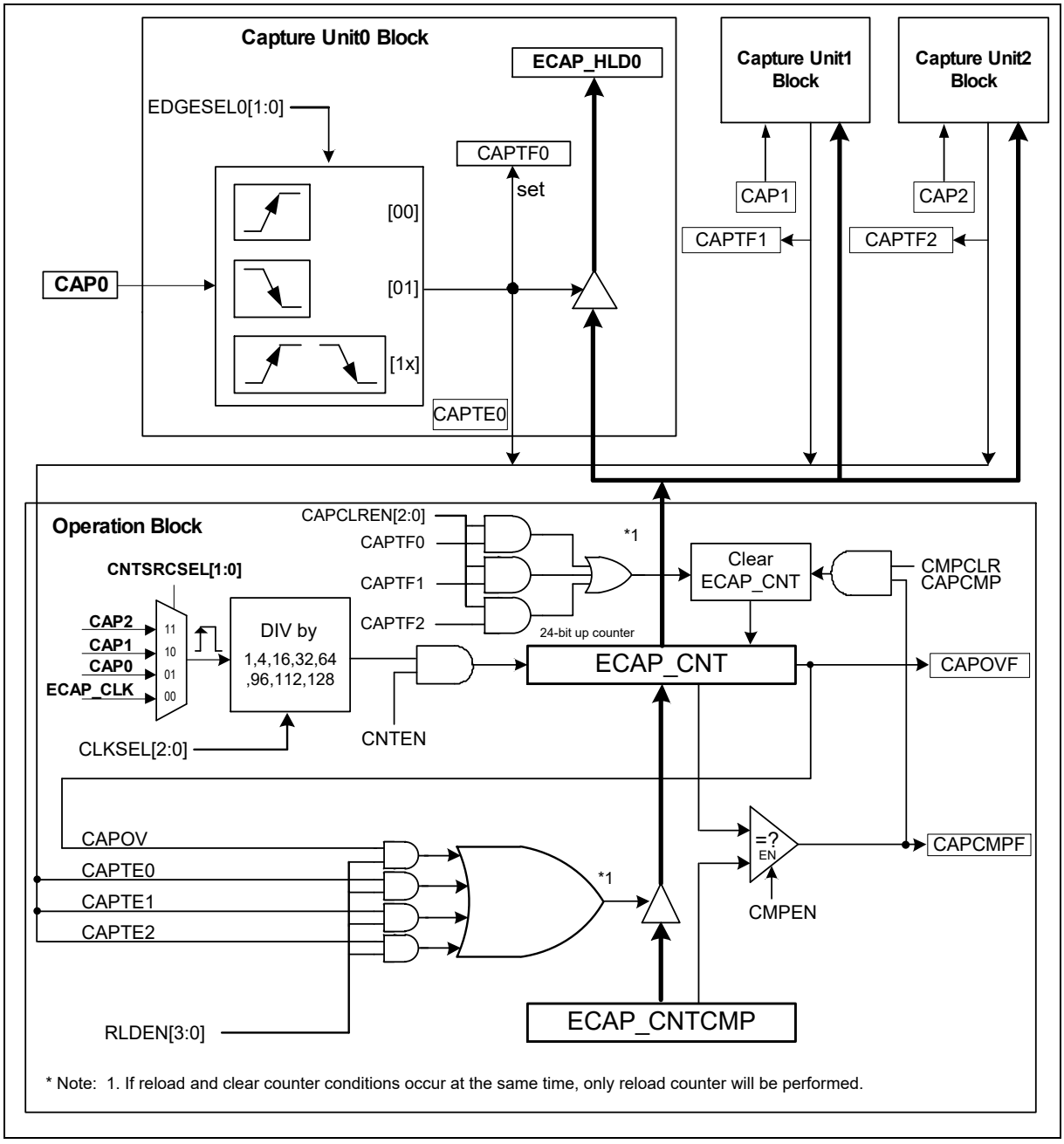


Figure 6.18-4 Input Capture Timer/Counter Function Block

6.18.5.2 Input Capture Timer/Counter Operation

An Input Capture Timer/Counter unit consists of 2 main functional blocks, Capture block and Operation block. There are 3 Input Capture units in Capture block for 3 input channel.

The capture units function as detecting and measuring the pulse width and the period of a square wave. The input channel 0 to 2 have their own edge detectors, which are in Input Capture block but share with one capture timer/counter, ECAP_CNT, which is in Operation block. The edge trigger option is programmable through EDGESEL (ECAP_CTL1[5,4], [3,2], [1,0]) register supporting positive edge, negative edge and both edge triggers. Each capture unit consists of an enable control bit, IC0EN ~ IC2EN (ECAP_CTL0[6:4]) to enable/disable each input channel and a status bit CAP0 ~ CAP2 (ECAP_STATUS[10:8]) to let software monitor the current status of each channel.

The Input Capture supports reload mode and compare mode. For both mode, the capture counter (ECAP_CNT) serves as a 24-bit up-counting counter whose clock comes from the output of the clock divider and is gated with CNTEN, and the clock source of the clock divider, which can be set by CLKSEL[2:0] to divide clock by 1,4,16,32,64,96,112 and 128, is programmable (by setting CNTSRCSEL[1:0]) to be from system clock source, ECAP_CLK or input channel CAP0 ~ CAP2. In reload mode, ECAP_CNTCMP serves as a reload register while in compare mode ECAP_CNTCMP serves as a compare register. The Input Capture Timer/Counter Enable bit (CAPEN) must be set to enable Input Capture Timer/Counter functions. More details of operation are described in the following.

Capture Function

Each time when the capture input detect a valid edge change, it triggers a valid capture event (CAPTE0~2) so that the content of the free running 24-bit capture counter ECAP_CNT will be captured/transferred into the capture hold registers, ECAP_HLD0~2 depending on which channel is triggered. This event also causes the corresponding flag CAPTFx (ECAP_STATUS[2:0]) to be set, which will generate an interrupt if the corresponding interrupt enable bit CAPIENx (ECAP_CTL0[18:16]) is set. Triggered Flags are set by hardware and should be cleared by software. Software can read the register ECAP_STATUS to get the status of flags and has to write 1 to the corresponding bit(s) of ECAP_STATUS to clear flag(s).

In addition, setting the CAPxCLREN(ECAP_CTL1[22:20]) will allow hardware to reset capture counter (ECAP_CNT) automatically whenever the event happens.

Compare Mode

The compare function is enabled by setting the CMPEN (ECAP_CTL0[28]) bit to 1, and ECAP_CNTCMP will serve as a compare register. As ECAP_CNT counting up, upon matching ECAP_CNTCMP value, the flag, CAPCMPF (ECAP_STATUS[4]), will be set, which will generate an interrupt, CMP_INT, if compare interrupt enable bit, CMPIEN (ECAP_CTL0[21]), is set.

Besides, setting the CMPCLR (ECAP_CTL0[25]) will allow hardware to make capture counter cleared to 0 automatically after a compare-match event occurs.

Reload Mode

The Input Capture Timer/Counter can also be configured for reload mode. The reload function is enabled by setting the RLDEN[3:0] bits (ECAP_CTL1[11:8]) – OVRLDEN, CAPxRLDEN – to 1, and each bit enables a reload source.

In this mode, ECAP_CNTCMP serves as a reload register. If OVRLDEN is set, a reload event is generated and causes the content of the ECAP_CNTCMP register to be loaded into the ECAP_CNT register when ECAP_CNT overflows. Furthermore, CAPxRLDEN, x=0~2, are enable bits of making CAPTE2 ~ CAPTE0 as reload sources.

One thing should be noted is that if CAPxCLREN as well as CAPxRLDEN are set, when a valid trigger event (CAPTEx) occurs, only RELAOD function will be executed.

6.18.5.3 Input Capture Timer/Counter Interrupt Architecture

Figure 6.18-5 demonstrates the architecture of Input Capture Timer/Counter interrupt module. There are 5 interrupt sources (OVF_INT, CMP_INT, CAPTF0_INT~CAPTF2_INT), which are logical 'OR' together, in a input capture unit, and each one has an interrupt flag (CAPOVF, CAPCMPF, CAPTF0~CAPTF2), which can trigger Interrupt (ECAP_INT), as well as an the enable control bit (OVIEN, CMPIEN, CAPIEN0~CAPIEN2) to enable/disable the flag.

Note that all the interrupt flags are set by hardware and must be cleared by software by writing 1 to the bit (ECAP_STATUS[5:4],[2:0]) corresponding to the flag.

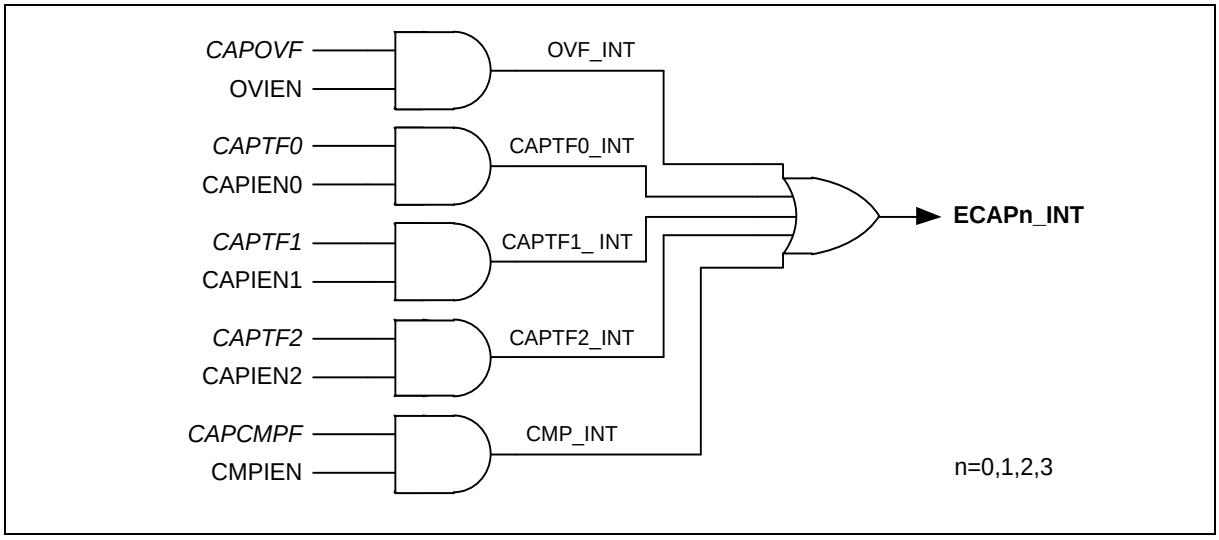


Figure 6.18-5 Input Capture Timer/Counter Interrupt Architecture Diagram

6.18.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
ECAP Base Address: **ECAPn_BA = 0x400B_4000 + (0x0000_1000 * n)** **n=0, 1, 2, 3**				
ECAP_CNT	ECAPn_BA+0x00	R/W	Input Capture Counter (24-bit Up Counter)	0x0000_0000
ECAP_HLD0	ECAPn_BA+0x04	R/W	Input Capture Hold Register 0	0x0000_0000
ECAP_HLD1	ECAPn_BA+0x08	R/W	Input Capture Hold Register 1	0x0000_0000
ECAP_HLD2	ECAPn_BA+0x0C	R/W	Input Capture Hold Register 2	0x0000_0000
ECAP_CNTCMP	ECAPn_BA+0x10	R/W	Input Capture Compare Register	0x0000_0000
ECAP_CTL0	ECAPn_BA+0x14	R/W	Input Capture Control Register 0	0x0000_0000
ECAP_CTL1	ECAPn_BA+0x18	R/W	Input Capture Control Register 1	0x0000_0000
ECAP_STATUS	ECAPn_BA+0x1C	R/W	Input Capture Status Register	0x0000_0000

6.18.7 Register Description

Input Capture Counter (ECAP_CNT)

Register	Offset	R/W	Description	Reset Value
ECAP_CNT	ECAPn_BA+0x00	R/W	Input Capture Counter (24-bit Up Counter)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CNT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CNT**	**Input Capture Timer/Counter** The input Capture Timer/Counter is a 24-bit up-counting counter. The clock source for the counter is from the clock divider.

Input Capture Counter Hold Register (ECAP_HLD0~2)

Register	Offset	R/W	Description	Reset Value
ECAP_HLD0	ECAPn_BA+0x04	R/W	Input Capture Hold Register 0	0x0000_0000
ECAP_HLD1	ECAPn_BA+0x08	R/W	Input Capture Hold Register 1	0x0000_0000
ECAP_HLD2	ECAPn_BA+0x0C	R/W	Input Capture Hold Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
HOLD							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
HOLD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HOLD							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**HOLD**	**Input Capture Counter Hold Register** When an active input capture channel detects a valid edge signal change, the ECAPCNT value is latched into the corresponding holding register. Each input channel has its own holding register named by ECAP_HLDx where x is from 0 to 2 to indicate inputs from IC0 to IC2, respectively.

Input Capture Counter Compare Register (ECAP_CNTCMP)

Register	Offset	R/W	Description	Reset Value
ECAP_CNTCMP	ECAPn_BA+0x10	R/W	Input Capture Compare Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CNTCMP							
15	14	13	12	11	10	9	8
CNTCMP							
7	6	5	4	3	2	1	0
CNTCMP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CNTCMP**	**Input Capture Counter Compare Register** If the compare function is enabled (CMPEN = 1), this register (ECAP_CNTCMP) is used to compare with the capture counter (ECAP_CNT). If the reload control is enabled (RLDEN[n] = 1, n=0~3), an overflow event or capture events will trigger the hardware to load the value of this register (ECAP_CNTCMP) into ECAP_CNT.

Input Capture Timer/Counter Control Register (ECAP_CTL0)

Register	Offset	R/W	Description	Reset Value
ECAP_CTL0	ECAPn_BA+0x14	R/W	Input Capture Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CAPEN	CMPEN	Reserved		CMPCLREN	CNTEN
23	22	21	20	19	18	17	16
Reserved		CMPIEN	OVIEN	Reserved	CAPIEN2	CAPIEN1	CAPIEN0
15	14	13	12	11	10	9	8
Reserved		CAPSEL2		CAPSEL1		CAPSEL0	
7	6	5	4	3	2	1	0
Reserved	IC2EN	IC1EN	IC0EN	CAPNFDIS	NFCLKSEL		

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**CAPEN**	**Input Capture Timer/Counter Enable Bit** 0 = Input Capture function Disabled. 1 = Input Capture function Enabled.
[28]	**CMPEN**	**Compare Function Enable Bit** The compare function in input capture timer/counter is to compare the dynamic counting ECAP_CNT with the compare register ECAP_CNTCMP, if ECAP_CNT value reaches ECAP_CNTCMP, the flag CAPCMPF will be set. 0 = The compare function Disabled. 1 = The compare function Enabled.
[25]	**CMPCLREN**	**Input Capture Counter Cleared by Compare-match Control** If this bit is set to 1, the capture counter (ECAP_CNT) will be cleared to 0 when the compare-match event (CAPCMPF = 1) occurs. 0 = Compare-match event (CAPCMPF) can clear capture counter (ECAP_CNT) Disabled. 1 = Compare-match event (CAPCMPF) can clear capture counter (ECAP_CNT) Enabled.
[24]	**CNTEN**	**Input Capture Counter Start Counting Control** Setting this bit to 1, the capture counter (ECAP_CNT) starts up-counting synchronously with the clock from the . 0 = ECAP_CNT stop counting. 1 = ECAP_CNT starts up-counting.
[23:22]	**Reserved**	Reserved.
[21]	**CMPIEN**	**CAPCMPF Trigger Input Capture Interrupt Enable Bit** 0 = The flag CAPCMPF can trigger Input Capture interrupt Disabled. 1 = The flag CAPCMPF can trigger Input Capture interrupt Enabled.
[20]	**OVIEN**	**CAPOVF Trigger Input Capture Interrupt Enable Bit** 0 = The flag CAPOVF can trigger Input Capture interrupt Disabled. 1 = The flag CAPOVF can trigger Input Capture interrupt Enabled.

Bits	Description	
[19]	**Reserved**	Reserved.
[18]	**CAPIEN2**	**Input Capture Channel 2 Interrupt Enable Bit** 0 = The flag CAPTF2 can trigger Input Capture interrupt Disabled. 1 = The flag CAPTF2 can trigger Input Capture interrupt Enabled.
[17]	**CAPIEN1**	**Input Capture Channel 1 Interrupt Enable Bit** 0 = The flag CAPTF1 can trigger Input Capture interrupt Disabled. 1 = The flag CAPTF1 can trigger Input Capture interrupt Enabled.
[16]	**CAPIEN0**	**Input Capture Channel 0 Interrupt Enable Bit** 0 = The flag CAPTF0 can trigger Input Capture interrupt Disabled. 1 = The flag CAPTF0 can trigger Input Capture interrupt Enabled.
[15:14]	**Reserved**	Reserved.
[13:12]	**CAPSEL2**	**CAP2 Input Source Selection** 00 = CAP2 input is from port pin ICAP2. 01 = Reserved. 10 = CAP2 input is from signal CHX of QEI controller unit n. 11 = Reserved. **Note:** Input capture unit n matches QEIn, where n = 0~1.
[11:10]	**CAPSEL1**	**CAP1 Input Source Selection** 00 = CAP1 input is from port pin ICAP1. 01 = CAP1 input is from ACMP1 output. 10 = CAP1 input is from signal CHB of QEI controller unit n. 11 = Reserved. **Note:** Input capture unit n matches QEIn, where n = 0~1.
[9:8]	**CAPSEL0**	**CAP0 Input Source Selection** 00 = CAP0 input is from port pin ICAP0. 01 = CAP0 input is from ACMP0 output. 10 = CAP0 input is from signal CHA of QEI controller unit n. 11 = Reserved. **Note:** Input capture unit n matches QEIn, where n = 0~1.
[7]	**Reserved**	Reserved.
[6]	**IC2EN**	**Port Pin IC2 Input to Input Capture Unit Enable Bit** 0 = IC2 input to Input Capture Unit Disabled. 1 = IC2 input to Input Capture Unit Enabled.
[5]	**IC1EN**	**Port Pin IC1 Input to Input Capture Unit Enable Bit** 0 = IC1 input to Input Capture Unit Disabled. 1 = IC1 input to Input Capture Unit Enabled.
[4]	**IC0EN**	**Port Pin IC0 Input to Input Capture Unit Enable Bit** 0 = IC0 input to Input Capture Unit Disabled. 1 = IC0 input to Input Capture Unit Enabled.
[3]	**CAPNFDIS**	**Input Capture Noise Filter Disable Bit** 0 = Noise filter of Input Capture Enabled. 1 = Noise filter of Input Capture Disabled (Bypass).

Bits	Description	
[2:0]	**NFCLKSEL**	**Noise Filter Clock Pre-divide Selection** To determine the sampling frequency of the Noise Filter clock 000 = ECAP_CLK. 001 = ECAP_CLK/2. 010 = ECAP_CLK/4. 011 = ECAP_CLK/16. 100 = ECAP_CLK/32. 101 = ECAP_CLK/64.

Input Capture Timer/Counter Control Register (ECAP_CTL1)

Register	Offset	R/W	Description	Reset Value
ECAP_CTL1	ECAPn_BA+0x18	R/W	Input Capture Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	CAP2CLREN	CAP1CLREN	CAP0CLREN	Reserved		CNTSRCSEL	
15	14	13	12	11	10	9	8
Reserved	CLKSEL			OVRLDEN	CAP2RLDEN	CAP1RLDEN	CAP0RLDEN
7	6	5	4	3	2	1	0
Reserved		EDGESEL2		EDGESEL1		EDGESEL0	

Bits	Description	
[31:18]	**Reserved**	Reserved.
[22]	**CAP2CLREN**	**Capture Counter Cleared by Capture Event2 Control** 0 = Event CAPTE2 can clear capture counter (ECAP_CNT) Disabled. 1 = Event CAPTE2 can clear capture counter (ECAP_CNT) Enabled.
[21]	**CAP1CLREN**	**Capture Counter Cleared by Capture Event1 Control** 0 = Event CAPTE1 can clear capture counter (ECAP_CNT) Disabled. 1 = Event CAPTE1 can clear capture counter (ECAP_CNT) Enabled.
[20]	**CAP0CLREN**	**Capture Counter Cleared by Capture Event0 Control** 0 = Event CAPTE0 can clear capture counter (ECAP_CNT) Disabled. 1 = Event CAPTE0 can clear capture counter (ECAP_CNT) Enabled.
[17:16]	**CNTSRCSEL**	**Capture Timer/Counter Clock Source Selection** Select the capture timer/counter clock source. 00 = ECAP_CLK (default). 01 = CAP0. 10 = CAP1. 11 = CAP2.
[15]	**Reserved**	Reserved.

Bits	Description	
[14:12]	**CLKSEL**	**Capture Timer Clock Divide Selection** The capture timer clock has a pre-divider with eight divided options controlled by CLKSEL[2:0]. 000 = ECAP_CLK/1. 001 = ECAP_CLK/4. 010 = ECAP_CLK/16. 011 = ECAP_CLK/32. 100 = ECAP_CLK/64. 101 = ECAP_CLK/96. 110 = ECAP_CLK/112. 111 = ECAP_CLK/128.
[11]	**Reserved**	Reserved.
[11]	**OVRLDEN**	**Capture Counter's Reload Function Triggered by Overflow Enable Bit** 0 = The reload triggered by CAPOV Disabled. 1 = The reload triggered by CAPOV Enabled.
[10]	**CAP2RLDEN**	**Capture Counter's Reload Function Triggered by Event CAPTE2 Enable Bit** 0 = The reload triggered by Event CAPTE2 Disabled. 1 = The reload triggered by Event CAPTE2 Enabled.
[9]	**CAP1RLDEN**	**Capture Counter's Reload Function Triggered by Event CAPTE1 Enable Bit** 0 = The reload triggered by Event CAPTE1 Disabled. 1 = The reload triggered by Event CAPTE1 Enabled.
[8]	**CAP0RLDEN**	**Capture Counter's Reload Function Triggered by Event CAPTE0 Enable Bit** 0 = The reload triggered by Event CAPTE0 Disabled. 1 = The reload triggered by Event CAPTE0 Enabled.
[7:6]	**Reserved**	Reserved.
[5:4]	**EDGESEL2**	**Channel 2 Captured Edge Selection** Input capture2 can detect falling edge change only, rising edge change only or both edge changes 00 = Detect rising edge only. 01 = Detect falling edge only. 1x = Detect both rising and falling edge.
[3:2]	**EDGESEL1**	**Channel 1 Captured Edge Selection** Input capture1 can detect falling edge change only, rising edge change only or both edge change 00 = Detect rising edge only. 01 = Detect falling edge only. 1x = Detect both rising and falling edge.

Bits	Description	
[1:0]	**EDGESEL0**	**Channel 0 Captured Edge Selection** Input capture0 can detect falling edge change only, rising edge change only or both edge change 00 = Detect rising edge only. 01 = Detect falling edge only. 1x = Detect both rising and falling edge.

Input Capture Timer/Counter Status Register (ECAP_STATUS)

Register	Offset	R/W	Description	Reset Value
ECAP_STATUS	ECAPn_BA+0x1C	R/W	Input Capture Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					CAP2	CAP1	CAP0
7	6	5	4	3	2	1	0
Reserved		CAPOVF	CAPCMPF	Reserved	CAPTF2	CAPTF1	CAPTF0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[8]	**CAP2**	**Value of Input Channel 2, CAP2 (Read Only)** Reflecting the value of input channel 2, CAP2. **Note:** The bit is read only and write is ignored.
[7]	**CAP1**	**Value of Input Channel 1, CAP1 (Read Only)** Reflecting the value of input channel 1, CAP1 **Note:** The bit is read only and write is ignored.
[6]	**CAP0**	**Value of Input Channel 0, CAP0 (Read Only)** Reflecting the value of input channel 0, CAP0 **Note:** The bit is read only and write is ignored.
[5]	**CAPOVF**	**Input Capture Counter Overflow Flag** Flag is set by hardware when counter (ECAP_CNT) overflows from 0x00FF_FFFF to 0. 0 = No overflow event has occurred since last clear. 1 = Overflow event(s) has/have occurred since last clear. **Note:** This bit is only cleared by writing 1 to it.
[4]	**CAPCMPF**	**Input Capture Compare-match Flag** If the input capture compare function is enabled, the flag is set by hardware when capture counter (ECAP_CNT) up counts and reaches the ECAP_CNTCMP value. 0 = ECAP_CNT has not matched ECAP_CNTCMP value since last clear. 1 = ECAP_CNT has matched ECAP_CNTCMP value at least once since last clear. **Note:** This bit is only cleared by writing 1 to it.
[3]	**Reserved**	Reserved.

Bits	Description	
[2]	**CAPTF2**	**Input Capture Channel 2 Triggered Flag** When the input capture channel 2 detects a valid edge change at CAP2 input, it will set flag CAPTF2 to high. 0 = No valid edge change has been detected at CAP2 input since last clear. 1 = At least a valid edge change has been detected at CAP2 input since last clear. **Note:** This bit is only cleared by writing 1 to it.
[1]	**CAPTF1**	**Input Capture Channel 1 Triggered Flag** When the input capture channel 1 detects a valid edge change at CAP1 input, it will set flag CAPTF1 to high. 0 = No valid edge change has been detected at CAP1 input since last clear. 1 = At least a valid edge change has been detected at CAP1 input since last clear. **Note:** This bit is only cleared by writing 1 to it.
[0]	**CAPTF0**	**Input Capture Channel 0 Triggered Flag** When the input capture channel 0 detects a valid edge change at CAP0 input, it will set flag CAPTF0 to high. 0 = No valid edge change has been detected at CAP0 input since last clear. 1 = At least a valid edge change has been detected at CAP0 input since last clear. **Note:** This bit is only cleared by writing 1 to it.

6.19 USCI - Universal Serial Control Interface Controller (USCI)

6.19.1 Overview

The Universal Serial Control Interface (USCI) is a flexible interface module covering several serial communication protocols. The user can configure this controller as UART, SPI, or I²C functional protocol.

6.19.2 Features

The controller can be individually configured to match the application needs. The following protocols are supported:

- UART
- SPI
- I²C

6.19.3 Block Diagram

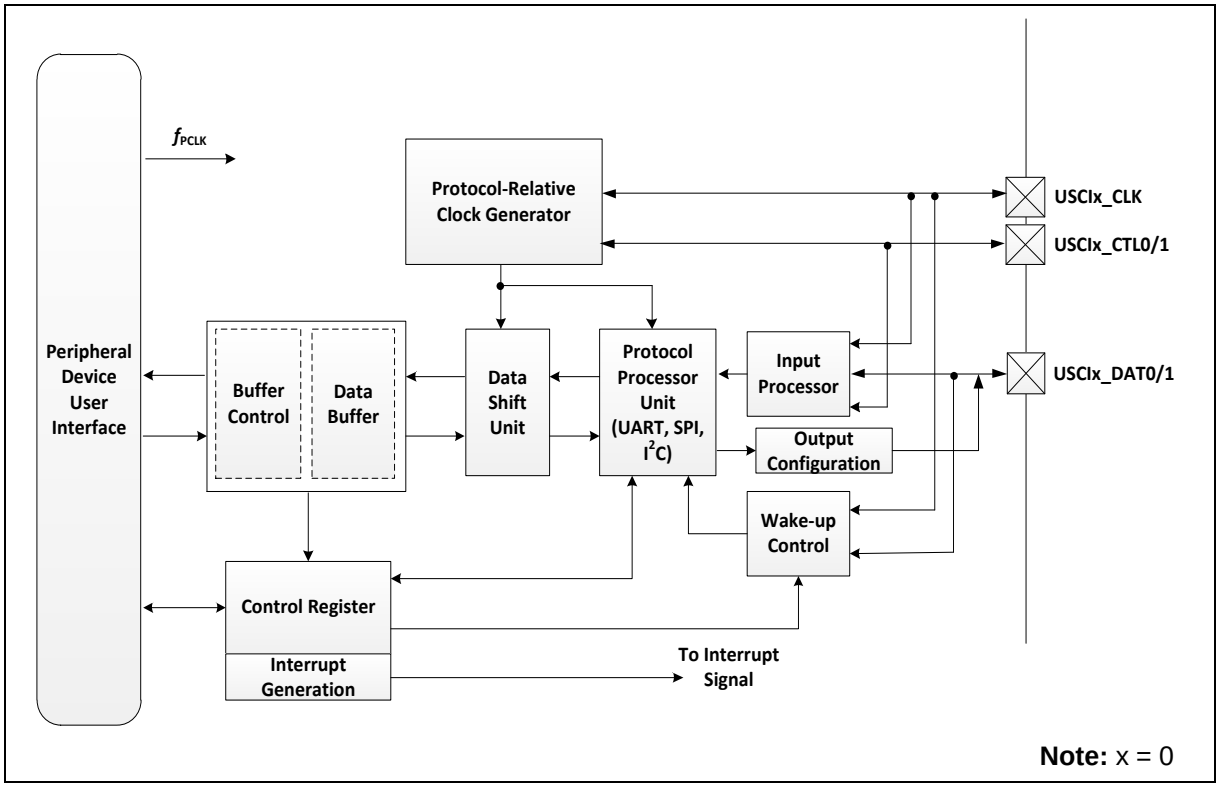


Figure 6.19-1 USCI Block Diagram

6.19.4 Functional Description

The structure of the Universal Serial Control Interface (USCI) controller is shown in Figure 6.19-1 USCI Block Diagram. The input signal is implemented in input processor. The data buffers and the data shift unit support the data transfers. Each protocol-specific function is handled by the protocol processor unit. The timing and time event control signals of the specific protocol are handled by the protocol-relative clock generator. All the protocol-specific events are processed in the interrupt generation unit. The wake-up function of the specific protocol is implemented in the wake-up control unit.

The USCI is equipped with three protocols including UART, SPI, and I²C. They can be selected by FUNMODE (USCI_CTL [2:0]). Note that the FUNMODE must be set to 0 before changing protocol.

6.19.4.1 I/O Processer

Input Signal

All input stages offer the similar feature set. They are used for all protocols.

Table 6.19-1lists the relative input signals for each selected protocol. Each input signal is handled by an input processor for signal conditioning, such as signal inverse selection control, or a digital input filter.

Selected Protocol		UART	SPI	I²C
Serial Bus Clock Input	USCIx_CLK	-	SPI_CLK	SCL
Control Input	USCIx_CTL0	nCTS	SPI_SS	-
	USCIx_CTL1	-	-	-
Data Input	USCIx_DAT0	RX	SPI_MOSI_0	SDA
	USCIx_DAT1	-	SPI_MISO_0	-

Table 6.19-1 Input Signals for Different Protocols

The description of protocol-specific items are given in the related protocol chapters.

General Input Structure

The input structures of data and control signals include inverter, digital filter and edge detection (data signal only).

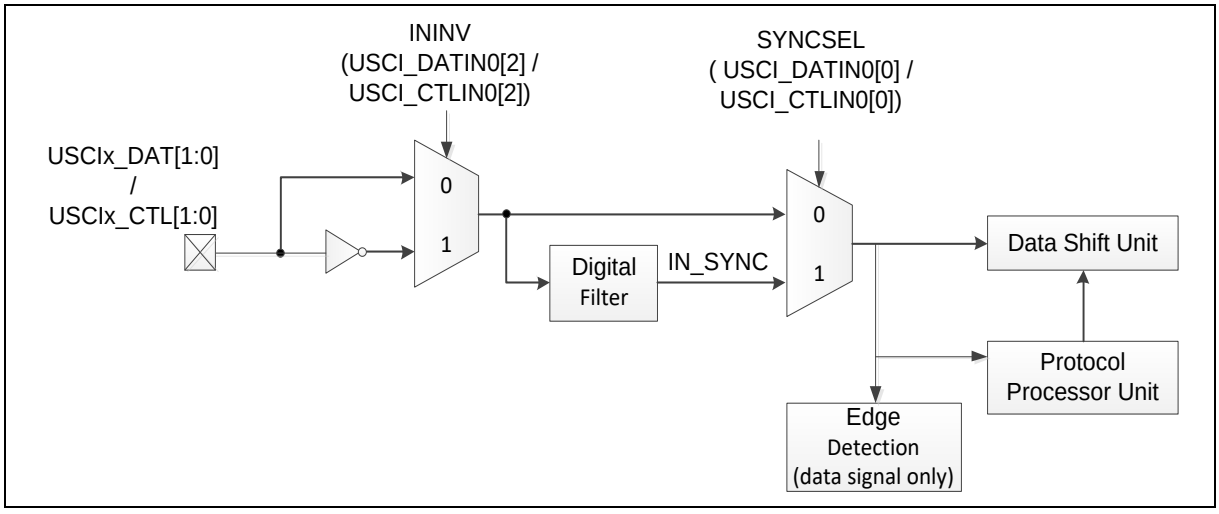


Figure 6.19-2 Input Conditioning for USCIx_DAT[1:0] and USCIx_CTL[1:0]

The input structure of USCIx_CLK is similar to USCIx_CTL[1:0] input structure, except it does not support inverse function.

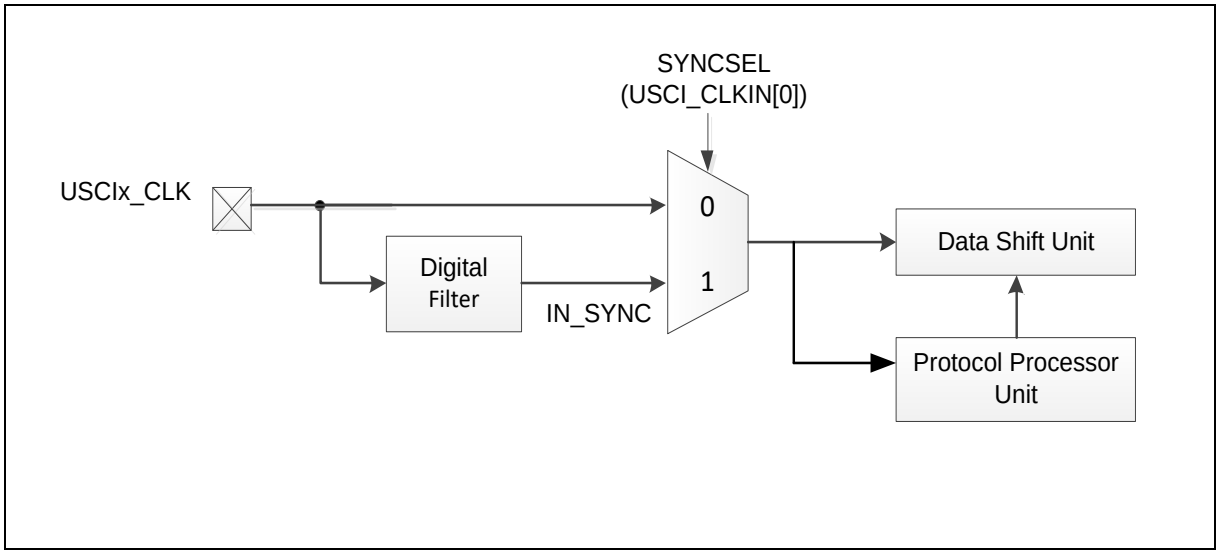


Figure 6.19-3 Input Conditioning for USCIx_CLK

All configurations of control, clock and data input structures are in USCI_CTLIN0, USCI_CLKIN and USCI_DATIN0 registers respectively. EDGEDET (USCI_DATIN0[4:3]) is used to select the edge detection condition. Note that the EDGEDET for USCI_DATIN0 must be set 2'b10 in UART mode. The programmable edge detection indicates that the desired event has occurred by activating the trigger signal.

ININV (USCI_DATIN0[2] / USCI_CTLIN0[2]) allows a polarity inversion of the selected input signal to adapt the input signal polarity to the internal polarity of the data shift unit and the protocol state machine.

If the SYNCSEL (USCI_DATIN0[0] / USCI_CTLIN0[0] / USCI_CLKIN[0]) is set to 0, the paths of input signals do not contain any delay due to synchronization or filtering. If there is noise on the input signals, there is the possibility to synchronize the input signal (signal IN_SYNC is synchronized to f_{PCLK}). The synchronized input signal is taken into account by SYNCSEL = 1. The synchronization leads to a delay in the signal path of 2-3 times the period of f_{PCLK}.

Output Signals

Table 6.19-2 shows the relative output signals for each protocol. The number of actually used outputs depends on the selected protocol and they can be classified according to their meaning for the protocols.

Selected Protocol		UART	SPI	I²C
Serial Bus Clock Output	USCIx_CLK	-	SPI_CLK	SCL
Control Output	USCIx_CTL0	-	SPI_SS	-
	USCIx_CTL1	nRTS	-	-
Data Output	USCIx_DAT0	-	SPI_MOSI_0	SDA
	USCIx_DAT1	TX	SPI_MISO_0	-

Table 6.19-2 Output Signals for Different Protocols

The description of protocol-specific items are given in the related protocol chapters.

6.19.4.2 Data Buffering

The data handling of the USCI controller is based on a Data Shift Unit (DSU) and a buffer structure. Both of the data shift and buffer registers are 16-bit wide. The inputs of Data Shift Unit include the shift data, the serial bus clock, and the shift control. The output pin of transmission can be USCIx_DAT0 pin or USCIx_DAT1 pin depends on what protocol is selected.

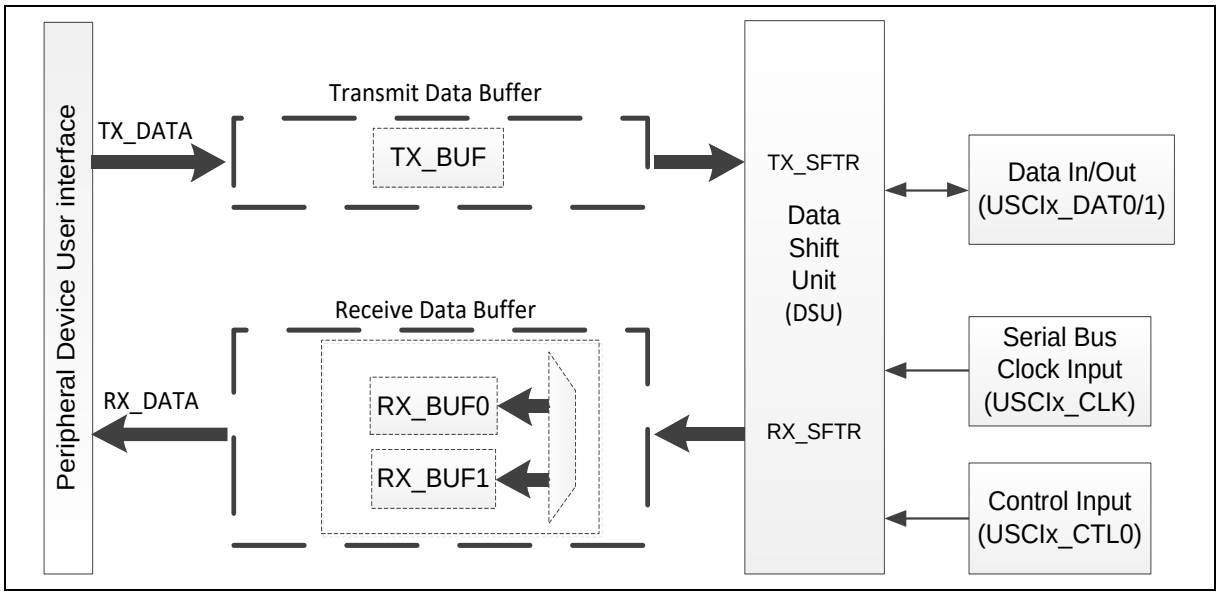


Figure 6.19-4 Block Diagram of Data Buffering

The operation of data handling includes:

- The peripheral device user interface (APB) is used to handle data, interrupts, status and control information.
- A transmitter includes transmit shift register (TX_SFTR) and a transmit data buffer (TX_BUF). The TXFULL / TXEMPTY (USCI_BUFSTS[9:8]) and TXENDIF (USCI_PROTSTS[2]) can indicate the status of transmitter.
- A receiver includes receive shift register (RX_SFTR) and a double receive buffer structure (RX_BUF0, RX_BUF1). In double buffer structure, user need not care about the reception sequence and two received data can be hold if user does not read the data of USCI_RXDAT register in time.

Data Access Structure

The Data Access Structure includes read access to received data and write access of data to be transmitted. The received data is stored in the receiver buffers including RX_BUF0 and RX_BUF1. User need not care about the reception sequence. The receive buffer can be accessed by reading USCI_RXDAT register. The first received data is read out first and the next received data becomes visible in USCI_RXDAT and can be read out next.

Transmit data can be loaded to TX_BUF by writing to the transmit register USCI_TXDAT.

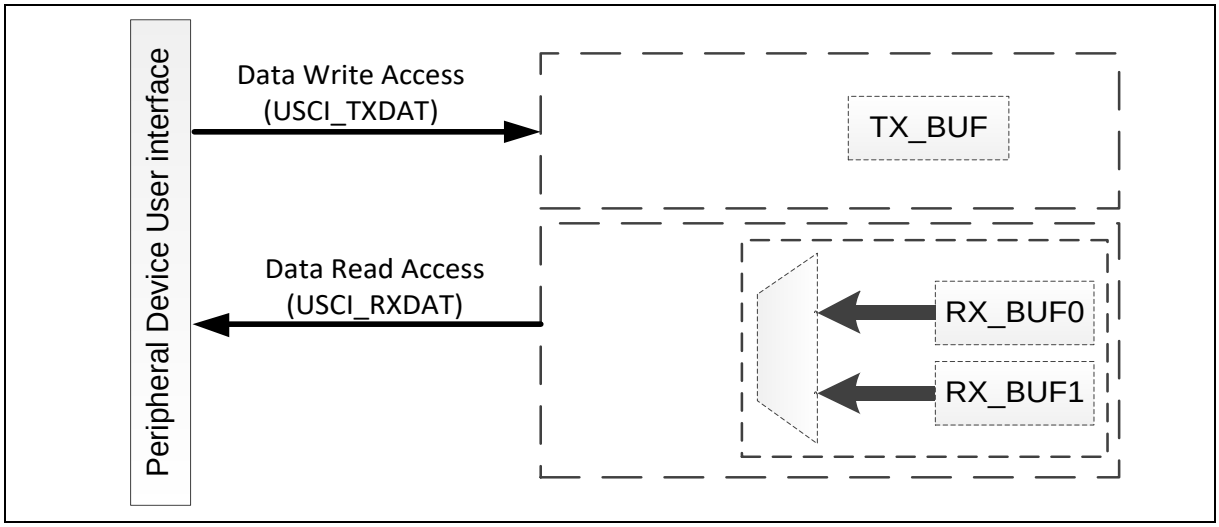


Figure 6.19-5 Data Access Structure

Transmit Data Path

The transmit data path is based on 16-bit wide transmit shift register (TX_SFTR) and transmit buffer TX_BUF. The data transfer parameters like data word length is controlled commonly for transmission and reception by the line control register USCI_LINECTL.

Transmit Buffering

The transmit shift register cannot be directly accessed by user. It is updated automatically with the value stored in the transmit buffer (TX_BUF) if a currently transmitted data is finished and new data is valid for transmission.

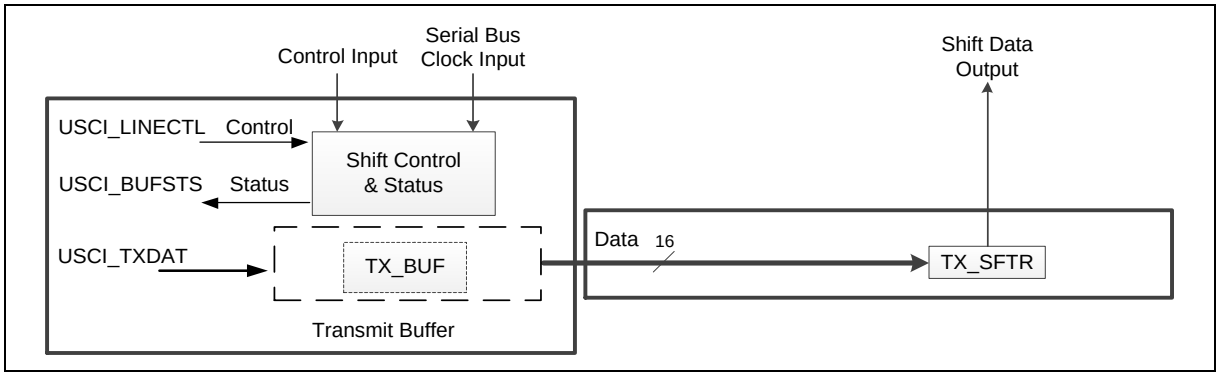


Figure 6.19-6 Transmit Data Path

Transmit Data Validation

The status of TXEMPTY (USCI_BUFSTS[8]) indicates the transmission data is valid or not in the transmit buffer (TX_BUF) and the TXSTIF (USCI_PROTSTS[1]) labels the start conditions for each data.

- If the USCI controller is a Master, the data transfer can only be started with valid data in the transmit buffer (TX_BUF). In this case, the transmit shift register is loaded with the content of transmit buffer.

  Note: Master defines the start of data transfer.

- If the USCI controller is a Slave, a data transfer requested by Master and it has to be started independently of the status in transmit buffer (TX_BUF). If a data transfer is requested and started by the Master, the transmit shift register is loaded from specific protocol control signal if it is valid for transmission.

Note: Slave can not define the start itself, but has to react.

- The timing of loading data from transmit buffer to data shift unit depends on protocol configurations.
- **UART:** A transmission of the data word in transmit buffer can be started if TXEMPTY = 0 in normal operation. In auto flow control, A transmission of the data word in transmit buffer can be started while TXEMPTY = 0 and USCIx_CTL0 in active stage.
- **SPI:** In Master mode, data transmission will be started when TXEMPTY (USCI_BUFSTS[8]) is 0. In Slave mode, the data transmission can be started only when slave selection signal is at active state and clock is presented on USCIx_CLK pin.
- **I²C:** A transmission of the data byte in transmit buffer can be started if TXEMPTY = 0.
- A transmission data which is located in transmit buffer can be started if the TXEMPTY (USCI_BUFSTS [8]) = 0. The content of the transmit buffer (in TX_BUF condition) should not be overwritten with new data while it is valid for transmission and a new transmission can start. If the content of TX_BUF has to be changed, user can set TXRST (USCI_BUFCTL [16]) to 1 to clear the content of TX_BUF before updating the data. Moreover, TXEMPTY (USCI_BUFSTS [8]) will be cleared automatically when transmit buffer (TX_BUF) is updated with new data. While a transmission is in progress, TX_BUF can be loaded with new data. User has to update the TX_BUF before a new transmission.

Receive Data Path

The receive data path is based on 16-bit wide receive shift register RX_SFTR and receive buffers RX_BUF0 and RX_BUF1. The data transfer parameters like data word length, or the shift direction are controlled commonly for transmission and reception by the line control register USCI_LINECTL. Register USCI_BUFSTS monitors the data validation of USCI_RXDAT.

Receive Buffering

The receive shift register cannot be directly accessed by user, but its content is automatically loaded into the receive buffer if a complete data word has been received or the frame is finished. The received data words in Receive Buffer can be read out automatically from register USCI_RXDAT.

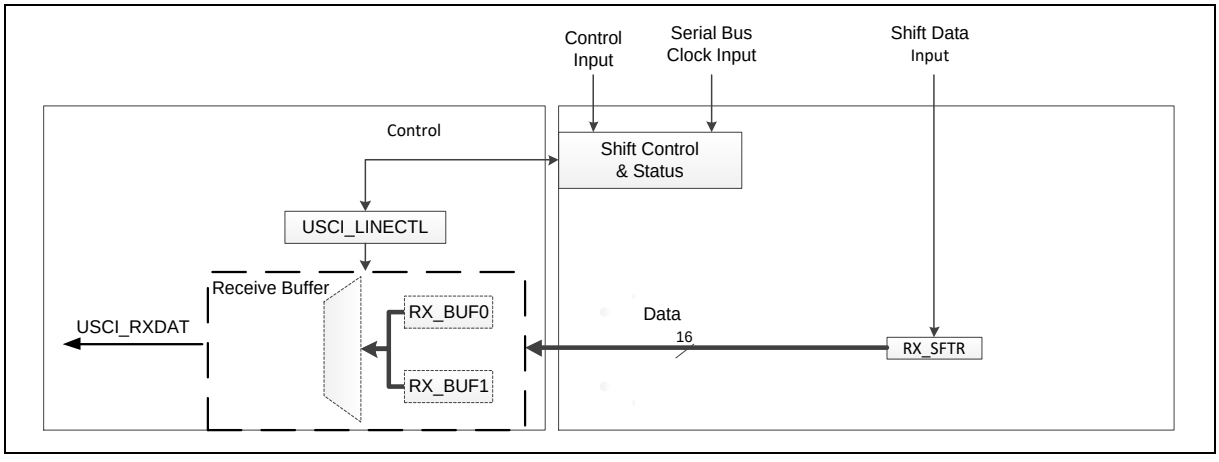


Figure 6.19-7 Receive Data Path

6.19.4.3 Port Direction Control

In SPI protocol with half-duplex configurations, the data port is bidirectional. Port direction control is intended to control the pin direction through a dedicated hardware interface.

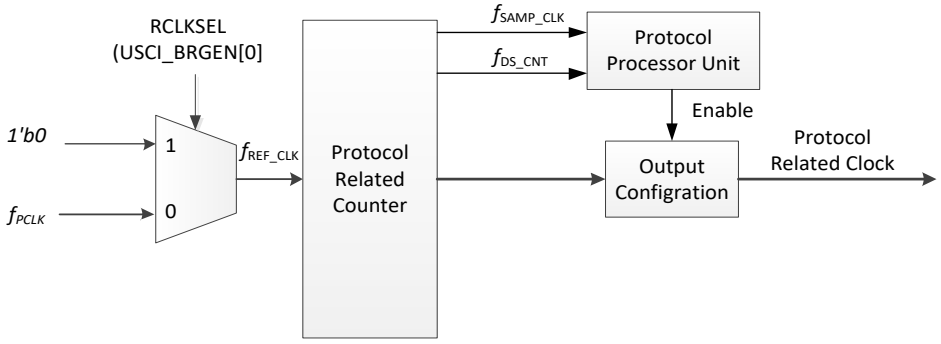
The direction of selected pin is controlled by PORTDIR (USCI_TXDAT[16]). When user writes USCI_TXDAT register, the transmit data and its port direction are settled simultaneously.

6.19.4.4 Protocol Control and Status

The protocol-related control and status information are located in the protocol control register USCI_PROTCTL and in the protocol status register USCI_PROTSTS. These registers are shared between the available protocols. As a consequence, the meaning of the bit positions in these registers is different within the protocols. Refer to each protocol's relative register for detail information.

6.19.4.5 Protocol-Relative Clock Generator

The USCI controller contains a protocol-relative clock generator and it is controlled by register USCI_BRGEN. It is reset when the USCI_BRGEN register is written. The structured of protocol-relative clock generator is shown below.



Note: Refer the **Basic Clock Divider Counter** section to get the f_{SAMP_CLK}

Figure 6.19-8 Protocol-Relative Clock Generator

The protocol related counter contains basic clock divider counter and timing measurement counter. It is based on a divider stages, providing the frequencies needed for the different protocols. It contains:

- The basic clock divider counter provides the protocol relative clock signal and other protocol-related signals (f_{SAMP_CLK} and f_{DS_CLK}).
- The timing measurement counter for time interval measurement, e.g. baud rate detection on UART protocol.
- The output signals of protocol relative clock generator can be made available on pins (e.g USCIx_CLK for SPI).

Basic Clock Divider Counter

The basic clock divider counter is used for an integer division delivering f_{REF_CLK2}, f_{REF_CLK}, f_{DIV_CLK}, f_{SCLK}, and f_{SAMP_CLK} . The frequencies of this divider are controlled by PTCLKSEL (USCI_BRGEN [1]), CLKDIV (USCI_BRGEN [25:16]), SPCLKSEL (USCI_BRGEN [3:2]).

The basic clock divider counter is used to generate the relative protocol timing signals.

$$f_{\mathrm{DIV_CLK}} = f_{REF_CLK} \times \frac{1}{\mathrm{CLKDIV} + 1} \text{ if } \mathrm{PTCLKSEL} = 0$$

$$f_{\mathrm{DIV_CLK}} = f_{REF_CLK} \times \frac{1}{(\mathrm{CLKDIV} + 1) \times 2} \text{ if } \mathrm{PTCLKSEL} = 1$$

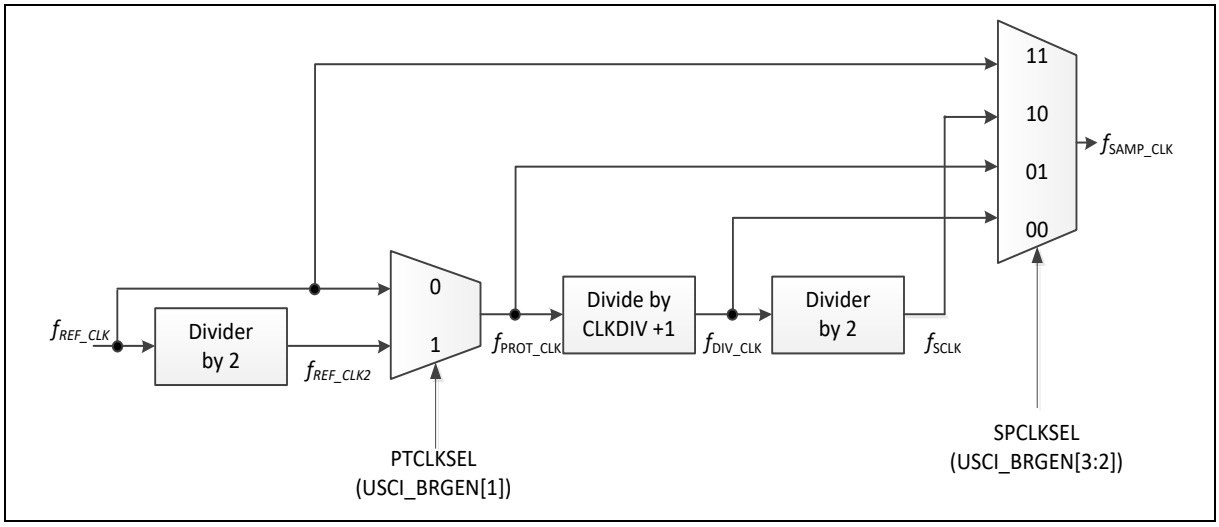


Figure 6.19-9 Basic Clock Divider Counter

Timing Measurement Counter

The timing measurement counter is used for time interval measurement and is enabled by TMCNTEN (USCI_BRGEN [4]) = 1. When TMCNTSRC (USCI_BRGEN [5]) is set to 1, the timer works on f_{DIV_CLK}, otherwise, the timer works independently from f_{PROT_CLK}. Therefore, any serial data reception or transmission can continue while the timer is performing timing measurements. The timer counts the length of protocol-related signals with f_{PROT_CLK} or f_{DIV_CLK}. It stops counting when it reaches the user-specified value.

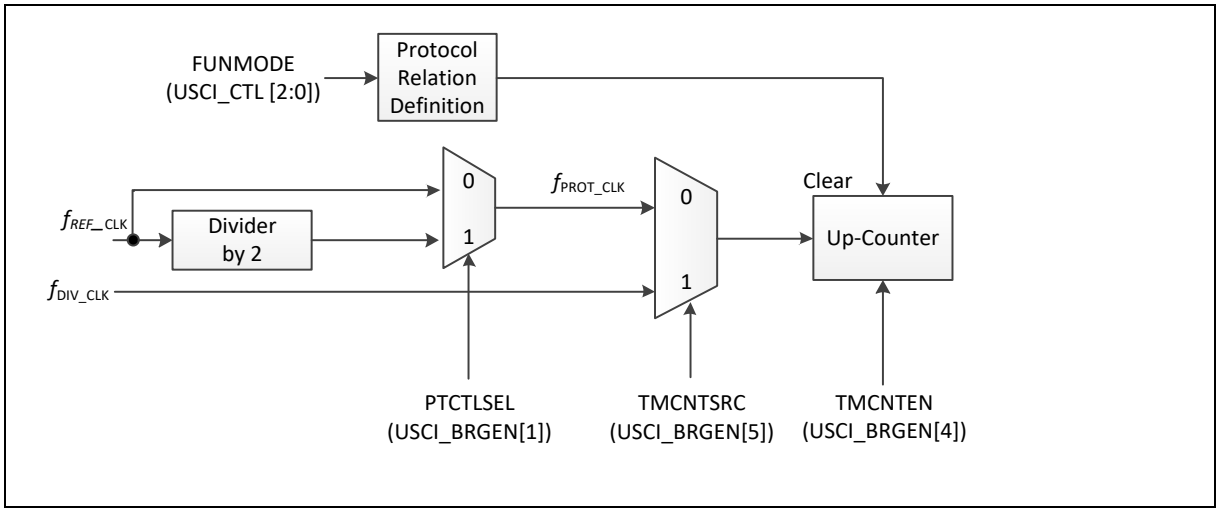


Figure 6.19-10 Block of Timing Measurement Counter

The timing measurement counter is used to perform time-out function or auto-baud rate mechanism. Its functionality depends on the selected protocol as shown below.

- UART: The timing measurement counter is used in auto baud rate detection.
- SPI: The timing measurement counter is used for counting the slave time-out period.
- I^2C: The timing measurement counter indicates time-out clock cycle.

Sample Time Counter

A sample time counter associated to the protocol related counter defining protocol specific timings, such shift control signals or bit timings, based on the input frequency f_{SAMP_CLK}. The sample time counter

allows generating time intervals for protocol-specific purposes. The period of a sample frequency f_{PDS_CNT} is given by the selected input frequency f_{SAMP_CLK} and the programmed pre-divider value (PDSCNT (USCI_BRGEN [9:8])). The meaning of the sample time depends on the selected protocol. Please refer to the corresponding chapters for more protocol-specific information.

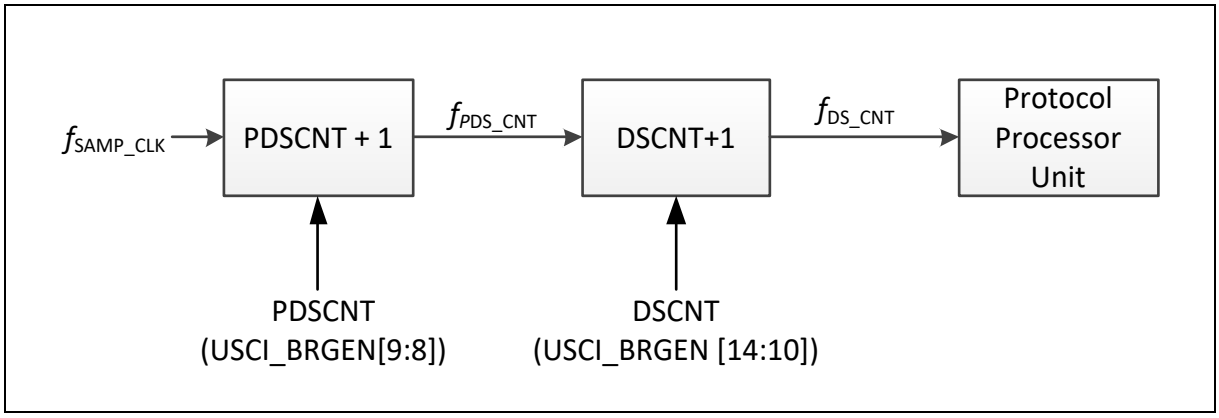


Figure 6.19-11 Sample Time Counter

6.19.4.6 Data Transfer Events and Interrupts

The data transfer events are based on the transmission or reception of a data word. The related indication flags are located in register USCI_PROTSTS. All events can be individually enabled for interrupt generation. If the FUNMODE (USCI_CTL [2:0]) is set to 0, the USCI is disabled. When FUNMODE (USCI_CTL [2:0]) is setting for a protocol port, the internal states will be controlled by logic hardware of the selected protocol.

- Transmit start interrupt event to indicate that a data word has been started:

  A transmit start interrupt event occurs when the data is loaded into transmitted shift register. It is indicated by flag TXSTIF (USCI_PROTSTS [1]) and, if enabled, leads to transmit start interrupt.

- Transmit end interrupt event to indicate that a data word transmission has been done:

  A transmit end interrupt event occurs when the current transmit data in shift register had been finished. It is indicated by flag TXENDIF (USCI_PROTSTS [2]) and, if enabled, leads to transmit end interrupt. This event also indicates when the shift control settings (word length, shift direction, etc.) are internally “frozen” for the current data word transmission. In UART and I^2C mode, the transmit data valid is according to TXEMPTY (USCI_BUFSTS [8]) and protocol relative internal signal with the transmit end interrupt event.

- Receiver start event to indicate that a data word reception has started:

  When the receive clock edge that shifts in the first bit of a new data word is detected and reception is enabled, a receiver start event occurs. It is indicated by flag RXSTIF (USCI_PROTSTS [3]) and, if enabled, leads to receiver start interrupt.

- Receive event to indicate that a data word has been received:

  If a new received word becomes available in the receive buffer, a receive event occurs. It is indicated by flag RXENDIF (USCI_PROTSTS [4]) and, if enabled, leads to receive interrupt.

- Data lost event to indicate a loss of the newest received data word:

  If the data word available in register USCI_RXDAT (oldest data word from RX_BUF0 or RX_BUF1) has not been read out and the receive buffer is FULL, the new incoming data will lose and this event occurs. It is indicated by flag RXOVIF (USCI_BUFSTS[3]) and, if enabled, leads to a protocol interrupt.

The general event and interrupt structure is shown in Figure 6.19-12.

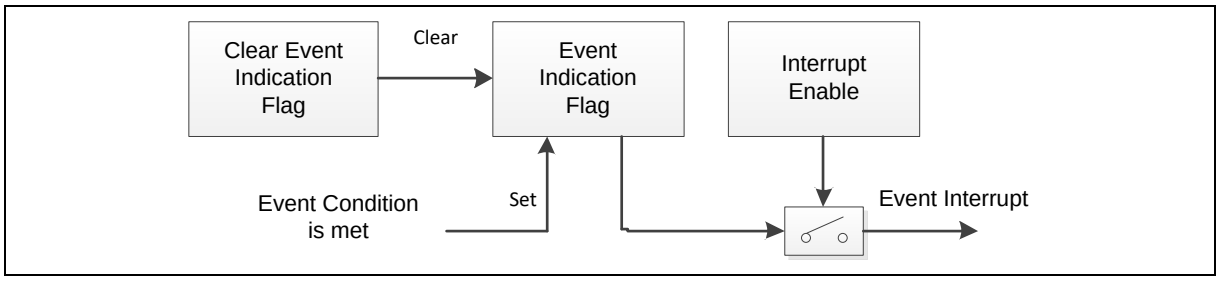


Figure 6.19-12 Event and Interrupt Structure

Each general interrupt enable can set by RXENDIEN, RXSTIEN, TXENDIEN, and TXSTIEN of USCI_INTEN [4:1]. The events are including receive end interrupt event, receive start interrupt event, transmit end interrupt event, and transmit start interrupt event. For protocol-specific interrupt, it is specified in each protocol interrupt enable register.

If a defined condition is met, an event is detected and an event indication flag becomes automatically set. The flag stays set until it is cleared by software. If enabled, an interrupt can be generated if an event is detected.

The registers, bits and bit fields indicate the data transfer events and control the general interrupts of a USCI are shown in Table 6.19-3.

Event	Indication Flag	Indication Cleared By	Interrupt Enabled By
Transmit start interrupt event	TXSTIF (USCI_PROTSTS [1])	It is cleared by software writes 1 to corresponding interrupt bit of USCI_PROTSTS.	TXSTIEN (USCI_INTEN [1])
Transmit end interrupt event	TXENDIF (USCI_PROTSTS [2])		TXENDIEN (USCI_INTEN [2])
Receive start interrupt event	RXSTIF (USCI_PROTSTS [3])		RXSTIEN (USCI_INTEN [3])
Receive end interrupt event	RXENDIF (USCI_PROTSTS [4])		RXENDIEN (USCI_INTEN [4])

Table 6.19-3 Data Transfer Events and Interrupt Handling

6.19.4.7 Protocol-specific Events and Interrupts

These events are related to protocol-specific actions that are described in the corresponding protocol chapters. The related indication flags are located in register USCI_PROTSTS. All events can be individually enabled for the generation of the common protocol interrupt.

Event	Indication Flag	Indication Cleared By	Interrupt Enabled By
Protocol-specific events in UART mode	USCI_PROTSTS [17:16] and USCI_PROTSTS [11:5]	It is cleared by software writes 1 to corresponding interrupt bit of USCI_PROTSTS.	USCI_PROTIEN[2:1]
Protocol-specific events in SPI mode	USCI_PROTSTS [9:8], USCI_PROTSTS [6:5]		USCI_PROTIEN [3:0]
Protocol-specific events in I^2C mode	USCI_PROTSTS [13:8], USCI_PROTSTS [5]		USCI_PROTIEN [6:0]

Table 6.19-4 Protocol-specific Events and Interrupt Handling

6.19.4.8 Wake-up

The protocol-related wake-up functional information is located in the Wake-up Control Register (USCI_WKCTL) and in the Wake-up Status Register (USCI_WKSTS). These registers are shared between the available protocols. As a consequence, the meaning of the bit positions in these registers is different within the protocols.

6.19.4.9 PDMA

The USCI supports PDMA transfer function. When PDMAEN (USCI_PDMACTL [3]) is set to 1, the PDMA function is enabled.

When TXPDMAEN (USCI_PDMACTL [1]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

When RXPDMAEN (USCI_PDMACTL [2]) is set to 1, the controller will start the PDMA reception process. USCI will issue request to PDMA controller automatically when there is data in the receive FIFO buffer.

In UART function, the requirement of RXPDMAEN will be cleared and hold if there is any error condition events including frame error, parity error or break detection. The user shall read out the current data and then the requirement of RXPDMAEN will send to the PDMA module in the next data.

6.20 USCI – UART Mode

6.20.1 Overview

The asynchronous serial channel UART covers the reception and the transmission of asynchronous data frames. It performs a serial-to-parallel conversion on data received from the peripheral, and a parallel-to-serial conversion on data transmitted from the controller. The receiver and transmitter are independent, and the transmission and reception can be started separately.

The UART controller also provides auto flow control. There are three conditions to wake up the system.

6.20.2 Features

- Supports one transmit buffer and two receive buffer for data payload
- Supports hardware auto flow control function
- Supports programmable baud-rate generator
- Support 9-bit Data Transfer (Support 9-bit RS-485)
- Baud rate detection possible by built-in capture event of baud rate generator
- Supports PDMA transfer
- Supports Wake-up function (Incoming Data and nCTS Wakeup Only)

6.20.3 Block Diagram

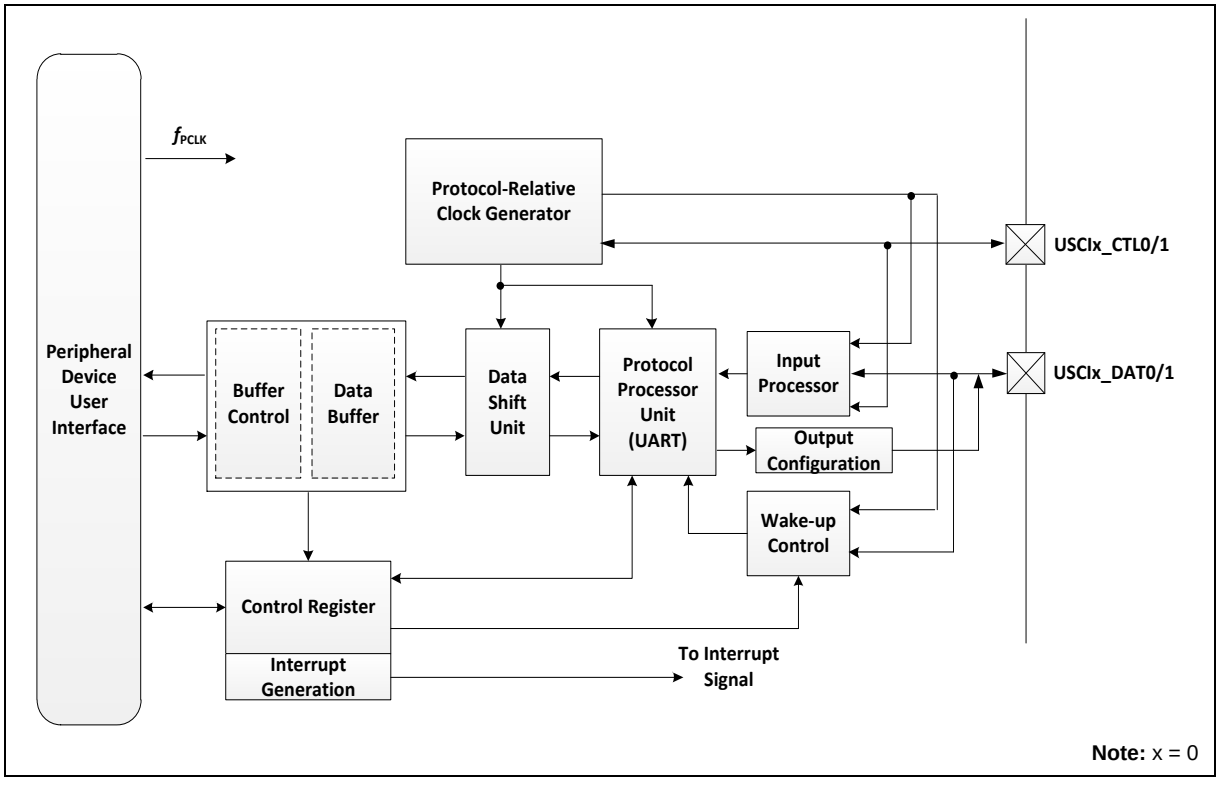


Figure 6.20-1 USCI-UART Mode Block Diagram

6.20.4 Basic Configuration

6.20.4.1 USCI0_UART Basic Configuration

- Clock Source Configuration
- Enable USCI0 clock (USCI0CKEN) on CLK_APBCLK1[8] register.
- Reset USCI0 (USCI0RST) on SYS_IPRST2[8] register.
- Enable USCI0_UART function USCI0_CTL[2:0] register, USCI0_CTL[2:0]=0x2.

6.20.5 Functional Description

6.20.5.1 UUART Common Functional Description

Please refer to section USCI for detailed information.

6.20.5.2 Signal Description

An UART connection is characterized by the use of a single connection line between a transmitter and a receiver. The receiver input signal (RXD) is handled by the input stage USCIx_DAT0 and the transmit output (TXD) signal is handled by the output stage of USCIx_DAT1.

For full-duplex communication, an independent communication line is needed for each transfer direction. Figure 6.20-2 shows an example with a point-to-point full-duplex connection between two communication partners UART module A and UART module B.

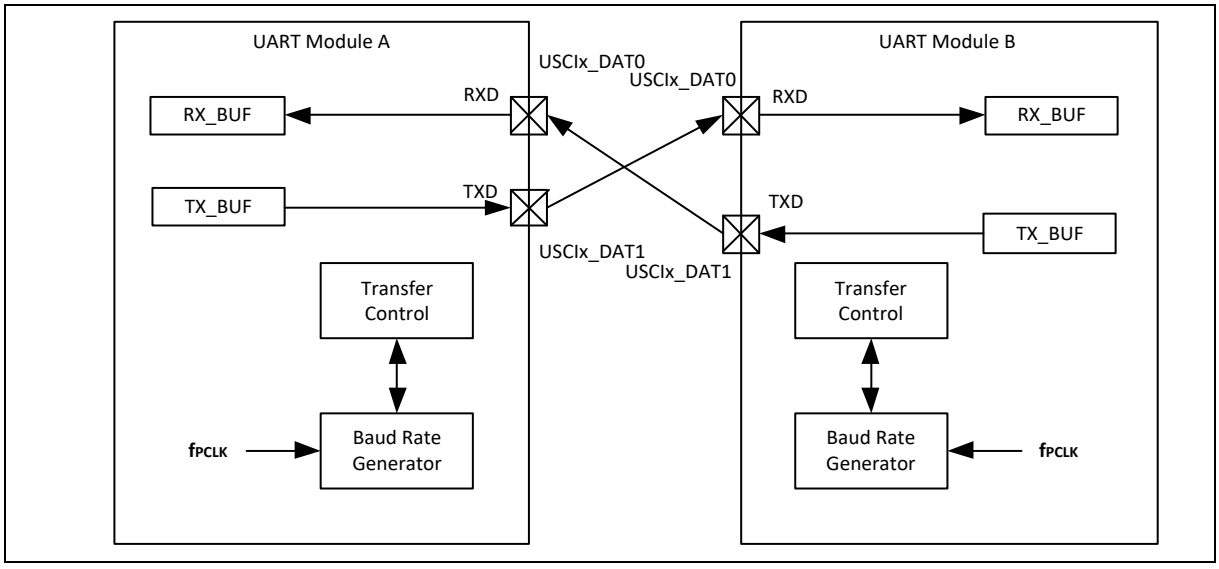


Figure 6.20-2 UART Signal Connection for Full-Duplex Communication

Input Signal

For UART protocol, the number of input signals is shown in Table 6.20-1. Each input signal is handled by an input processor for signal conditioning, such as signal inverse selection control, or a digital input filter. They can be classified according to their meaning for the protocols (see Table 6.20-1).

Selected Protocol		UART
Control Input	USCIx_CTL0	nCTS
	USCIx_CTL1	-
Data Input	USCIx_DAT0	RX
	USCIx_DAT1	-

Table 6.20-1 Input Signals for UART Protocol

Output Signals

For UART protocol, up to each protocol-related output signals are available. The number of actually used outputs depends on the selected protocol. They can be classified according to their meaning for the protocols.

Selected Protocol		UART
Control Output	USCIx_CTL0	-
	USCIx_CTL1	nRTS
Data Output	USCIx_DAT0	-
	USCIx_DAT1	TX

Table 6.20-2 Output Signals for UART Protocol

6.20.5.3 Frame Format

A standard UART frame is shown in Figure 6.20-3. It consists of:

- An idle time with the signal level 1.
- One start of frame bit (SOF) with the signal level 0.
- 6~13 bit data
- A PARITY bit (P), programmable for either even or odd parity. It is optionally possible to handle frames without PARITY bit.
- One or two STOP bits with the signal level 1.

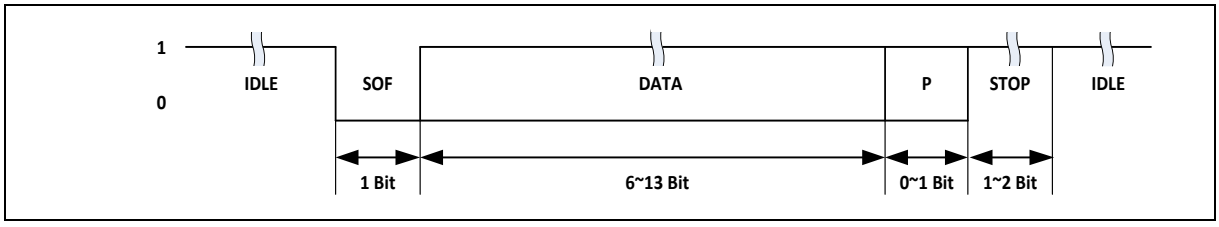


Figure 6.20-3 UART Standard Frame Format

The protocol specific bits (SOF, P, STOP) are automatically handled by the UART protocol state machine and do not appear in the data flow via the receive and transmit buffers.

START bit

The receiver input signal USCIx_DAT0 is checked for a falling edge. An SOF bit is detected when a falling edge occurs while the receiver is idle or after the sampling point of the last STOP bit. To increase noise immunity, the SOF bit timing starts with the first falling edge that is detected. If the sampled bit value of the SOF is 1, the previous falling edge is considered to be due to noise and the receiver is considered to be idle again.

Data Field

The length of the data field (number of data bits) can be programmed by the bit field of DWIDTH (UUART_LINECTL[11:8]). It can vary between 6 to 13 data bits.

Note: In UART protocol, the data transmission order is LSB first by setting LSB (UUART_LINECTL[0]) to 1.

PARITY bit

The UART allows parity generation for transmission and parity check for reception on frame base. The type of parity can be selected by bit field PARITYEN (UUART_PROTCTL[1]) and EVENPARITY (UUART_PROTCTL[2]), common for transmission and reception (no parity, even or odd parity). If the parity handling is disabled, the UART frame does not contain any PARITY bit. For consistency reasons, all communication partners have to be programmed to the same parity mode.

After the last data bit of the data field, the transmitter automatically sends out its calculated PARITY bit if parity generation has been enabled. The receiver interprets this bit as received parity and compares it to its internally calculated one. The result of the parity check and frame check (STOP bit) are monitored in the protocol status registers (UUART_PROTSTS). The register contains bits to monitor a protocol-related status and protocol-related error indication (FRMERR, PARITYERR).

STOP bit

Each UART frame is completed by 1 or 2 of STOP bits with the signal level 1 (same level as the idle level). The number of STOP bits is programmable by bit STOPB (UUART_PROTCTL[0]). A new START bit can be transferred directly after the last STOP bit.

Transfer Status Indication

RXBUSY (UUART_PROTSTS[10]) indicates the receiver status.

The receiver status can be monitored by RXBUSY bit. In this case, bit RXBUSY is set during a complete frame reception from the beginning of the start of frame bit to the end of the last STOP bit.

6.20.5.4 Operating Mode

To operate the UART protocol, the following issues have to be considered:

Select UART Mode

The UART protocol can be selected by setting FUNMODOE (UUART_CTL[2:0]) to 0x2 and the UART protocol can be enabled by setting PROTEN (UUART_PROTCTL [31]) to 1. Note that the FUNMODE must be set 0 before protocol changing and it is recommended to configure all parameters of the UART before UART protocol is enabled.

Pin Connections

The USCIx_DAT0 pin is used for UART receive data input signal (RX) in UART protocol. The property of input data signal can be configured in UUART_DATIN0. It is suggested to set EDGEDET (UUART_DATIN0[4:3]) as 10B for START bit detection.

The USCIx_DAT1 pin is used for UART transmit data output signal (TX) in UART protocol. The property of output data signal can be configured in UUART_LINECTL.

The USCIx_CTL0 pin is used for UART clear to send signal (nCTS) in UART protocol. The property of input control signal can be configured in UUART_CTLIN0.

The USCIx_CTL1 pin is used for UART request to send signal (nRTS) in UART protocol. The property of output control signal can be configured in UUART_LINECTL.

Bit Timing Configuration

The desired baud rate setting has to be selected, comprising the baud rate generator and the bit timing.

Frame Format Configuration

The word length, the STOP bit number, and the parity mode has to be set up according to the application requirements by programming UUART_LINECTL and the UUART_PROTCTL register. If required by the application, the data input and output signals can be inverted. The data transmission order is LSB first by setting LSB (UUART_LINECTL[0]) to 1.

6.20.5.5 Bit Timing

In UART mode, each frame bit is divided into data sample time in order to provide granularity in the sub-

bit range to adjust the sample point to the application requirements. The number of data sample time per bit is defined by bit fields DSCNT (UUART_BRGEN[14:10]) and the length of a data sample time is given by PDSCNT (UUART_BRGEN[9:8]).

In the example given in Figure 6.20-4, one bit time is composed of 16 data sample time DSCNT(UUART_BRGEN[14:10]) = 15. It is not recommended to program less and equal than 4 data sample time per bit time.

The position of the sampling point for the bit value is fixed in 1/2 samples time. It is possible to sample the bit value to take the average of samples.

The bit timing setup (number of data sample time) is common for the transmitter and the receiver because they use the same hardware circuit.

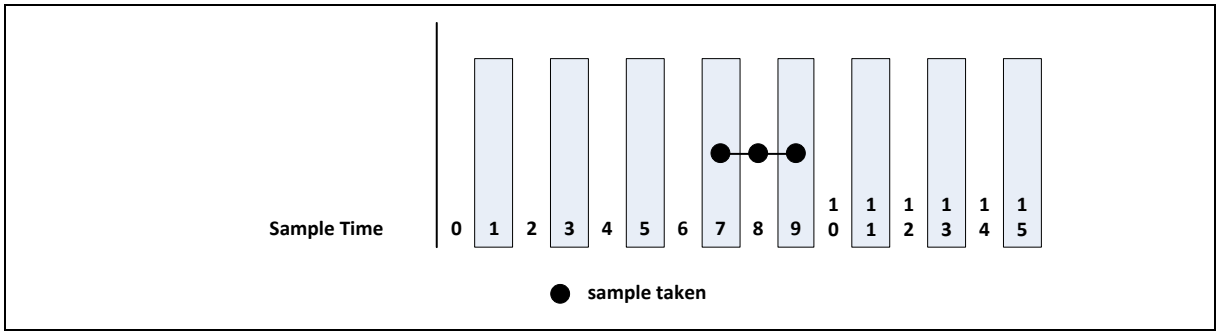


Figure 6.20-4 UART Bit Timing (Data Sample Time)

6.20.5.6 Baud Rate Generation

The baud rate f_{UART} in UART mode depends on the number of data sample time per bit time and their timing. The baud rate setting should only be changed while the transmitter and the receiver are idle. The bits RCLKSEL, SPCLKSEL, PDSCNT, and DSCNT define the baud rate setting:

RCLKSEL (UUART_BRGEN [0])

to define the input frequency f_{REF_CLK}

SPCLKSEL (UUART_BRGEN[3:2])

to define the multiple source of the sample clock f_{SAMP_CLK}

PDSCNT (UUART_BRGEN [9:8])

to define the length of a data sample time (division of f_{REF_CLK} by 1, 2, 3, or 4)

DSCNT (UUART_BRGEN [14:10])

to define the number of data sample time per bit time

The standard setting is given by RCLKSEL = 0 ($f_{REF_CLK} = f_{PCLK}$), PTCLKSEL = 0 ($f_{PROT_CLK} = f_{REF_CLK}$) and SPCLKSEL = 0 ($f_{SAMP_CLK} = f_{DIV_CLK}$). Under these conditions, the baud rate is given by:

$$f_{\mathrm{UART}} = f_{REF_CLK} \times \frac{1}{\mathrm{CLKDIV}+1} \times \frac{1}{\mathrm{PDSCNT}+1} \times \frac{1}{\mathrm{DSCNT}+1}$$

In order to generate slower frequencies, additional divide-by-2 stages can be selected by PTCLKSEL = 1 ($f_{PROT_CLK} = f_{REF_CLK2}$), leading to:

$$f_{\mathrm{UART}} = \frac{f_{REF_CLK}}{2} \times \frac{1}{\mathrm{CLKDIV}+1} \times \frac{1}{\mathrm{PDSCNT}+1} \times \frac{1}{\mathrm{DSCNT}+1}$$

If SPCLKSEL = 2 ($f_{SAMP_CLK} = f_{SCLK}$), and RCLKSEL = 0 ($f_{REF_CLK} = f_{PCLK}$), PTCLKSEL = 0 ($f_{PROT_CLK} = f_{REF_CLK}$). The baud rate is given by:

$$f_{\mathrm{UART}} = f_{REF_CLK} \times \frac{1}{\mathrm{CLKDIV}+1} \times \frac{1}{2} \times \frac{1}{\mathrm{PDSCNT}+1} \times \frac{1}{\mathrm{DSCNT}+1}$$

There is error tolerance for the UART baud rate after setting the baud rate parameter. Table 6.20-3 lists the relative error percentage examples for user to calculate his relative baud rate setting. The clock source is standard setting (SPCLKSEL=0, PTCLKSEL=0 and RCLKSEL=0).

HCLK Source	PCLK Source	Expect Baud Rate	CLKDIV (UUART_BRGEN[25:16])	DSCNT (UUART_BRGEN[14:10])	PDSCNT	Active Baud Rate	Error Percentage
12 MHz	HCLK	115200	0xC	0x7	0x0	115384	0.16%
12 MHz	HCLK	9600	0x7C	0x9	0x0	9600	0%
12 MHz	HCLK	2400	0x1F3	0x9	0x0	2400	0%

Table 6.20-3 Baud Rate Relationship

6.20.5.7 Auto Baud Rate Detection

The UART controller supports auto baud rate detection function. It is used to identify the input baud rate from the receiver signal (USCIx_DAT0) and then revised the baud rate clock divider CLKDIV (UUART_BRGEN[25:16]) after the baud rate function done to meet the detected baud rate information. According the section of Timing Measurement Counter, the timing measurement counter is used for time interval measurement of the input signal (USCIx_DAT0) and the actual timer value is captured into bit field BRDETITV (UUART_PROTCTL [24:16]) in each falling edge of the detected signal.

When the ABREN (UUART_PROTCTL[6]) bit is enabled, the 0x55 data patterns is necessary for auto baud rate detection. The falling edge of input signal starts the baud rate counter and it loads the timing measurement counter value into the BRDETITV (UUART_PROTCTL [24:16]) in the next falling edge. It is suggested to use the f_{DIV_CLK} (TMCNTSRC (UUART_BRGENC[5]) =1) as the counter source.

The CLKDIV (UUART_BRGEN[25:16]) will be revised by BRDETITV (UUART_PROTCTL [25:16]) after the auto baud rate function done (the time of 4th falling edge of input signal). If the user want to receive the next successive frame correctly, it is better to set the value of CLKDIV (UUART_BRGEN[25:16]) and DSCNT (UUART_BRGEN[14:10]) as the same value (the value shall be among the rang of 0xF and 0x5 because the DSCNT is used to define the sample counter of each bit and the PDSCNT (UUART_BRGEN[9:8]) is 0x0.

During the auto baud rate detection, the ABRDETIF (UUART_PROTSTS[9]) and the BRDETITV (UUART_PROTCTL [24:16]) will be updated after each falling edge of input signal and the auto baud rate pattern, 0x55, won't be received into the receiver buffer after the frame done. The bit of ABREN will be cleared by hardware after the 4th falling edge of input signal is detected thus the user can read the status of ABREN to know the auto baud rate function is done or not.

If the CLKDIV and DSCNT are not set as the same value in calculation the auto baud rate function, the user shall calculate the proper average baud rate by the value of BRDETITV and CLKDIV after the auto baud rate function done.

If the baud rate of input signal is very slower and the bit time of timing measurement counter can't calculate the correct period of the input bit time, there is a ABERRSTS bit (UUART_PROTSTS[11]) to indicate the error information of the auto baud rate detection. At this time, the user shall revise the value of CLKDIV and require the Host device to send the 0x55 pattern again.

According the limitation of timing measurement counter, the maximum auto baud rate detection is 0x1FE for BRDETITV. The UART Auto Baud Rate Control is shown in Figure 6.20-5.

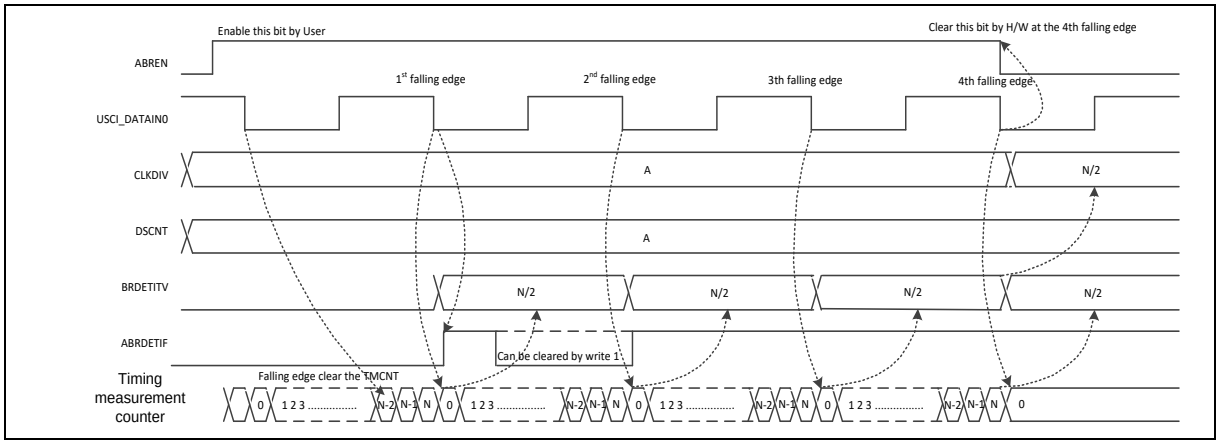


Figure 6.20-5 UART Auto Baud Rate Control

6.20.5.8 Auto Flow Control

The UART supports hardware auto-flow control that provides nRTS flow control by indicator RXFULL (UUART_BUFSTS[1]) on receiver buffer. When the buffer is full (RXFULL = 1), the nRTS is de-asserted.

The UART also provides nCTS flow control on transmitter. The nCTS is used to control the transmitted data is sent out when the nCTS is asserted.

6.20.5.9 RS-485 Support

The UART controller can play the role of the RS-485 master transmitter will identify an address character by setting the parity (9-th bit) to 1. For data characters, the parity is set to 0. Software can use the bit15 of each data to control the PARITY bit (PARITYEN (UUART_PROTCTL[1]) be set) when the STICKEN (UUART_PROTCTL[26]) is set. For example, if the STICKEN is set to 1 and data sequence are 0x8015, 0x8033, 0x0055, 0x0033 and 0x80AA the transmitted parity of data 0x15, 0x33, 0x55 0x33 and 0xAA will be 1, 1, 0, 0 and 1.

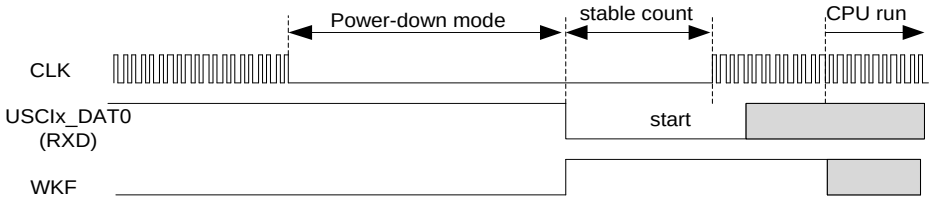
The UART controller can also play as an RS-485 addressable slave, the protocol-related error of PARITYERR (UUART_PROTSTS[5]) can be acted as the address bit detection when the PARITYEN (UUART_PROTCTL[1]), EVENPARITY (UUART_PROTCTL[2]) and STICKEN (UUART_PROTCTL[26]) were set. If the PARITYERR was set, it means that the address bit in the received bus is detected otherwise, the data is received into Buffer.

6.20.5.10 Wake-up Function

The USCI Controller in UART mode supports wake-up system function. The wake-up source includes incoming data and nCTS pin. Each wake-up source description is as follows:

(a) Incoming data wake-up

When system is in power-down and both of the WKEN (UUART_WKCTL [0]) and DATWKEN (UUART_PROTCTL[9]) are set, the toggle of incoming data pin can wake-up the system. In order to receive the incoming data after the system wake-up, the WAKECNT (UUART_PROTCTL[14:11]) shall be set. These bits field of WAKECNT (UUART_PROCTL[14:11]) indicate how many clock cycle selected by f_{PDS_CNT} do the controller can get the 1st bit (START bit) when the device is wakeup from Power-down mode. The incoming data wake-up is shown in Figure 6.20-6.



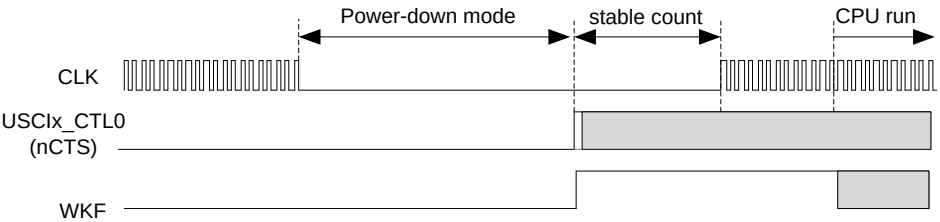
Note: Stable count means HCLK source recovery stable count.

Figure 6.20-6 Incoming Data Wake-Up

(b) nCTS pin wake-up

When system is in power-down and both of the WKEN (UUART_WKCTL [0]) and CTSWKEN (UUART_PROTCTL[10]) are set, the toggle of nCTS pin can wake-up the system. The nCTS wake-up is shown in Figure 6.20-7 and Figure 6.20-8.

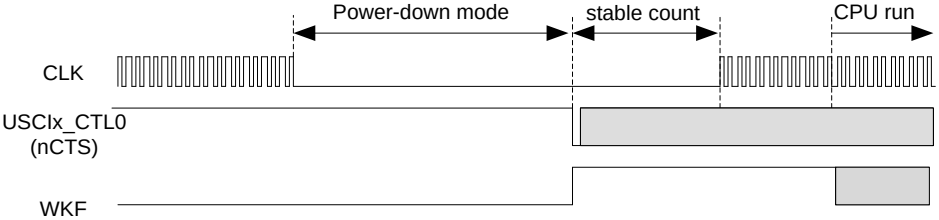
Case 1(nCTS transition from low to high):



Note: Stable count means HCLK source recovery stable count.

Figure 6.20-7 nCTS Wake-Up Case 1

Case 2 (nCTS transition from high to low):



Note 1: Stable count means HCLK source recovery stable count.

Figure 6.20-8 nCTS Wake-Up Case 2

6.20.5.11 Interrupt Events

The UART provided interrupt for protocol event and data transfer event. The description shows below:

Protocol Interrupt Events

The following protocol-related events are generated in UART mode and can lead to a protocol interrupt.

Please note that the bits in register UUART_PROTSTS are not automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

Receiver Line Status

The protocol-related error FRMERR (UUART_PROTSTS[6]) or PARITYERR (UUART_PROTSTS[5]) are two flags that are assigned to each received data word in the corresponding receiver buffer status registers.

In UART mode, the result of the parity check by the protocol-related error indication (0 = received PARITY bit equal to calculated parity value), and the result of frame check by the protocol-related error indication (0 = received STOP bit equal to the format value '1'). This information is elaborated for each data frame.

The break error flag BREAK (UUART_PROTSTS[7]) is assigned when the receive data is 0, the received parity and the STOP bit are also 0.

The interrupt indicates that there are parity error, frame error or the break data detection in the BREAK, FRMERR, PARITYERR (UUART_PROTSTS[7:5]) bits.

Auto Baud Rate Detection

The auto baud rate interrupt, ABRDETIF (UUART_PROTSTS [9]), indicates that the timing measurement counter has getting 2-bit duration for auto baud rate capture function.

The auto baud rate detection function will be enabled in the first falling edge of receiver signal. The auto baud rate detection function is measurement after the next following falling is detected and it is finished when the frame transfer done. After the transfer done, the timing measurement counter value divided by twice is equal to the number of sample time per bit. The user can read the value of BRDETITV (UUART_PROTCTL[24:16]) and write into the baud rate generator register CLKDIV (UUART_BRGEN[25:16]).

Data Transfer Interrupt Handling

The data transfer interrupts indicate events related to UART frame handling.

Transmit Start Interrupt

Bit TXSTIF (UUART_PROTSTS [1]) is set after the START bit of a data word. In buffer mode, this is the earliest point in time when a new data word can be written to UUART_TXDAT.

Transmitter Finished

This interrupt indicates that the transmitter has completely finished all data in the buffer. Bit TXENDIF (UUART_PROTSTS [2]) becomes set at the end of the last STOP bit.

Receiver Starts Interrupt

Bit RXSTIF (UUART_PROTSTS [3]) is set after the sample point of the START bit.

Receiver Frame Finished

This interrupt indicates that the receiver has completely finished a frame. Bit RXENDIF (UUART_PROTSTS [4]) becomes set at the end of the last receive bit.

6.20.5.12 Programming Example

The following steps are used to configure the UART protocol setting and the data transmission.

1. Set FUNMODE (UUART_CTL[2:0]) to 0x2 to select UART protocol.
2. Write baud rate generator register UUART_BRGEN to select desired baud rate.
   - Set SPCLKSEL (UUART_BRGEN[3:2]), PTCLKSEL (UUART_BRGEN[1]) and RCLKSEL (UUART_BRGEN[0]) to select the clock source.
   - Configure CLKDIV (UUART_BRGEN[25:16]), DSCNT (UUART_BRGEN[14:10]) and PDSCNT (UUART_BRGEN[9:8]) to determine the baud rate divider.
3. Write line control register UUART_LINECTL and protocol control register UUART_PROTCTL

to configure the transmission data format and UART protocol setting.

- Program data field length in DWIDTH (UUART_LINECTL[11:8]).
- Enable PARITY bit and determine the PARITY bit type by setting EVENPARITY (UUART_PROTCTL[2]) and PARITYEN (UUART_PROTCTL[1]).
- Configure STOP bit length by setting STOPB (UUART_PROTCTL[0]).
- Enable LSB (UUART_LINECTL[0]) to select LSB first transmission for UART protocol.
- Set EDGEDET (UUART_DATIN0[4:3]) to “10” to select the detected edge as falling edge for receiver START bit detection.

4. Set PROTEN (UUART_PROTCTL[31]) to 1 to enable UART protocol.
5. Transmit and receive data.
   - Write transmit data register UUART_TXDAT to transmit data.
   - Wait until TXSTIF(UUART_PROTSTS[1]) is set and then user can write the next data in UUART_TXDAT.
   - When TXENDIF(UUART_PROTSTS[2]) is set, the transmit buffer is empty and the STOP bit of the last data has been transmitted.
   - If RXENDIF(UUART_PROTSTS[4]) is set, the receiver has finished a data frame completely. User can get the data by reading receive data register UUART_RXDAT.

6.20.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
USCI_UART Base Address: **UUARTn_BA = 0x400D_0000 + (0x1000 * n)** **n= 0**				
UUART_CTL	UUARTn_BA+0x00	R/W	USCI Control Register	0x0000_0000
UUART_INTEN	UUARTn_BA+0x04	R/W	USCI Interrupt Enable Register	0x0000_0000
UUART_BRGEN	UUARTn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00
UUART_DATIN0	UUARTn_BA+0x10	R/W	USCI Input Data Signal Configuration Register 0	0x0000_0000
UUART_CTLIN0	UUARTn_BA+0x20	R/W	USCI Input Control Signal Configuration Register 0	0x0000_0000
UUART_CLKIN	UUARTn_BA+0x28	R/W	USCI Input Clock Signal Configuration Register	0x0000_0000
UUART_LINECTL	UUARTn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000
UUART_TXDAT	UUARTn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000
UUART_RXDAT	UUARTn_BA+0x34	R	USCI Receive Data Register	0x0000_0000
UUART_BUFCTL	UUARTn_BA+0x38	R/W	USCI Transmit/Receive Buffer Control Register	0x0000_0000
UUART_BUFSTS	UUARTn_BA+0x3C	R	USCI Transmit/Receive Buffer Status Register	0x0000_0101
UUART_PDMACTL	UUARTn_BA+0x40	R/W	USCI PDMA Control Register	0x0000_0000
UUART_WKCTL	UUARTn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000
UUART_WKSTS	UUARTn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000
UUART_PROTCTL	UUARTn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0000
UUART_PROTIEN	UUARTn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000
UUART_PROTSTS	UUARTn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

6.20.7 Register Description

USCI Control Register (UUART_CTL)

Register	Offset	R/W	Description	Reset Value
UUART_CTL	UUARTn_BA+0x00	R/W	USCI Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**FUNMODE**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**FUNMODE**	**Function Mode** This bit field selects the protocol for this USCI controller. Selecting a protocol that is not available or a reserved combination disables the USCI. When switching between two protocols, the USCI has to be disabled before selecting a new protocol. Simultaneously, the USCI will be reset when user write 000 to FUNMODE. 000 = The USCI is disabled. All protocol related state machines are set to idle state. 001 = The SPI protocol is selected. 010 = The UART protocol is selected. 100 = The I^2C protocol is selected. Others = Reserved.

USCI Interrupt Enable Register (UUART_INTEN)

Register	Offset	R/W	Description	Reset Value
UUART_INTEN	UUARTn_BA+0x04	R/W	USCI Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**RXENDIEN**	**RXSTIEN**	**TXENDIEN**	**TXSTIEN**	**Reserved**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**RXENDIEN**	**Receive End Interrupt Enable Bit** This bit enables the interrupt generation in case of a receive finish event. 0 = The receive end interrupt Disabled. 1 = The receive end interrupt Enabled.
[3]	**RXSTIEN**	**Receive Start Interrupt Enable Bit** This bit enables the interrupt generation in case of a receive start event. 0 = The receive start interrupt Disabled. 1 = The receive start interrupt Enabled.
[2]	**TXENDIEN**	**Transmit End Interrupt Enable Bit** This bit enables the interrupt generation in case of a transmit finish event. 0 = The transmit finish interrupt Disabled. 1 = The transmit finish interrupt Enabled.
[1]	**TXSTIEN**	**Transmit Start Interrupt Enable Bit** This bit enables the interrupt generation in case of a transmit start event. 0 = The transmit start interrupt Disabled. 1 = The transmit start interrupt Enabled.
[0]	**Reserved**	Reserved.

USCI Baud Rate Generator Register (UUART_BRGEN)

Register	Offset	R/W	Description	Reset Value
UUART_BRGEN	UUARTn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00

31	30	29	28	27	26	25	24
Reserved						CLKDIV	
23	22	21	20	19	18	17	16
CLKDIV							
15	14	13	12	11	10	9	8
Reserved	DSCNT					PDSCNT	
7	6	5	4	3	2	1	0
Reserved		TMCNTSRC	TMCNTEN	SPCLKSEL		PTCLKSEL	RCLKSEL

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25:16]	**CLKDIV**	**Clock Divider** This bit field defines the ratio between the protocol clock frequency f_{PROT_CLK} and the clock divider frequency f_{DIV_CLK} ($f_{DIV_CLK} = f_{PROT_CLK} / (CLKDIV+1)$). **Note:** In UART function, it can be updated by hardware in the 4th falling edge of the input data 0x55 when the auto baud rate function (ABREN(UUART_PROTCTL[6])) is enabled. The revised value is the average bit time between bit 5 and bit 6. The user can use revised CLKDIV and new BRDETITV (UUART_PROTCTL[24:16]) to calculate the precise baud rate.
[15]	**Reserved**	Reserved.
[14:10]	**DSCNT**	**Denominator for Sample Counter** This bit field defines the divide ratio of the sample clock f_{SAMP_CLK}. The divided frequency $f_{DS_CNT} = f_{PDS_CNT} / (DSCNT+1)$. **Note:** The maximum value of DSCNT is 0xF on UART mode and suggest to set over 4 to confirm the receiver data is sampled in right value.
[9:8]	**PDSCNT**	**Pre-divider for Sample Counter** This bit field defines the divide ratio of the clock division from sample clock f_{SAMP_CLK}. The divided frequency $f_{PDS_CNT} = f_{SAMP_CLK} / (PDSCNT+1)$.
[7:6]	**Reserved**	Reserved.
[5]	**TMCNTSRC**	**Timing Measurement Counter Clock Source Selection** 0 = Timing measurement counter with f_{PROT_CLK}. 1 = Timing measurement counter with f_{DIV_CLK}.
[4]	**TMCNTEN**	**Timing Measurement Counter Enable Bit** This bit enables the 10-bit timing measurement counter. 0 = Timing measurement counter is Disabled. 1 = Timing measurement counter is Enabled.
[3:2]	**SPCLKSEL**	**Sample Clock Source Selection** This bit field used for the clock source selection of a sample clock (f_{SAMP_CLK}) for the

		protocol processor. 00 = f_{SAMP_CLK} is selected to f_{DIV_CLK}. 01 = f_{SAMP_CLK} is selected to f_{PROT_CLK}. 10 = f_{SAMP_CLK} is selected to f_{SCLK}. 11 = f_{SAMP_CLK} is selected to f_{REF_CLK}.
[1]	**PTCLKSEL**	**Protocol Clock Source Selection** This bit selects the source signal of protocol clock (f_{PROT_CLK}). 0 = Reference clock f_{REF_CLK}. 1 = f_{REF_CLK2} (its frequency is half of f_{REF_CLK}).
[0]	**RCLKSEL**	**Reference Clock Source Selection** This bit selects the source signal of reference clock (f_{REF_CLK}). 0 = Peripheral device clock f_{PCLK}. 1 = Reserved.

USCI Input Data Signal Configuration (UUART_DATIN0)

Register	Offset	R/W	Description	Reset Value
UUART_DATIN0	UUARTn_BA+0x10	R/W	USCI Input Data Signal Configuration Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**EDGEDET**		**ININV**	**Reserved**	**SYNCSEL**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:3]	**EDGEDET**	**Input Signal Edge Detection Mode** This bit field selects which edge actives the trigger event of input data signal. 00 = The trigger event activation is disabled. 01 = A rising edge activates the trigger event of input data signal. 10 = A falling edge activates the trigger event of input data signal. 11 = Both edges activate the trigger event of input data signal. **Note:** In UART function mode, it is suggested to set this bit field as 0x2.
[2]	**ININV**	**Input Signal Inverse Selection** This bit defines the inverter enable of the input asynchronous signal. 0 = The un-synchronized input signal will not be inverted. 1 = The un-synchronized input signal will be inverted.
[1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Signal Synchronization Selection** This bit selects if the un-synchronized input signal (with optionally inverted) or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit.

USCI Input Control Signal Configuration (UUART_CTLIN0)

Register	Offset	R/W	Description	Reset Value
UUART_CTLIN0	UUARTn_BA+0x20	R/W	USCI Input Control Signal Configuration Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					ININV	Reserved	SYNCSEL

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**ININV**	**Input Signal Inverse Selection** This bit defines the inverter enable of the input asynchronous signal. 0 = The un-synchronized input signal will not be inverted. 1 = The un-synchronized input signal will be inverted.
[1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Synchronization Signal Selection** This bit selects if the un-synchronized input signal (with optionally inverted) or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit.

USCI Input Clock Signal Configuration (UUART_CLKIN)

Register	Offset	R/W	Description	Reset Value
UUART_CLKIN	UUARTn_BA+0x28	R/W	USCI Input Clock Signal Configuration Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**SYNCSEL**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Synchronization Signal Selection** This bit selects if the un-synchronized input signal or the synchronized (and optionally filtered) signal can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit.

USCI Line Control Register (UUART_LINECTL)

Register	Offset	R/W	Description	Reset Value
UUART_LINECTL	UUARTn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**DWIDTH**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CTLOINV	**Reserved**	**DATOINV**	**Reserved**				**LSB**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**DWIDTH**	**Word Length of Transmission** This bit field defines the data word length (amount of bits) for reception and transmission. The data word is always right-aligned in the data buffer. USCI support word length from 4 to 16 bits. 0000 = The data word contains 16 bits located at bit positions [15:0]. 0001 = Reserved. 0010 = Reserved. 0011 = Reserved. 0100 = The data word contains 4 bits located at bit positions [3:0]. 0101 = The data word contains 5 bits located at bit positions [4:0]. 0110 = The data word contains 6 bits located at bit positions [5:0]. 0111 = The data word contains 7 bits located at bit positions [6:0]. 1000 = The data word contains 8 bits located at bit positions [7:0]. 1001 = The data word contains 9 bits located at bit positions [8:0]. 1010 = The data word contains 10 bits located at bit positions [9:0]. 1011 = The data word contains 11 bits located at bit positions [10:0]. 1100 = The data word contains 12 bits located at bit positions [11:0]. 1101 = The data word contains 13 bits located at bit positions [12:0]. 1110 = The data word contains 14 bits located at bit positions [13:0]. 1111 = The data word contains 15 bits located at bit positions [14:0]. **Note:** In UART protocol, the length can be configured as 6~13 bits.
[7]	**CTLOINV**	**Control Signal Output Inverse Selection** This bit defines the relation between the internal control signal and the output control signal. 0 = No effect. 1 = The control signal will be inverted before its output. **Note:** In UART protocol, the control signal means nRTS signal.
[6]	**Reserved**	Reserved.

[5]	**DATOINV**	**Data Output Inverse Selection** This bit defines the relation between the internal shift data value and the output data signal of USCIx_DAT1 pin. 0 = The value of USCIx_DAT1 is equal to the data shift register. 1 = The value of USCIx_DAT1 is the inversion of data shift register.
[4:1]	**Reserved**	Reserved.
[0]	**LSB**	**LSB First Transmission Selection** 0 = The MSB, which bit of transmit/receive data buffer depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, the bit 0 of data buffer, will be transmitted/received first.

USCI Transmit Data Register (UUART_TXDAT)

Register	Offset	R/W	Description	Reset Value
UUART_TXDAT	UUARTn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TXDAT**	**Transmit Data** Software can use this bit field to write 16-bit transmit data for transmission.

USCI Receive Data Register (UUART_RXDAT)

Register	Offset	R/W	Description	Reset Value
UUART_RXDAT	UUARTn_BA+0x34	R	USCI Receive Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RXDAT**	**Received Data** This bit field monitors the received data which stored in receive data buffer. **Note:** RXDAT[15:13] indicate the same frame status of BREAK, FRMERR and PARITYERR (UUART_PROTSTS[7:5]).

USCI Transmitter/Receive Buffer Control Register (UUART_BUFCTL)

Register	Offset	R/W	Description	Reset Value
UUART_BUFCTL	UUARTn_BA+0x38	R/W	USCI Transmit/Receive Buffer Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						RXRST	TXRST
15	14	13	12	11	10	9	8
RXCLR	RXOVIEN	Reserved					
7	6	5	4	3	2	1	0
TXCLR	Reserved						

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset the receive-related counters, state machine, and the content of receive shift register and data buffer. **Note 1:** It is cleared automatically after one PCLK cycle. **Note 2:** It is suggested to check the RXBUSY (UUART_PROTSTS[10]) before this bit will be set to 1.
[16]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset the transmit-related counters, state machine, and the content of transmit shift register and data buffer**.** **Note:** It is cleared automatically after one PCLK cycle.
[15]	**RXCLR**	**Clear Receive Buffer** 0 = No effect. 1 = The receive buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the buffer is not taking part in data traffic. **Note:** It is cleared automatically after one PCLK cycle.
[14]	**RXOVIEN**	**Receive Buffer Overrun Error Interrupt Enable Bit** 0 = Receive overrun interrupt Disabled. 1 = Receive overrun interrupt Enabled.
[13:8]	**Reserved**	Reserved.
[7]	**TXCLR**	**Clear Transmit Buffer** 0 = No effect. 1 = The transmit buffer is cleared (filling level is cleared and output pointer is set to input pointer value). Should only be used while the buffer is not taking part in data traffic. **Note:** It is cleared automatically after one PCLK cycle.
[6:0]	**Reserved**	Reserved.

USCI Transmit/Receive Buffer Status Register (UUART_BUFSTS)

Register	Offset	R/W	Description	Reset Value
UUART_BUFSTS	UUARTn_BA+0x3C	R	USCI Transmit/Receive Buffer Status Register	0x0000_0101

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						TXFULL	TXEMPTY
7	6	5	4	3	2	1	0
Reserved				RXOVIF	Reserved	RXFULL	RXEMPTY

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**TXFULL**	**Transmit Buffer Full Indicator** 0 = Transmit buffer is not full. 1 = Transmit buffer is full.
[8]	**TXEMPTY**	**Transmit Buffer Empty Indicator** 0 = Transmit buffer is not empty. 1 = Transmit buffer is empty.
[7:4]	**Reserved**	Reserved.
[3]	**RXOVIF**	**Receive Buffer Over-run Error Interrupt Status** This bit indicates that a receive buffer overrun error event has been detected. If RXOVIEN (UUART_BUFCTL[14]) is enabled, the corresponding interrupt request is activated. It is cleared by software writes 1 to this bit. 0 = A receive buffer overrun error event has not been detected. 1 = A receive buffer overrun error event has been detected.
[2]	**Reserved**	Reserved.
[1]	**RXFULL**	**Receive Buffer Full Indicator** 0 = Receive buffer is not full. 1 = Receive buffer is full.
[0]	**RXEMPTY**	**Receive Buffer Empty Indicator** 0 = Receive buffer is not empty. 1 = Receive buffer is empty.

USCI PDMA Control Register (UUART_PDMACTL)

Register	Offset	R/W	Description	Reset Value
UUART_PDMACTL	UUARTn_BA+0x40	R/W	USCI PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PDMAEN**	**RXPDMAEN**	**TXPDMAEN**	**PDMARST**

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**PDMAEN**	**PDMA Mode Enable Bit** 0 = PDMA function Disabled. 1 = PDMA function Enabled.
[2]	**RXPDMAEN**	**PDMA Receive Channel Available** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[1]	**TXPDMAEN**	**PDMA Transmit Channel Available** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled.
[0]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the USCI's PDMA control logic. This bit will be cleared to 0 automatically.

USCI Wake-up Control Register (UUART_WKCTL)

Register	Offset	R/W	Description	Reset Value
UUART_WKCTL	UUARTn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**PDBOPT**	**Reserved**	**WKEN**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDBOPT**	**Power Down Blocking Option** 0 = If user attempts to enter Power-down mode by executing WFI while the protocol is in transferring, MCU will stop the transfer and enter Power-down mode immediately. 1 = If user attempts to enter Power-down mode by executing WFI while the protocol is in transferring, the on-going transfer will not be stopped and MCU will enter idle mode immediately.
[1]	**Reserved**	Reserved.
[0]	**WKEN**	**Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.

USCI Wake-up Status Register (UUART_WKSTS)

Register	Offset	R/W	Description	Reset Value
UUART_WKSTS	UUARTn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**WKF**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKF**	**Wake-up Flag** When chip is woken up from Power-down mode, this bit is set to 1. Software can write 1 to clear this bit.

USCI Protocol Control Register – UART (UUART_PROTCTL)

Register	Offset	R/W	Description	Reset Value
UUART_PROTCTL	UUARTn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PROTEN	DGE	BCEN	Reserved		STICKEN	Reserved	BRDETITV
23	22	21	20	19	18	17	16
BRDETITV							
15	14	13	12	11	10	9	8
Reserved	WAKECNT				CTSWKEN	DATWKEN	Reserved
7	6	5	4	3	2	1	0
Reserved	ABREN	RTSAUDIREN	CTSAUTOEN	RTSAUTOEN	EVENPARITY	PARITYEN	STOPB

Bits	Description	
[31]	**PROTEN**	**UART Protocol Enable Bit** 0 = UART Protocol Disabled. 1 = UART Protocol Enabled.
[30]	**DGE**	**Deglitch Enable Bit** 0 = Deglitch Disabled. 1 = Deglitch Enabled. **Note 1:** When this bit is set to logic 1, any pulse width less than about 150 ns will be considered a glitch and will be removed in the serial data input (RX). This bit acts only on RX line and has no effect on the transmitter logic. **Note 2:** It is recommended to set this bit only when operating at baud rate under 2.5 Mbps.
[29]	**BCEN**	**Transmit Break Control Enable Bit** 0 = Transmit Break Control Disabled. 1 = Transmit Break Control Enabled. **Note:** When this bit is set to logic 1, the serial data output (TX) is forced to the Spacing State (logic 0). This bit acts only on TX line and has no effect on the transmitter logic.
[28:27]	**Reserved**	Reserved.
[26]	**STICKEN**	**Stick Parity Enable Bit** 0 = Stick parity Disabled. 1 = Stick parity Enabled. **Note:** Refer to RS-485 Support section for detailed information.
[25]	**Reserved**	Reserved.
[24:16]	**BRDETITV**	**Baud Rate Detection Interval** This bit fields indicate how many clock cycle selected by TMCNTSRC (UUART_BRGEN [5]) does the slave calculates the baud rate in one bits. The order of the bus shall be 1 and 0 step by step (e.g. the input data pattern shall be 0x55). The user can read the value to know the current input baud rate of the bus whenever the ABRDETIF (UUART_PROTCTL[9]) is set. **Note:** This bit can be cleared to 0 by software writing '0' to the **BRDETITV**.

[15]	**Reserved**	Reserved.
[14:11]	**WAKECNT**	**Wake-up Counter** These bits field indicate how many clock cycle selected by f_{PDS_CNT} do the slave can get the 1st bit (START bit) when the device is wake-up from Power-down mode.
[10]	**CTSWKEN**	**nCTS Wake-up Mode Enable Bit** 0 = nCTS wake-up mode Disabled. 1 = nCTS wake-up mode Enabled.
[9]	**DATWKEN**	**Data Wake-up Mode Enable Bit** 0 = Data wake-up mode Disabled. 1 = Data wake-up mode Enabled.
[8:7]	**Reserved**	Reserved.
[6]	**ABREN**	**Auto-baud Rate Detect Enable Bit** 0 = Auto-baud rate detect function Disabled. 1 = Auto-baud rate detect function Enabled. **Note:** When the auto - baud rate detect operation finishes, hardware will clear this bit. The associated interrupt ABRDETIF (UUART_PROTSTS[9]) will be generated (If ABRIEN (UUART_PROTIEN [1]) is enabled).
[5]	**RTSAUDIREN**	**nRTS Auto Direction Enable Bit** When nRTS auto direction is enabled, if the transmitted bytes in the TX buffer is empty, the UART asserted nRTS signal automatically. 0 = nRTS auto direction control Disabled. 1 = nRTS auto direction control Enabled. **Note 1:** This bit is used for nRTS auto direction control for RS485. **Note 2:** This bit has effect only when the RTSAUTOEN is not set.
[4]	**CTSAUTOEN**	**nCTS Auto-flow Control Enable Bit** When nCTS auto-flow is enabled, the UART will send data to external device when nCTS input assert (UART will not send data to device if nCTS input is dis-asserted). 0 = nCTS auto-flow control Disabled. 1 = nCTS auto-flow control Enabled.
[3]	**RTSAUTOEN**	**nRTS Auto-flow Control Enable Bit** When nRTS auto-flow is enabled, if the receiver buffer is full (RXFULL (UUART_BUFSTS[1] =1), the UART will de-assert nRTS signal. 0 = nRTS auto-flow control Disabled. 1 = nRTS auto-flow control Enabled. **Note:** This bit has effect only when the RTSAUDIREN is not set.
[2]	**EVENPARITY**	**Even Parity Enable Bit** 0 = Odd number of logic 1's is transmitted and checked in each word. 1 = Even number of logic 1's is transmitted and checked in each word. **Note:** This bit has effect only when PARITYEN is set.
[1]	**PARITYEN**	**Parity Enable Bit** This bit defines the PARITY bit is enabled in an UART frame. 0 = The PARITY bit Disabled. 1 = The PARITY bit Enabled.
[0]	**STOPB**	**STOP bits** This bit defines the number of STOP bits in an UART frame. 0 = The number of STOP bits is 1.

		1 = The number of STOP bits is 2.

USCI Protocol Interrupt Enable Register – UART (UUART_PROTIEN)

Register	Offset	R/W	Description	Reset Value
UUART_PROTIEN	UUARTn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**RLSIEN**	**ABRIEN**	**Reserved**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**RLSIEN**	**Receive Line Status Interrupt Enable Bit** 0 = Receive line status interrupt Disabled. 1 = Receive line status interrupt Enabled. **Note:** UUART_PROTSTS[7:5] indicates the current interrupt event for receive line status interrupt.
[1]	**ABRIEN**	**Auto-baud Rate Interrupt Enable Bit** 0 = Auto-baud rate interrupt Disabled. 1 = Auto-baud rate interrupt Enabled.
[0]	**Reserved**	Reserved.

USCI Protocol Status Register – UART (UUART_PROTSTS)

Register	Offset	R/W	Description	Reset Value
UUART_PROTSTS	UUARTn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						CTSLV	CTSSYNCLV
15	14	13	12	11	10	9	8
Reserved				ABERRSTS	RXBUSY	ABRDETIF	Reserved
7	6	5	4	3	2	1	0
BREAK	FRMERR	PARITYERR	RXENDIF	RXSTIF	TXENDIF	TXSTIF	Reserved

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**CTSLV**	**nCTS Pin Status (Read Only)** This bit used to monitor the current status of nCTS pin input. 0 = nCTS pin input is low level voltage logic state. 1 = nCTS pin input is high level voltage logic state.
[16]	**CTSSYNCLV**	**nCTS Synchronized Level Status (Read Only)** This bit used to indicate the current status of the internal synchronized nCTS signal. 0 = The internal synchronized nCTS is low. 1 = The internal synchronized nCTS is high.
[15:12]	**Reserved**	Reserved.
[11]	**ABERRSTS**	**Auto-baud Rate Error Status** This bit is set when auto-baud rate detection counter overrun. When the auto-baud rate counter overrun, the user shall revise the CLKDIV (UUART_BRGEN[25:16]) value and enable ABREN (UUART_PROTCTL[6]) to detect the correct baud rate again. 0 = Auto-baud rate detect counter is not overrun. 1 = Auto-baud rate detect counter is overrun. **Note 1:** This bit is set at the same time of ABRDETIF. **Note 2:** This bit can be cleared by writing “1” to ABRDETIF or ABERRSTS.
[10]	**RXBUSY**	**RX Bus Status Flag (Read Only)** This bit indicates the busy status of the receiver. 0 = The receiver is Idle. 1 = The receiver is BUSY.
[9]	**ABRDETIF**	**Auto-baud Rate Interrupt Flag** This bit is set when auto-baud rate detection is done among the falling edge of the input data. If the ABRIEN (UUART_PROTIEN[1]) is set, the auto-baud rate interrupt will be

		generated. This bit can be set 4 times when the input data pattern is 0x55 and it is cleared before the next falling edge of the input bus. 0 = Auto-baud rate detect function is not done. 1 = One Bit auto-baud rate detect function is done. **Note:** This bit can be cleared by writing “1” to it.
[8]	**Reserved**	Reserved.
[7]	**BREAK**	**Break Flag** This bit is set to logic 1 whenever the received data input (RX) is held in the “spacing state” (logic 0) for longer than a full word transmission time (that is, the total time of “START bit” + data bits + parity + STOP bits). 0 = No Break is generated. 1 = Break is generated in the receiver bus. **Note:** This bit can be cleared by writing “1” among the BREAK, FRMERR and PARITYERR bits.
[6]	**FRMERR**	**Framing Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “STOP bit” (that is, the STOP bit following the last data bit or PARITY bit is detected as logic 0). 0 = No framing error is generated. 1 = Framing error is generated. **Note:** This bit can be cleared by writing “1” among the BREAK, FRMERR and PARITYERR bits.
[5]	**PARITYERR**	**Parity Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “PARITY bit”. 0 = No parity error is generated. 1 = Parity error is generated. **Note:** This bit can be cleared by writing “1” among the BREAK, FRMERR and PARITYERR bits.
[4]	**RXENDIF**	**Receive End Interrupt Flag** 0 = A receive finish interrupt status has not occurred. 1 = A receive finish interrupt status has occurred. **Note:** It is cleared by software writing 1 into this bit.
[3]	**RXSTIF**	**Receive Start Interrupt Flag** 0 = A receive start interrupt status has not occurred. 1 = A receive start interrupt status has occurred. **Note:** It is cleared by software writing 1 into this bit.
[2]	**TXENDIF**	**Transmit End Interrupt Flag** 0 = A transmit end interrupt status has not occurred. 1 = A transmit end interrupt status has occurred. **Note:** It is cleared by software writing 1 into this bit.
[1]	**TXSTIF**	**Transmit Start Interrupt Flag** 0 = A transmit start interrupt status has not occurred. 1 = A transmit start interrupt status has occurred. **Note 1:** It is cleared by software writing one into this bit. **Note 2:** Used for user to load next transmit data when there is no data in transmit buffer.
[0]	**Reserved**	Reserved.

6.21 USCI - SPI Mode

6.21.1 Overview

The SPI protocol of USCI controller applies to synchronous serial data communication and allows full duplex transfer. It supports both master and Slave operation mode with the 4-wire bi-direction interface. SPI mode of USCI controller performs a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device. The SPI mode is selected by FUNMODE (USPI_CTL[2:0]) = 0x1

This SPI protocol can operate as Master or Slave mode by setting the SLAVE (USPI_PROTCTL[0]) to communicate with the off-chip SPI Slave or master device. The application block diagrams in Master and Slave mode are shown below.

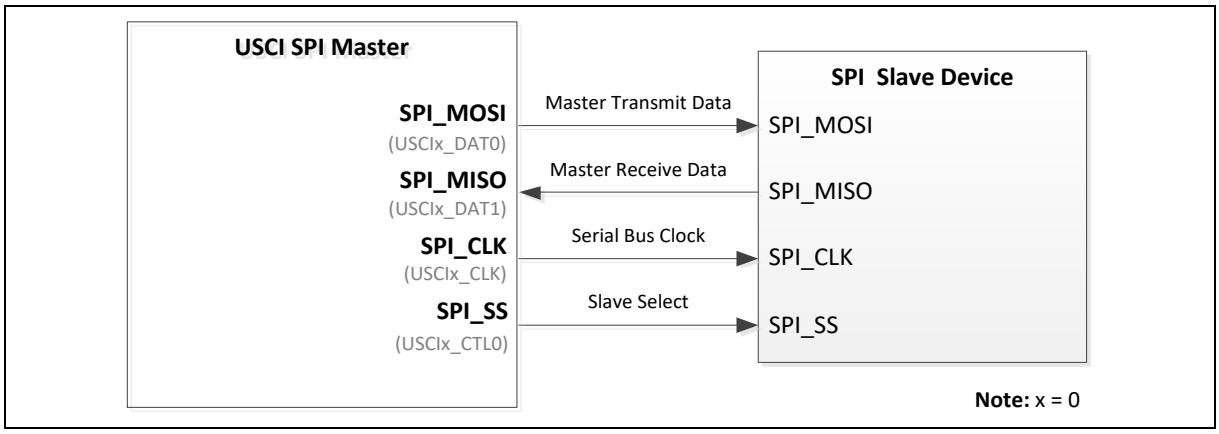


Figure 6.21-1 SPI Master Mode Application Block Diagram

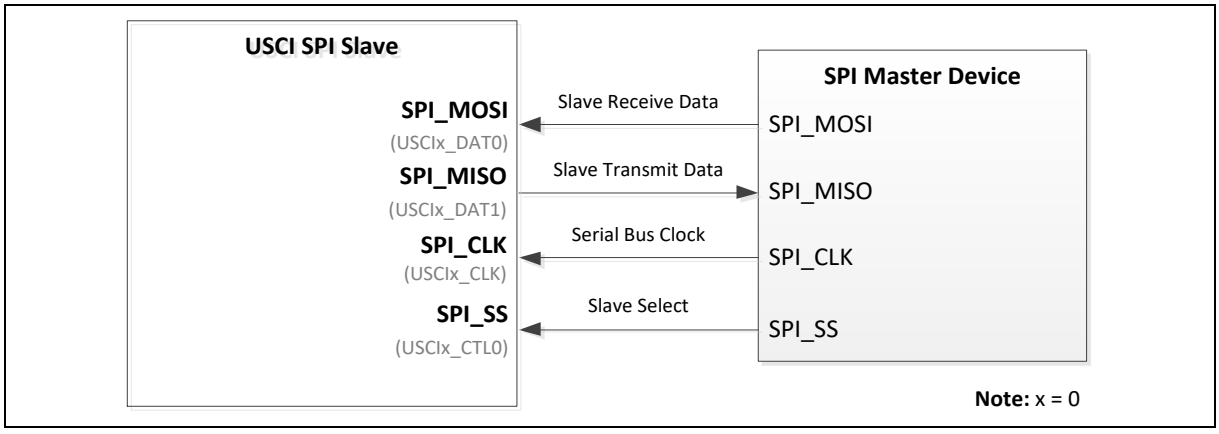


Figure 6.21-2 SPI Slave Mode Application Block Diagram

6.21.2 Features

- Supports Master or Slave mode operation (the maximum frequency -- Master = f_{PCLK} / 2, Slave < f_{PCLK} / 5)
- Configurable bit length of a transfer word from 4 to 16-bit
- Supports one transmit buffer and two receive buffers for data payload
- Supports MSB first or LSB first transfer sequence
- Supports Word Suspend function

- Supports PDMA transfer
- Supports 3-wire, no slave select signal, bi-direction interface
- Supports wake-up function by slave select signal in Slave mode
- Supports one data channel half-duplex transfer

6.21.3 Block Diagram

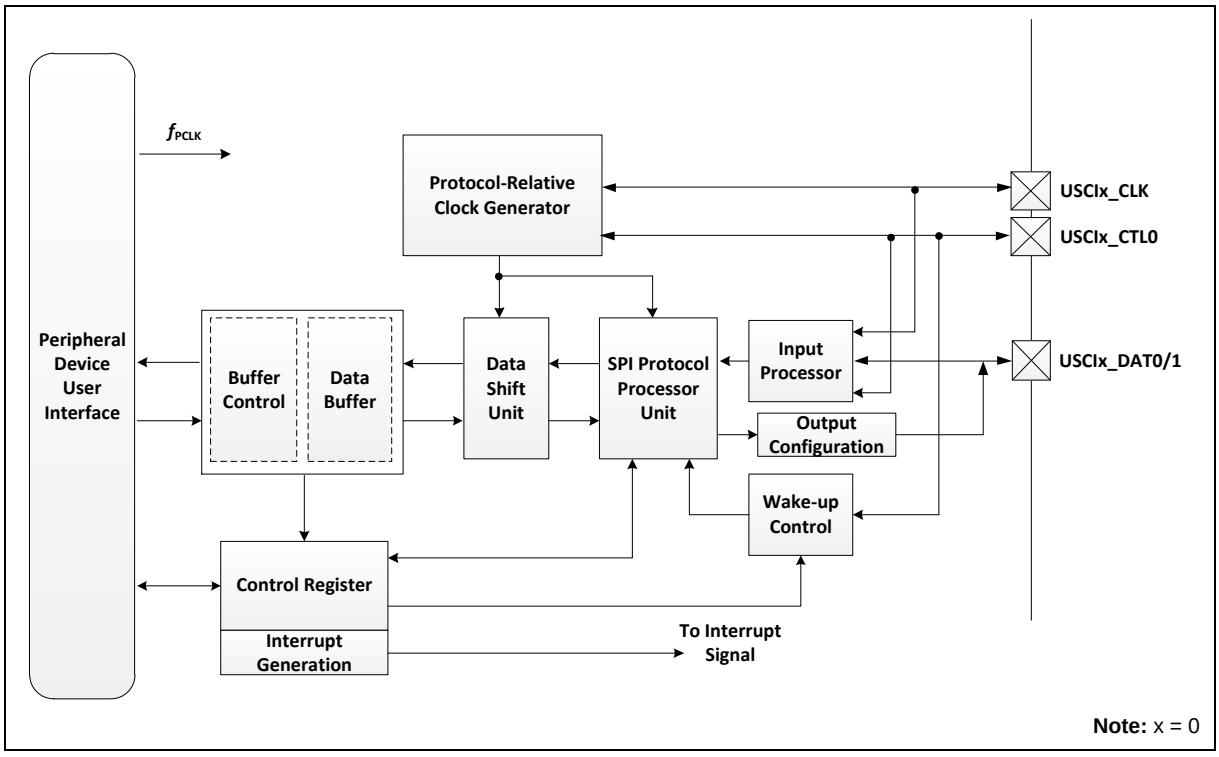


Figure 6.21-3 USCI SPI Mode Block Diagram

6.21.4 Basic Configuration

6.21.4.1 USCI0 SPI Basic Configurations

- Clock Source Configuration
  - Enable USCI0 peripheral clock in USCI0CKEN (CLK_APBCLK1[8]).
  - Enable USCI0_SPI functi on USPI_CTL[2:0] register, USPI_CTL[2:0]=3'b001
- Reset Configuration
  - Reset USCI0 controller in USCI0RST (SYS_IPRST2[8]).

6.21.5 Functional Description

6.21.5.1 USCI Common Functional Description

Please refer to functional description section of USCI for detailed information.

6.21.5.2 Signal Description

A device operating in Master mode controls the start and end of a data transfer, as well as the generation of the SPI bus clock and slave select signal. The slave select signal indicates the start and the end of a data transfer, and the master device can use it to enable the transmitting or receiving operations of Slave device. Slave device receives the SPI bus clock and optionally a slave select signal for data

transaction. The signals for SPI communication are shown below.

SPI Mode	Receive Data	Transmit Data	Serial Bus Clock	Slave Select
Full-duplex SPI Master	SPI_MISO (USCIx_DAT1)	SPI_MOSI (USCIx_DAT0)	SPI_CLK (USCIx_CLK)	SPI_SS (USCIx_CTL0)
Full-duplex SPI Slave	SPI_MOSI (USCIx_DAT0)	SPI_MISO (USCIx_DAT1)	SPI_CLK (USCIx_CLK)	SPI_SS (USCIx_CTL0)
Half-duplex SPI Master/Slave	SPI_MOSI (USCIx_DAT0)	SPI_MOSI (USCIx_DAT0)	SPI_CLK (USCIx_CLK)	SPI_SS (USCIx_CTL0)

SPI Communication Signals

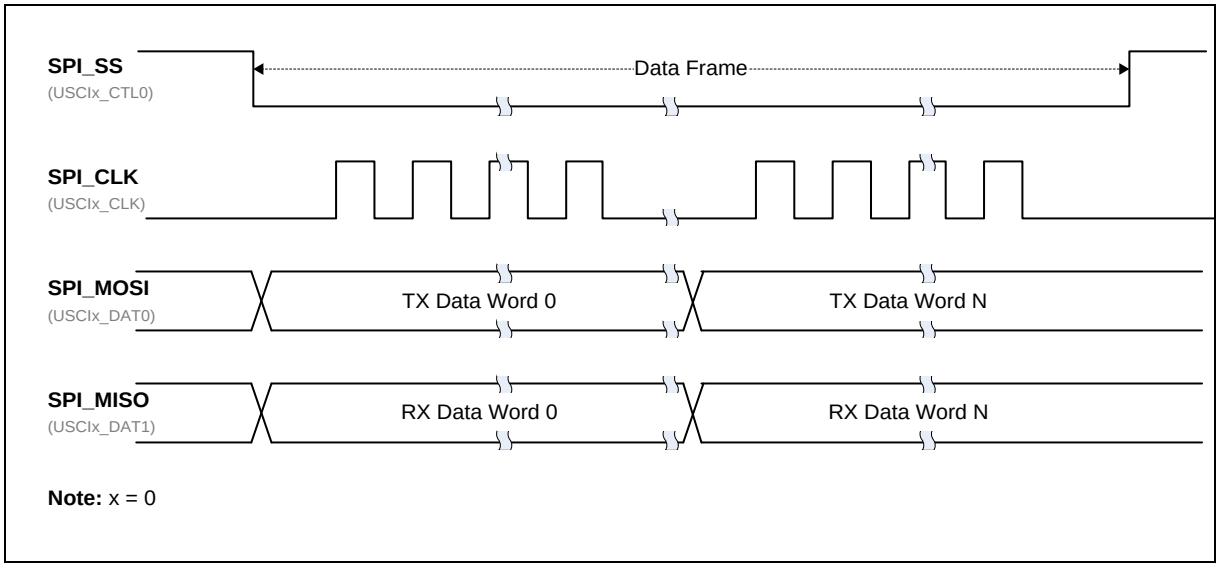


Figure 6.21-44-Wire Full-Duplex SPI Communication Signals (Master Mode)

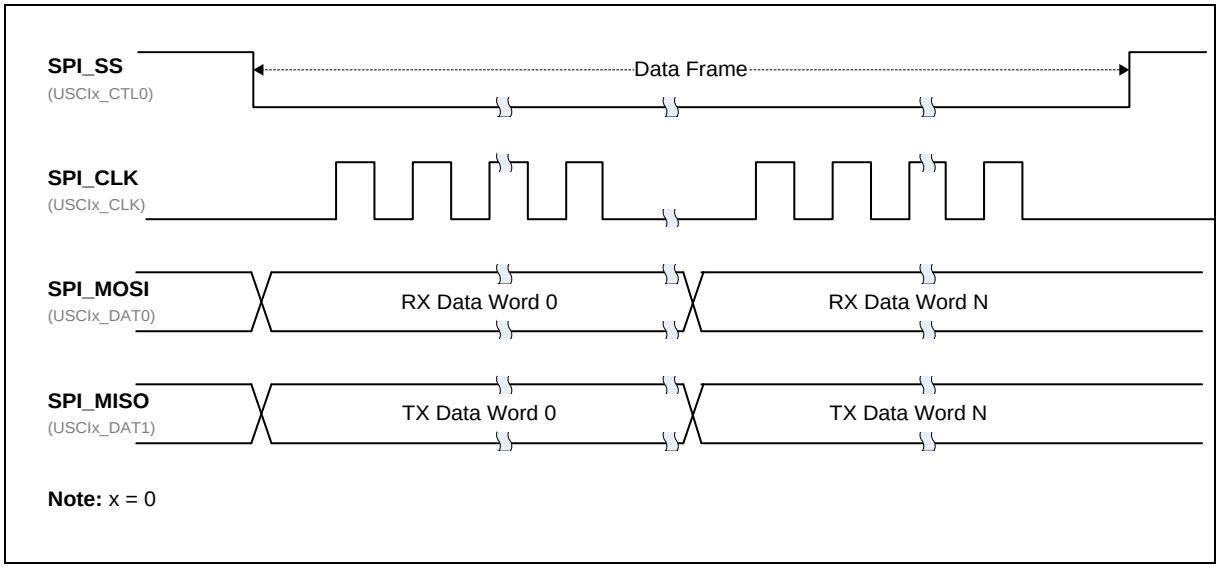


Figure 6.21-54-Wire Full-Duplex SPI Communication Signals (Slave Mode)

6.21.5.3 Serial Bus Clock Configuration

The USCI controller needs the peripheral clock to drive the USCI logic unit to perform the data transfer. The peripheral clock frequency is equal to PCLK frequency.

In Master mode, the frequency of the SPI bus clock is determined by protocol-relative clock generator. In general, the SPI bus clock is denoted as SPI clock. The frequency of SPI clock is half of f_{SAMP_CLK}, which can be selected by SPCLKSEL (USPI_BRGEN[3:2]). Refer to functional description section of USCI for details of protocol-relative clock generator.

In Slave mode, the SPI bus clock is provided by an off-chip Master device. The peripheral clock frequency, f_{PCLK}, of SPI Slave device must be 5-times faster than the serial bus clock rate of the SPI Master device connected together (i.e. the clock rate of serial bus clock < 1/5 peripheral clock f_{PCLK} in Slave mode).

In SPI protocol, SCLKMODE (USPI_PROTCTL[7:6]) defines not only the idle state of serial bus clock but also the serial clock edge used for transmit and receive data. Both Master and Slave devices on the same communication bus should have the same SCLKMODE configuration. The four kinds of serial bus clock configuration are shown below.

SCLKMODE [1:0]	SPI Clock Idle State	Transmit Timing	Receive Timing
0x0	Low	Falling edge	Rising edge
0x1	Low	Rising edge	Falling edge
0x2	High	Rising edge	Falling edge
0x3	High	Falling edge	Rising edge

Table 6.21-1 Serial Bus Clock Configuration

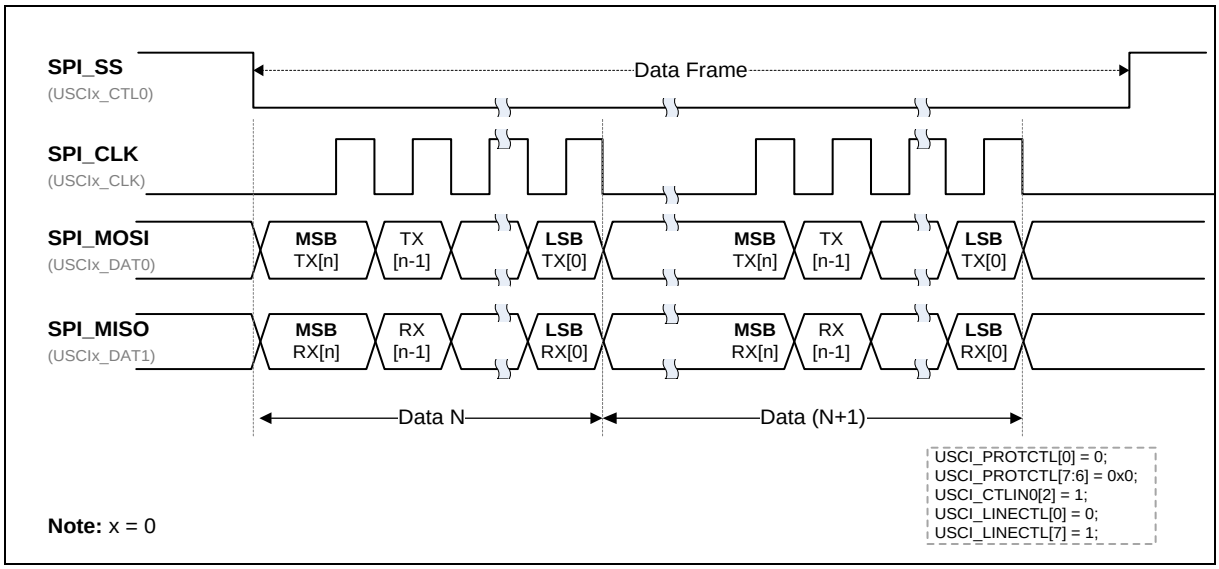


Figure 6.21-6 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x0)

Figure 6.21-7 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x1)

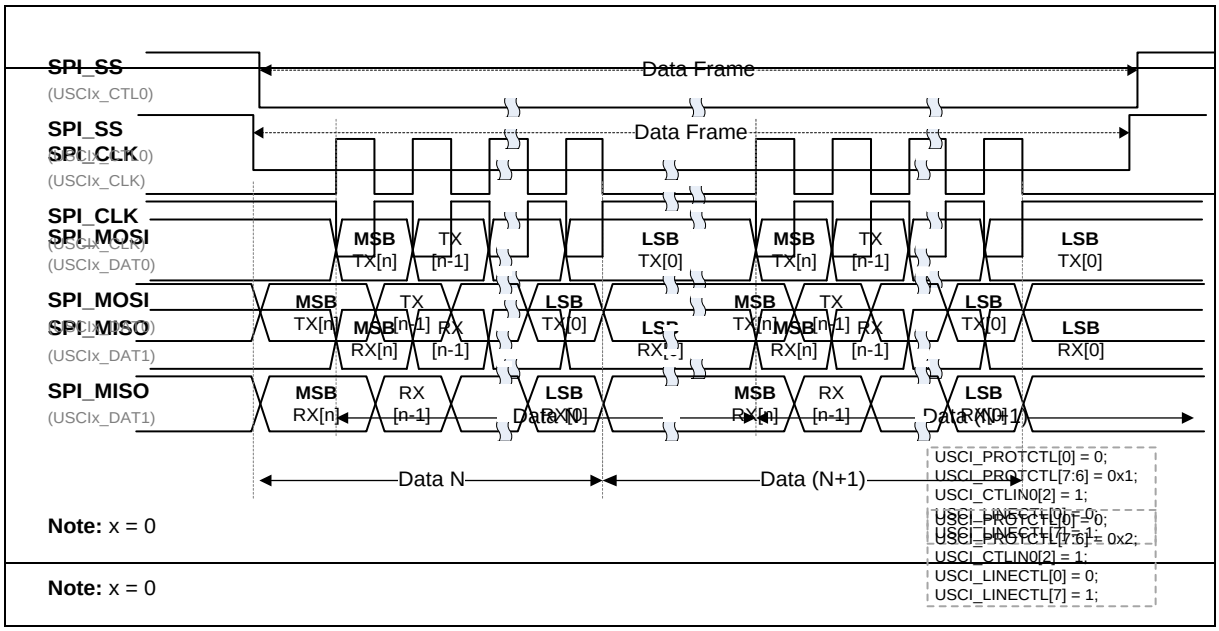


Figure 6.21-8 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x2)

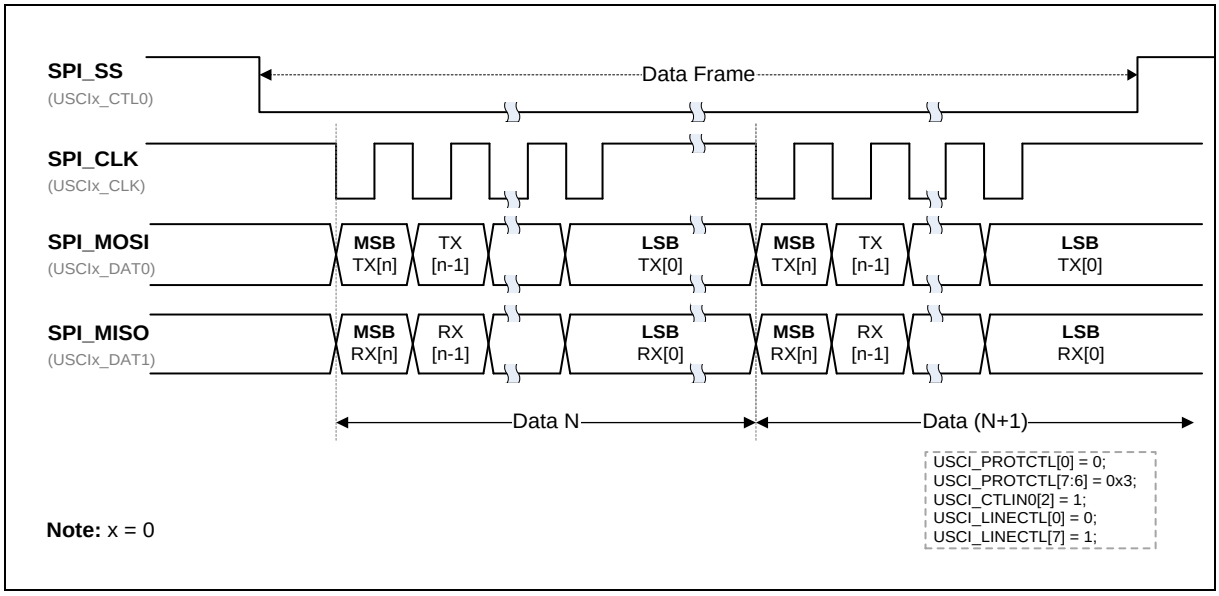


Figure 6.21-9 SPI Communication with Different SPI Clock Configuration (SCLKMODE=0x3)

6.21.5.4 Slave Select Signal

The slave selection signal of SPI protocol is active high by default. In SPI Master mode, the USCI controller can drive the control signal to off-chip SPI Slave device through slave select pin SPI_SS (USCIx_CTL0). In SPI Slave mode, the received slave select signal can be inverted by ININV (USPI_CTLIN0[2]).

If the slave select signal of external SPI Master device is low active, the ININV (USPI_CTLIN0[2]) setting of slave device should be set to 1 for the inversion of input control signal. If USCI operates as SPI Master mode, the output slave select inversion CTLOINV (USPI_LINECTL[7]) is also needed to set as 1 for the

external SPI Slave device whose slave select signal is active low.

The duration between the slave select active edge and the first SPI clock input edge shall over 2 USCI peripheral clock cycles.

The input slave select signal of SPI Slave has to be keep inactive for at least 2 USCI peripheral clock cycles between two consecutive frames in order to correctly detect the end of a frame.

6.21.5.5 Transmit and Receive Data

The bit length of a transmit/receive data word in SPI protocol of USCI controller is defined in DWIDTH (USPI_LINECTL[11:8]), and it can be configured up to 16-bit length for transmitting and receiving data in SPI communication.

The LSB bit (USPI_LINECTL[0]) defines the order of transfer data bit. If the LSB bit is set to 1, the transmission data sequence is LSB first. If the LSB bit is cleared to 0, the transmission data sequence is MSB first.

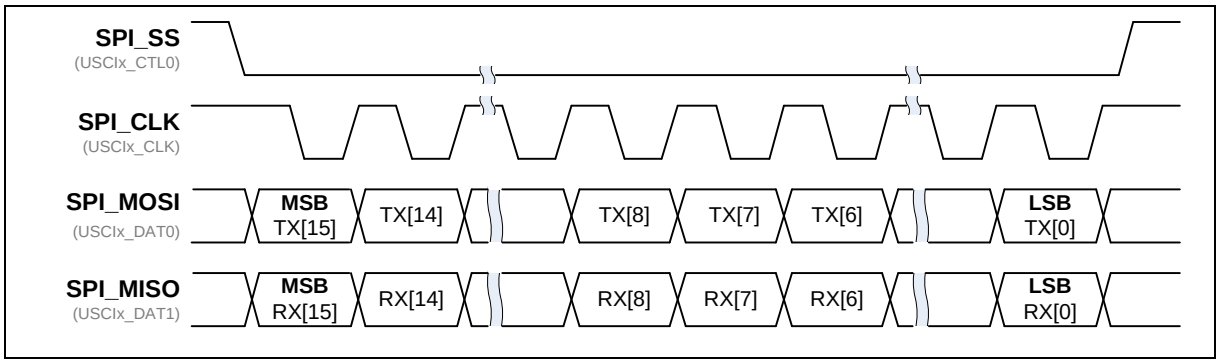


Figure 6.21-1016-bit Data Length in One Word Transaction with MSB First Format

6.21.5.6 Word Suspend

SUSPITV (USPI_PROTCTL[11:8]) provides a configurable suspend interval, 0.5 ~ 15.5 SPI clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SUSPITV (USPI_PROTCTL[11:8]) is 0x3 (3.5 SPI clock cycles).

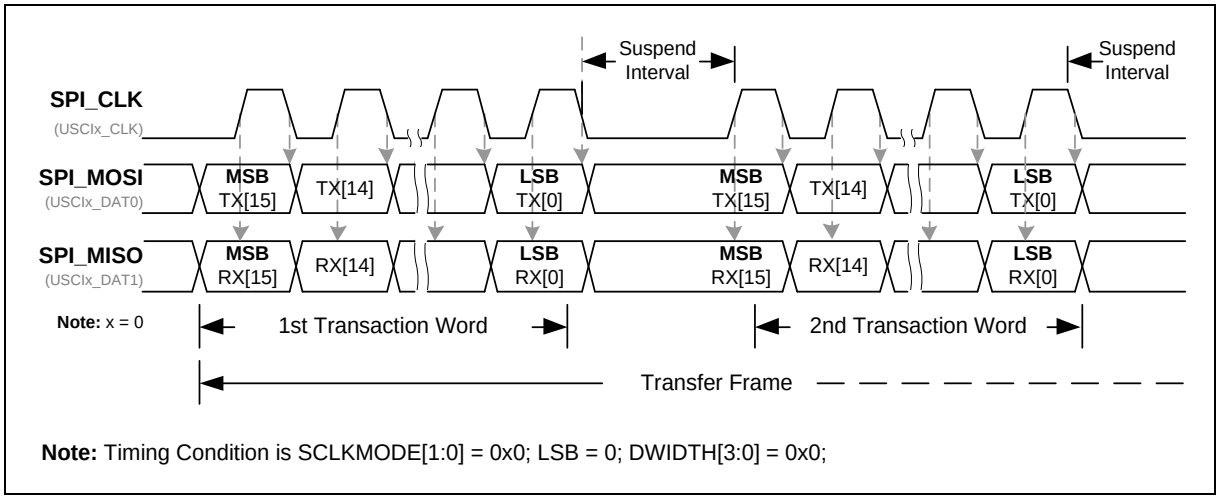


Figure 6.21-11 Word Suspend Interval between Two Transaction Words

6.21.5.7 Automatic Slave Select Function

AUTOSS (USPI_PROTCTL[3]) is used for SPI Master mode to enable the automatic slave select function. If the bit AUTOSS (USPI_PROTCTL[3]) is set, the slave select signal will be generated automatically and the setting value of SS (USPI_PROTCTL[2]) will not affect the output slave select (through USCIx_CTL0 line). This means that the slave select signal will be asserted by the USCI controller when the SPI data transfer is started by writing to the transmit buffer. And, it will be de-asserted after either all transaction is finished or one word transaction done if the value of SUSPITV (USPI_PROTCTL[11:8]) is equal to or great than 3.

If the AUTOSS bit (USPI_PROTCTL[3]) is cleared, the slave select on USCIx_CTL0 pin will be asserted/de-asserted by setting/clearing the SS (USPI_PROTCTL[2]). The internal slave select signal is active high and the CTLOINV (USPI_LINECTL[7]) can be used for the inversion of the slave select signal.

In SPI Master mode, if the value of SUSPITV (USPI_PROTCTL[11:8]) is less than 3 and the AUTOSS (USPI_PROTCTL[3]) is set as 1, the slave select signal will be kept at active state between two successive word transactions.

In SPI Slave mode, to recognize the inactive state of the slave select signal, the inactive period of the received slave select signal must be larger than 2 peripheral clock cycles between two successive transactions.

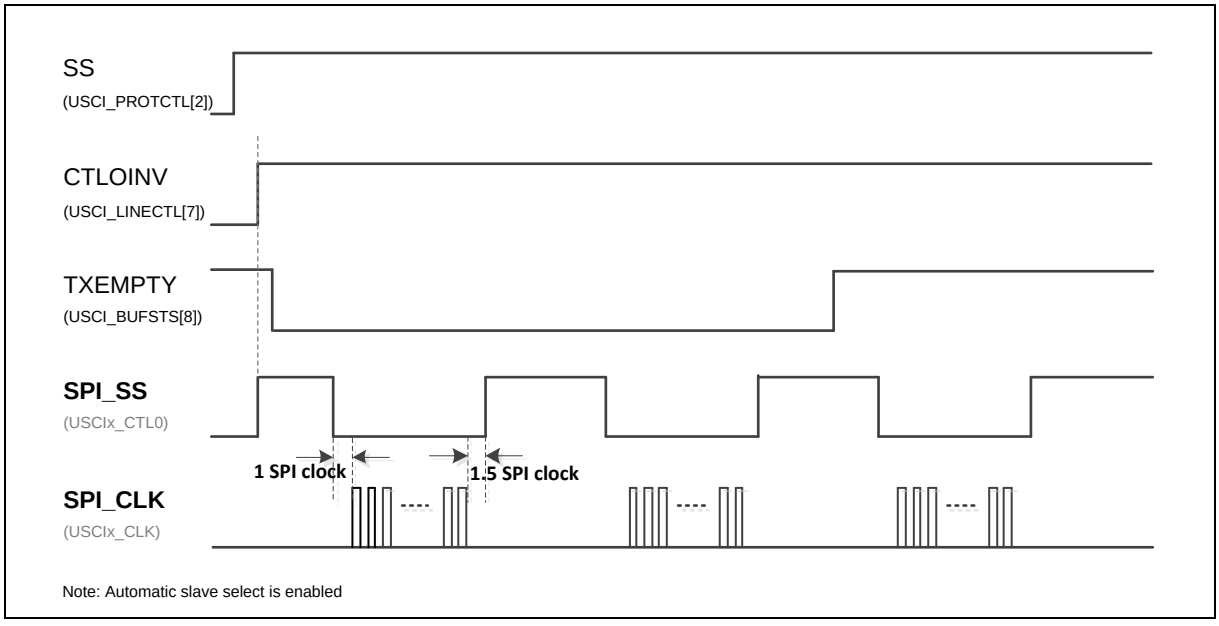


Figure 6.21-12 Auto Slave Select (SUSPITV ≧ 0x3)

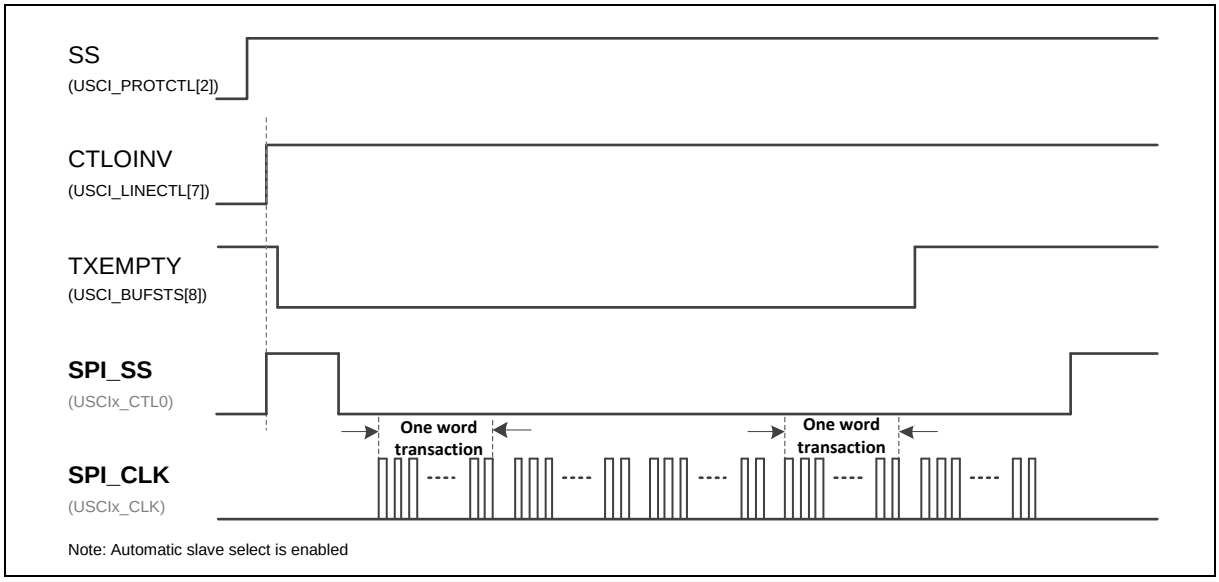


Figure 6.21-13 Auto Slave Select (SUSPITV < 0x3)

6.21.5.8 Slave 3-wire Mode

When the SLV3WIRE (USPI_PROTCTL[1]) is set by software to enable the Slave 3-wire mode, the USCI SPI communication can work with no slave select signal in Slave mode. The SLV3WIRE (USPI_PROTCTL[1]) only takes effect in SPI Slave mode. Only three pins, SPI_CLK (through USCIx_CLK line), SPI_MOSI (through USCIx_DAT0 line), and SPI_MISO (through USCIx_DAT1 line), are required to communicate with a SPI Master. When the SLV3WIRE (USPI_PROTCTL[1]) is set to 1, the SPI Slave will be ready to transmit/receive data after the SPI protocol is enabled by setting FUNMODE(USPI_CTL [2:0]) to 0x1.

6.21.5.9 Data Transfer Mode

The USCI controller supports full-duplex SPI transfer and one data channel half-duplex SPI transfer.

- Full-duplex SPI transfer

  In full-duplex SPI transfer, there are two data pins. One is used for transmitting data and the other is used for receiving data. Thus, data transmission and data reception can be performed simultaneously.

  SCLKMODE (USPI_PROTCTL[7:6]) defines the transition timing of the data shift output signal on USCIx_DAT0 pin. The transition may happen at the corresponding edge of SPI bus clock or active edge of slave select signal. The level of the last data bit of a data word is held on USCIx_DAT0 pin until the next data word begins with the next corresponding edge of the serial bus clock.

- One data channel half-duplex SPI transfer

  In one data channel half-duplex SPI transfer, there is only one data pin for data transfer. Thus, the data transmission and data reception are at different time interval. The data shift direction is determined by PORTDIR (USPI_TXDAT[16]). Refer to the register description for more detail information.

  The function of one data channel half-duplex SPI transfer is similar to the full-duplex SPI protocol. All the transfer data timing is the same as the full-duplex SPI transfer.

  Figure 6.21-14 shows the one output data channel and one input data channel half-duplex transfer diagrams with the external device.

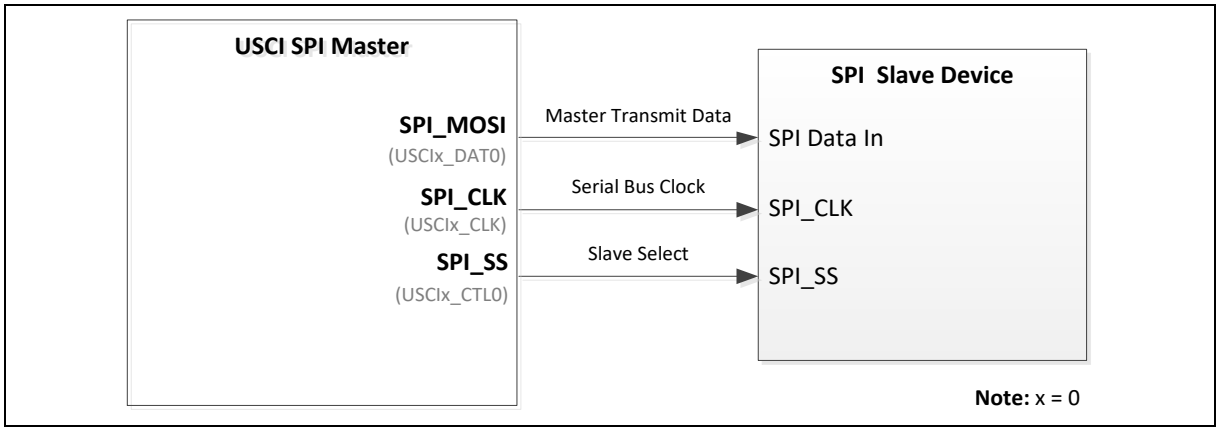


Figure 6.21-14 One Output Data Channel Half-duplex (SPI Master Mode)

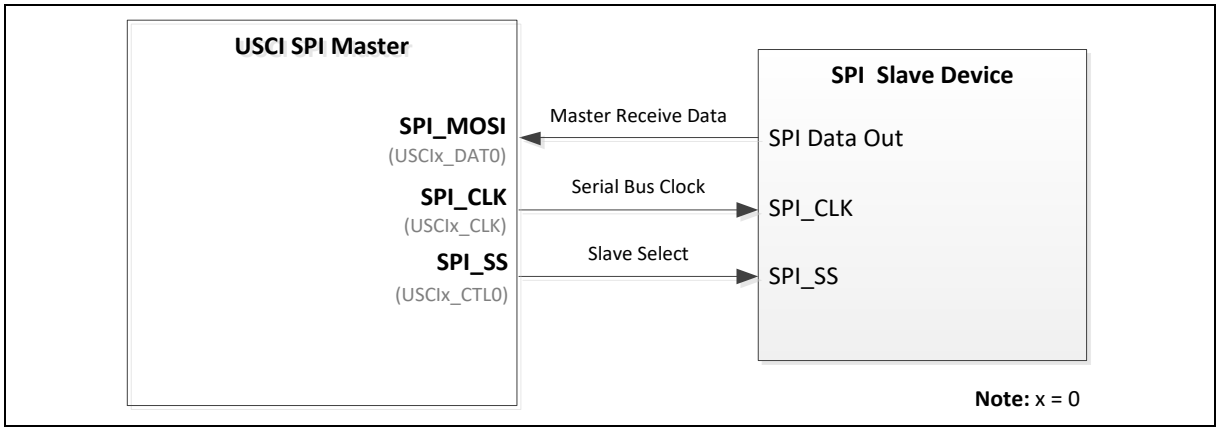


Figure 6.21-15 One Input Data Channel Half-duplex (SPI Master Mode)

The one data channel half-duplex transfer mode can be configured by TSMSEL[2:0] (USPI_PROTCTL[14:12]) and PORTDIR (USPI_TXDAT[16]) settings. When TSMSEL (USPI_PROTCTL[14:12]) is set to 0x4, one data channel half-duplex transfer mode is selected. The PORTDIR (USPI_TXDAT[16]) is used to define the direction of the corresponding transmit data. When the PORTDIR bit is set to 0, the USCI controller will send the corresponding data to external SPI device. When the PORTDIR bit is set to 1, the controller will read the corresponding data from the external SPI device.

For example, in one data channel half-duplex transfer mode with PORTDIR=0, USCI SPI transmits data through USCIx_DAT0 pin; if PORTDIR=1, USCI SPI receives data through USCIx_DAT0 pin.

6.21.5.10 Interrupt

Data Transfer Interrupts

- Transmit start interrupt

  The interrupt event TXSTIF (USPI_PROTSTS[1]) is set after the start of the first data bit of a transmit data word. It can be cleared only by writing 1 to it.

- Transmit end interrupt

  The interrupt event TXENDIF (USPI_PROTSTS[2]) is set after the start of the last data bit of the last transmit data which has been stored in transmit buffer. It can be cleared only by writing 1 to it.

- Receive start interrupt

The interrupt event RXSTIF (USPI_PROTSTS[3]) is set after the start of the first data bit of a receive data word. It can be cleared only by writing 1 to it.

- Receive end interrupt

  The interrupt event RXENDIF (USPI_PROTSTS[4]) is set after the start of the last data bit of a receive data word. It can be cleared only by writing 1 to it.

Protocol-Related Interrupts

- SPI slave select interrupt

  In SPI Slave mode, there are slave select active and in-active interrupt flags, SSACTIF (USPI_PROTSTS[9]) and SSINAIF (USPI_PROTSTS[8]), will be set to 1 when SLAVE (USPI_PROTCTL [0]) is set to 1 and Slave senses the slave select signal active or inactive. The SPI controller will issue an interrupt if SSINAIEN (USPI_PROTIEN[0]) or SSACTIEN (USPI_PROTIEN[1]), are set to 1. Because the internal slave select signal in SPI function is active high, the ININV (USPI_CTLIN0[2]) can be used for inverting the slave select signal comes from an active low device.

- Slave time-out interrupt

  In SPI Slave mode, there is Slave time-out function for user to know that there is no serial clock input during the period of one word transaction. The Slave time-out function uses the timing measurement counter for the calculation of Slave time-out period which is defined by SLVTOCNT (USPI_PROTCTL[25:16]). TMCNTSRC (USPI_BRGEN[5]) can be used for clock frequency selection of timing measurement counter to calculate the Slave time-out period.

  When the timing measurement counter is enabled by TMCNTEN (USPI_BRGEN[4]) and the setting value of SLVTOCNT (USPI_PROTCTL[25:16]) is not 0 in SPI Slave mode, the timing measurement counter will start counting after the first input serial clock of each received word data. This counter will be reset while receiving the following input serial clock and then keep counting. Finally, the timing measurement counter will be cleared and stopped after the finish of the current word transaction. If the value of the time-out counter is equal to or greater than the value of SLVTOCNT (USPI_PROTCTL[25:16]) before one word transaction is done, the Slave time-out interrupt event occurs and the SLVTOIF (USPI_PROTSTS[5]) will be set to 1.

Buffer-Related Interrupts

The buffer-related interrupts are available if there is transmit/receive buffer in USCI controller.

- Receive buffer overrun interrupt

  If there is receive buffer overrun event, RXOVIF (USPI_BUFSTS[3]) will be set as 1. It can be cleared by write 1 into it.

- Transmit buffer under-run interrupt

  If there is transmit buffer under-run event, TXUDRIF (USPI_BUFSTS[11]) will be set as 1. It can be cleared by write 1 into it.

6.21.5.11 Timing Diagram

The slave select signal of USCI SPI protocol is active high by default, and it can be inverted by CTLOINV (USPI_LINECTL[7]) setting.

The idle state of serial bus clock and the serial bus clock edge used for transmit/receive data can be configured by setting SCLKMODE (USPI_PROTCTL[7:6]). The bit length of a transaction word data is determined by DWIDTH (USPI_LINECTL[11:8]), and data bit transfer sequence is determined by LSB (USPI_LINECTL[0]). Four SPI timing diagrams for Master/Slave operations and the related settings are shown below.

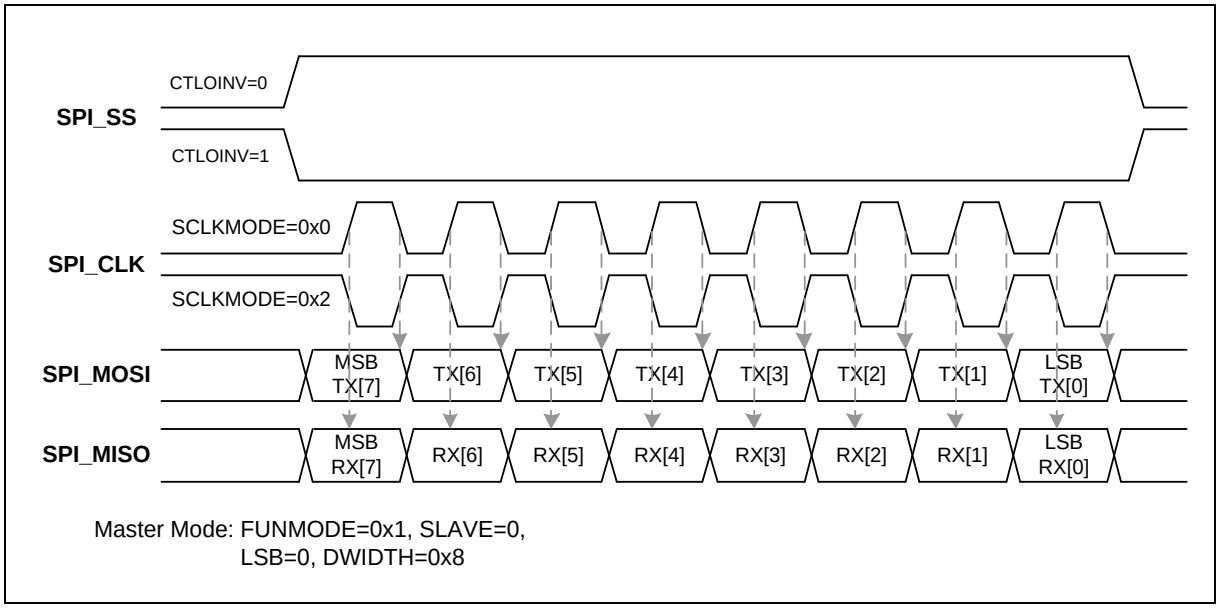


Figure 6.21-16 SPI Timing in Master Mode

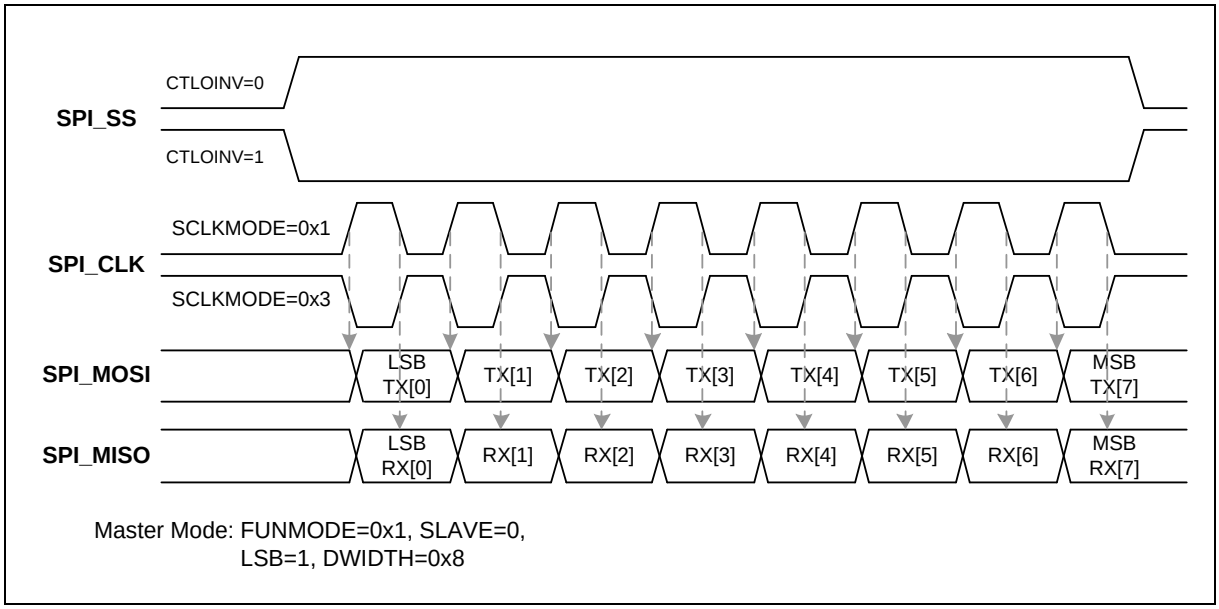


Figure 6.21-17 SPI Timing in Master Mode (Alternate Phase of Serial Bus Clock)

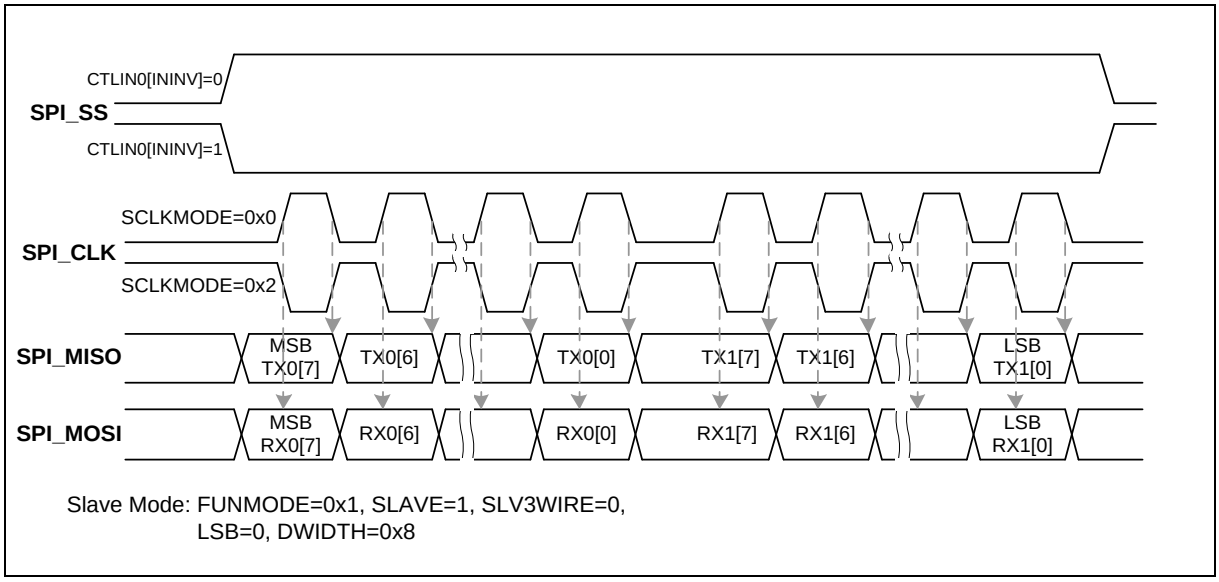


Figure 6.21-18 SPI Timing in Slave Mode

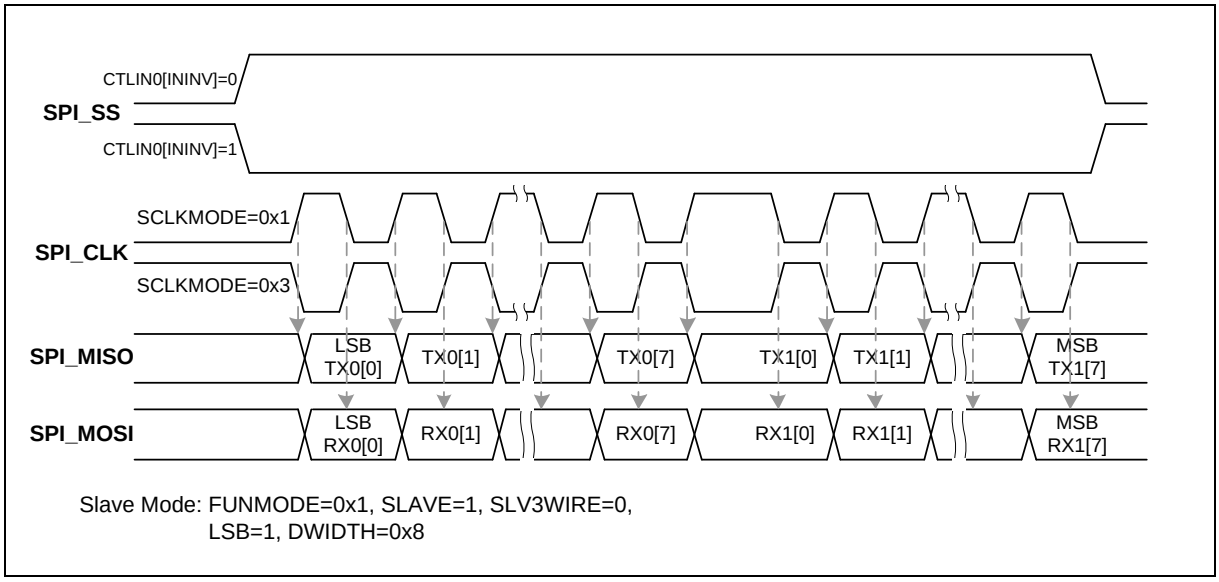


Figure 6.21-19 SPI Timing in Slave Mode (Alternate Phase of Serial Bus Clock)

6.21.5.12 Programming Flow

This section describes the programming flow for USCI SPI data transfer.

For Master mode:

1. Enable USCI peripheral clock by setting CLK_APBCLK1 register.
2. Configure user-specified pins as USCI function pins by setting corresponding multiple function control registers.
3. Set FUNMODE (USPI_CTL[2:0]) to 1 to select SPI mode.
4. Set USPI_BRGEN register to determine the SPI bus clock frequency.
5. According to the requirements of user's application, configure the settings as follows.
   - CTLOINV (USPI_LINECTL[7]): If the slave selection signal is active low, set this bit to 1;

otherwise, set it to 0.

- DWIDTH (USPI_LINECTL[11:8]): Data width setting.
- LSB (USPI_LINECTL[0]): LSB first or MSB first.
- TSMSEL (USPI_PROTCTL[14:12]): Full-duplex SPI transfer or one channel half-duplex SPI transfer.
- SCLKMODE (USPI_PROTCTL[7:6]): Determine the clock timing.
- AUTOSS (USPI_PROTCTL[3]): Enable automatic slave select function or not.
- SLAVE (USPI_PROTCTL[0]): Set to 0 for Master mode.
- Set PROTEN (USPI_PROTCTL[31]) to 1 to enable SPI protocol.

6. If automatic slave select function is disabled (AUTOSS=0), set SS (USPI_PROTCTL[2]) to 1 before data transfer; set SS to 0 to inactivate the slave selection signal by user's application.
7. Write USPI_TXDAT register to trigger SPI transfer. In half-duplex SPI transfer, the data pin direction is determined by PORTDIR (USPI_TXDAT[16]) setting.
8. User can get the received data by reading USPI_RXDAT register as long as RXEMPTY (USPI_BUFSTS[0]) is 0. The SPI data transfer can be triggered by writing USPI_TXDAT register as long as TXFULL (USPI_BUFSTS[9]) is 0.

For Slave mode:

1. Enable USCI peripheral clock by setting CLK_APBCLK1 register.
2. Configure user-specified pins as USCI function pins by setting corresponding multiple function control registers.
3. Set FUNMODE (USPI_CTL[2:0]) to 1 to select SPI mode.
4. According to the requirements of user's application, configure the settings as follows.
   - ININV (USPI_CTLIN0[2]): If the slave selection signal is active low, set this bit to 1; otherwise, set it to 0.
   - DWIDTH (USPI_LINECTL[11:8]): Data width setting.
   - LSB (USPI_LINECTL[0]): LSB first or MSB first.
   - TSMSEL (USPI_PROTCTL[14:12]): Full-duplex SPI transfer or one channel half-duplex SPI transfer.
   - SCLKMODE (USPI_PROTCTL[7:6]): Determine the clock timing.
   - SLAVE (USPI_PROTCTL[0]): Set to 1 for Slave mode.
5. Set PROTEN (USPI_PROTCTL[31]) to 1 to enable SPI protocol.
6. Write USPI_TXDAT register for transmission. In half-duplex SPI transfer, the data pin direction is determined by PORTDIR (USPI_TXDAT[16]) setting.
7. User can get the received data by reading USPI_RXDAT register as long as RXEMPTY (USPI_BUFSTS[0]) is 0. The next datum for transmission can be written to USPI_TXDAT register as long as TXFULL (USPI_BUFSTS[9]) is 0.

6.21.5.13 Wake-up Function

The USCI Controller in SPI mode supports wake-up system function. The wake-up source in SPI protocol is the transition of input slave select signal.

6.21.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
USCI_SPI Base Address: **USPIn_BA = 0x400D_0000 + (0x1000 * n)** **n= 0**				
USPI_CTL	USPIn_BA+0x00	R/W	USCI Control Register	0x0000_0000
USPI_INTEN	USPIn_BA+0x04	R/W	USCI Interrupt Enable Register	0x0000_0000
USPI_BRGEN	USPIn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00
USPI_DATIN0	USPIn_BA+0x10	R/W	USCI Input Data Signal Configuration Register 0	0x0000_0000
USPI_CTLIN0	USPIn_BA+0x20	R/W	USCI Input Control Signal Configuration Register 0	0x0000_0000
USPI_CLKIN	USPIn_BA+0x28	R/W	USCI Input Clock Signal Configuration Register	0x0000_0000
USPI_LINECTL	USPIn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000
USPI_TXDAT	USPIn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000
USPI_RXDAT	USPIn_BA+0x34	R	USCI Receive Data Register	0x0000_0000
USPI_BUFCTL	USPIn_BA+0x38	R/W	USCI Transmit/Receive Buffer Control Register	0x0000_0000
USPI_BUFSTS	USPIn_BA+0x3C	R	USCI Transmit/Receive Buffer Status Register	0x0000_0101
USPI_PDMACTL	USPIn_BA+0x40	R/W	USCI PDMA Control Register	0x0000_0000
USPI_WKCTL	USPIn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000
USPI_WKSTS	USPIn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000
USPI_PROTCTL	USPIn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0300
USPI_PROTIEN	USPIn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000
USPI_PROTSTS	USPIn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

6.21.7 Register Description

USCI Control Register (USPI_CTL)

Register	Offset	R/W	Description	Reset Value
USPI_CTL	USPIn_BA+0x00	R/W	USCI Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**FUNMODE**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**FUNMODE**	**Function Mode** This bit field selects the protocol for this USCI controller. Selecting a protocol that is not available or a reserved combination disables the USCI. When switching between two protocols, the USCI has to be disabled before selecting a new protocol. Simultaneously, the USCI will be reset when user write 000 to FUNMODE. 000 = The USCI is disabled. All protocol related state machines are set to idle state. 001 = The SPI protocol is selected. 010 = The UART protocol is selected. 100 = The I^2C protocol is selected. **Note:** Other bit combinations are reserved.

USCI Interrupt Enable Register (USPI_INTEN)

Register	Offset	R/W	Description	Reset Value
USPI_INTEN	USPIn_BA+0x04	R/W	USCI Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**RXENDIEN**	**RXSTIEN**	**TXENDIEN**	**TXSTIEN**	**Reserved**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**RXENDIEN**	**Receive End Interrupt Enable Bit** This bit enables the interrupt generation in case of a receive finish event. 0 = The receive end interrupt Disabled. 1 = The receive end interrupt Enabled. **Note:** The receive finish event happens when hardware receives the last bit of RX data into shift data unit.
[3]	**RXSTIEN**	**Receive Start Interrupt Enable Bit** This bit enables the interrupt generation in case of a receive start event. 0 = The receive start interrupt Disabled. 1 = The receive start interrupt Enabled. **Note:** For SPI master mode, the receive start event happens when SPI master sends slave select active and spi clock to the external SPI slave. For SPI slave mode, the receive start event happens when slave select of SPI slave is active and spi clock of SPI slave is inputed from the external SPI master.
[2]	**TXENDIEN**	**Transmit End Interrupt Enable Bit** This bit enables the interrupt generation in case of a transmit finish event. 0 = The transmit finish interrupt Disabled. 1 = The transmit finish interrupt Enabled. **Note:** The transmit finish event happens when hardware sends the last bit of TX data from shift data unit.
[1]	**TXSTIEN**	**Transmit Start Interrupt Enable Bit** This bit enables the interrupt generation in case of a transmit start event. 0 = The transmit start interrupt Disabled. 1 = The transmit start interrupt Enabled. **Note:** The transmit start event happens when hardware starts to move TX data from data buffer to shift data unit.
[0]	**Reserved**	Reserved.

USCI Baud Rate Generator Register (USPI_BRGEN)

Register	Offset	R/W	Description	Reset Value
USPI_BRGEN	USPIn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00

31	30	29	28	27	26	25	24
Reserved						CLKDIV	
23	22	21	20	19	18	17	16
CLKDIV							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		TMCNTSRC	TMCNTEN	SPCLKSEL		PTCLKSEL	RCLKSEL

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25:16]	**CLKDIV**	**Clock Divider** This bit field defines the ratio between the protocol clock frequency f_{PROT_CLK} and the clock divider frequency f_{DIV_CLK} (fDIV_CLK = fPROT_CLK / (CLKDIV+1)).
[15:6]	**Reserved**	Reserved.
[5]	**TMCNTSRC**	**Time Measurement Counter Clock Source Selection** 0 = Time measurement counter with f_{PROT_CLK}. 1 = Time measurement counter with f_{DIV_CLK}.
[4]	**TMCNTEN**	**Time Measurement Counter Enable Bit** This bit enables the 10-bit timing measurement counter. 0 = Time measurement counter Disabled. 1 = Time measurement counter Enabled.
[3:2]	**SPCLKSEL**	**Sample Clock Source Selection** This bit field used for the clock source selection of sample clock (f_{SAMP_CLK}) for the protocol processor. 00 = f_{DIV_CLK}. 01 = f_{PROT_CLK}. 10 = f_{SCLK}. 11 = f_{REF_CLK}.
[1]	**PTCLKSEL**	**Protocol Clock Source Selection** This bit selects the source of protocol clock (f_{PROT_CLK}). 0 = Reference clock f_{REF_CLK}. 1 = f_{REF_CLK2} (its frequency is half of f_{REF_CLK}).
[0]	**RCLKSEL**	**Reference Clock Source Selection** This bit selects the source of reference clock (f_{REF_CLK}). 0 = Peripheral device clock f_{PCLK}. 1 = Reserved.

USCI Input Data Signal Configuration (USPI_DATIN0)

Register	Offset	R/W	Description	Reset Value
USPI_DATIN0	USPIn_BA+0x10	R/W	USCI Input Data Signal Configuration Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**ININV**	**Reserved**	**SYNCSEL**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**ININV**	**Input Signal Inverse Selection** This bit defines the inverter enable of the input asynchronous signal. 0 = The un-synchronized input signal will not be inverted. 1 = The un-synchronized input signal will be inverted. **Note:** In SPI protocol, it is suggested this bit should be set as 0.
[1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Signal Synchronization Selection** This bit selects if the un-synchronized input signal (with optionally inverted) or the synchronized (and optionally filtered) signal, which is synchronized with PCLK, can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit. **Note:** In SPI protocol, it is suggested this bit should be set as 0.

USCI Input Control Signal Configuration (USPI_CTLIN0)

Register	Offset	R/W	Description	Reset Value
USPI_CTLIN0	USPIn_BA+0x20	R/W	USCI Input Control Signal Configuration Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**ININV**	**Reserved**	**SYNCSEL**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**ININV**	**Input Signal Inverse Selection** This bit defines the inverter enable of the input asynchronous signal. 0 = The un-synchronized input signal will not be inverted. 1 = The un-synchronized input signal will be inverted.
[1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Synchronization Signal Selection** This bit selects if the un-synchronized input signal (with optionally inverted) or the synchronized (and optionally filtered) signal, which is synchronized with PCLK, can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit. **Note:** In SPI protocol, it is suggested this bit should be set as 0.

USCI Input Clock Signal Configuration (USPI_CLKIN)

Register	Offset	R/W	Description	Reset Value
USPI_CLKIN	USPIn_BA+0x28	R/W	USCI Input Clock Signal Configuration Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**SYNCSEL**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SYNCSEL**	**Input Synchronization Signal Selection** This bit selects if the un-synchronized input signal or the synchronized (and optionally filtered) signal, which is synchronized with PCLK, can be used as input for the data shift unit. 0 = The un-synchronized signal can be taken as input for the data shift unit. 1 = The synchronized signal can be taken as input for the data shift unit. **Note:** In SPI protocol, it is suggested this bit should be set as 0.

USCI Line Control Register (USPI_LINECTL)

Register	Offset	R/W	Description	Reset Value
USPI_LINECTL	USPIn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**DWIDTH**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CTLOINV	**Reserved**	**DATOINV**	**Reserved**				**LSB**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**DWIDTH**	**Word Length of Transmission** This bit field defines the data word length (amount of bits) for reception and transmission. The data word is always right-aligned in the data buffer. USCI support word length from 4 to 16 bits. 0x0: The data word contains 16 bits located at bit positions [15:0]. 0x1: Reserved. 0x2: Reserved. 0x3: Reserved. 0x4: The data word contains 4 bits located at bit positions [3:0]. 0x5: The data word contains 5 bits located at bit positions [4:0]. ... 0xF: The data word contains 15 bits located at bit positions [14:0].
[7]	**CTLOINV**	**Control Signal Output Inverse Selection** This bit defines the relation between the internal control signal and the output control signal. 0 = No effect. 1 = The control signal will be inverted before its output. **Note:** The control signal has different definitions in different protocol. In SPI protocol, the control signal means slave select signal.
[6]	**Reserved**	Reserved.
[5]	**DATOINV**	**Data Output Inverse Selection** This bit defines the relation between the internal shift data value and the output data signal of USCIx_DAT0/1 pins. 0 = Data output values of USCIx_DAT0/1 pins are not inverted. 1 = Data output values of USCIx_DAT0/1 pins are inverted.
[4:1]	**Reserved**	Reserved.
[0]	**LSB**	**LSB First Transmission Selection**

		0 = The MSB, which bit of transmit/receive data buffer depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, the bit 0 of data buffer, will be transmitted/received first.

USCI Transmit Data Register (USPI_TXDAT)

Register	Offset	R/W	Description	Reset Value
USPI_TXDAT	USPIn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**PORTDIR**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXDAT							

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**PORTDIR**	**Port Direction Control** This bit field is only available while USCI operates in SPI protocol (FUNMODE = 0x1) with half-duplex transfer. It is used to define the direction of the data port pin. When software writes USPI_TXDAT register, the transmit data and its port direction are settled simultaneously. 0 = The data pin is configured as output mode. 1 = The data pin is configured as input mode.
[15:0]	**TXDAT**	**Transmit Data** Software can use this bit field to write 16-bit transmit data for transmission. In order to avoid overwriting the transmit data, user have to check TXEMPTY (USPI_BUFSTS[8]) status before writing transmit data into this bit field.

USCI Receive Data Register (USPI_RXDAT)

Register	Offset	R/W	Description	Reset Value
USPI_RXDAT	USPIn_BA+0x34	R	USCI Receive Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RXDAT**	**Received Data** This bit field monitors the received data which stored in receive data buffer.

USCI Transmitter/Receive Buffer Control Register (USPI_BUFCTL)

Register	Offset	R/W	Description	Reset Value
USPI_BUFCTL	USPIn_BA+0x38	R/W	USCI Transmit/Receive Buffer Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						RXRST	TXRST
15	14	13	12	11	10	9	8
RXCLR	RXOVIEN	Reserved					
7	6	5	4	3	2	1	0
TXCLR	TXUDRIEN	Reserved					

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset the receive-related counters, state machine, and the content of receive shift register and data buffer. **Note:** It is cleared automatically after one PCLK cycle.
[16]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset the transmit-related counters, state machine, and the content of transmit shift register and data buffer**.** **Note 1:** It is cleared automatically after one PCLK cycle. **Note 2:** Write 1 to this bit will set the output data pin to zero if USPI_PROTCTL[28]=0.
[15]	**RXCLR**	**Clear Receive Buffer** 0 = No effect. 1 = The receive buffer is cleared. Should only be used while the buffer is not taking part in data traffic. **Note:** It is cleared automatically after one PCLK cycle.
[14]	**RXOVIEN**	**Receive Buffer Overrun Interrupt Enable Bit** 0 = Receive overrun interrupt Disabled. 1 = Receive overrun interrupt Enabled.
[13:8]	**Reserved**	Reserved.
[7]	**TXCLR**	**Clear Transmit Buffer** 0 = No effect. 1 = The transmit buffer is cleared. Should only be used while the buffer is not taking part in data traffic. **Note:** It is cleared automatically after one PCLK cycle.
[6]	**TXUDRIEN**	**Slave Transmit Under Run Interrupt Enable Bit**

		0 = Transmit under-run interrupt Disabled. 1 = Transmit under-run interrupt Enabled.
[5:0]	**Reserved**	Reserved.

USCI Transmit/Receive Buffer Status Register (USPI_BUFSTS)

Register	Offset	R/W	Description	Reset Value
USPI_BUFSTS	USPIn_BA+0x3C	R	USCI Transmit/Receive Buffer Status Register	0x0000_0101

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**TXUDRIF**	**Reserved**	**TXFULL**	**TXEMPTY**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**RXOVIF**	**Reserved**	**RXFULL**	**RXEMPTY**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11]	**TXUDRIF**	**Transmit Buffer Under-run Interrupt Status** This bit indicates that a transmit buffer under-run event has been detected. If enabled by TXUDRIEN (USPI_BUFCTL[6]), the corresponding interrupt request is activated. It is cleared by software writes 1 to this bit 0 = A transmit buffer under-run event has not been detected. 1 = A transmit buffer under-run event has been detected.
[10]	**Reserved**	Reserved.
[9]	**TXFULL**	**Transmit Buffer Full Indicator** 0 = Transmit buffer is not full. 1 = Transmit buffer is full.
[8]	**TXEMPTY**	**Transmit Buffer Empty Indicator** 0 = Transmit buffer is not empty. 1 = Transmit buffer is empty and available for the next transmission datum.
[7:4]	**Reserved**	Reserved.
[3]	**RXOVIF**	**Receive Buffer Over-run Interrupt Status** This bit indicates that a receive buffer overrun event has been detected. If RXOVIEN (USPI_BUFCTL[14]) is enabled, the corresponding interrupt request is activated. It is cleared by software writes 1 to this bit. 0 = A receive buffer overrun event has not been detected. 1 = A receive buffer overrun event has been detected.
[2]	**Reserved**	Reserved.
[1]	**RXFULL**	**Receive Buffer Full Indicator** 0 = Receive buffer is not full. 1 = Receive buffer is full.
[0]	**RXEMPTY**	**Receive Buffer Empty Indicator** 0 = Receive buffer is not empty.

		1 = Receive buffer is empty.

USCI PDMA Control Register (USPI_PDMACTL)

Register	Offset	R/W	Description	Reset Value
USPI_PDMACTL	USPIn_BA+0x40	R/W	USCI PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PDMAEN**	**RXPDMAEN**	**TXPDMAEN**	**PDMARST**

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**PDMAEN**	**PDMA Mode Enable Bit** 0 = PDMA function Disabled. 1 = PDMA function Enabled.
[2]	**RXPDMAEN**	**PDMA Receive Channel Available** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[1]	**TXPDMAEN**	**PDMA Transmit Channel Available** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled.
[0]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the USCI's PDMA control logic. This bit will be cleared to 0 automatically.

USCI Wake-up Control Register (USPI_WKCTL)

Register	Offset	R/W	Description	Reset Value
USPI_WKCTL	USPIn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**PDBOPT**	**Reserved**	**WKEN**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDBOPT**	**Power Down Blocking Option** 0 = If user attempts to enter Power-down mode by executing WFI while the protocol is in transferring, MCU will stop the transfer and enter Power-down mode immediately. 1 = If user attempts to enter Power-down mode by executing WFI while the protocol is in transferring, the on-going transfer will not be stopped and MCU will enter idle mode immediately.
[1]	**Reserved**	Reserved.
[0]	**WKEN**	**Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.

USCI Wake-up Status Register (USPI_WKSTS)

Register	Offset	R/W	Description	Reset Value
USPI_WKSTS	USPIn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**WKF**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKF**	**Wake-up Flag** When chip is woken up from Power-down mode, this bit is set to 1. Software can write 1 to clear this bit.

USCI Protocol Control Register – USPI_PROTCTL (SPI)

Register	Offset	R/W	Description	Reset Value
USPI_PROTCTL	USPIn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0300

31	30	29	28	27	26	25	24
PROTEN	Reserved		TXUDRPOL	Reserved		SLVTOCNT	
23	22	21	20	19	18	17	16
SLVTOCNT							
15	14	13	12	11	10	9	8
Reserved	TSMSEL			SUSPITV			
7	6	5	4	3	2	1	0
SCLKMODE		Reserved		AUTOSS	SS	SLV3WIRE	SLAVE

Bits	Description	
[31]	**PROTEN**	**SPI Protocol Enable Bit** 0 = SPI Protocol Disabled. 1 = SPI Protocol Enabled.
[30:29]	**Reserved**	Reserved.
[28]	**TXUDRPOL**	**Transmit Under-run Data Polarity for Slave** This bit defines the transmitting data value of USCIx_DAT1 when no data is available for transferring. 0 = The output data value is 0 if TX under run event occurs. 1 = The output data value is 1 if TX under run event occurs.
[27:26]	**Reserved**	Reserved.
[25:16]	**SLVTOCNT**	**Slave Mode Time-out Period for Slave Only** In Slave mode, this bit field is used for Slave time-out period. This bit field indicates how many clock periods (selected by TMCNTSRC, USPI_BRGEN[5]) between the two edges of input SCLK will assert the Slave time-out event. Writing 0x0 into this bit field will disable the Slave time-out function. Example: Assume SLVTOCNT is 0x0A and TMCNTSRC (USPI_BRGEN[5]) is 1, it means the time-out event will occur if the state of SPI bus clock pin is not changed more than (10+1) periods of f_{DIV_CLK}.
[15]	**Reserved**	Reserved.
[14:12]	**TSMSEL**	**Transmit Data Mode Selection** This bit field describes how receive and transmit data is shifted in and out. TSMSEL = 000b: Full-duplex SPI. TSMSEL = 100b: Half-duplex SPI. Other values are reserved. **Note:** Changing the value of this bit field will produce the TXRST and RXRST to clear the TX/RX data buffer automatically.
[11:8]	**SUSPITV**	**Suspend Interval for Master Only** This bit field provides the configurable suspend interval between two successive transmit/receive transaction in a transfer. The definition of the suspend interval is the

		interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the suspend interval is obtained according to the following equation. (SUSPITV[3:0] + 0.5) * period of SPI_CLK clock cycle Example: SUSPITV = 0x0 … 0.5 SPI_CLK clock cycle. SUSPITV = 0x1 … 1.5 SPI_CLK clock cycle. …… SUSPITV = 0xE … 14.5 SPI_CLK clock cycle. SUSPITV = 0xF … 15.5 SPI_CLK clock cycle.
[7:6]	**SCLKMODE**	**Serial Bus Clock Mode** This bit field defines the SCLK idle status, data transmit, and data receive edge. MODE0 = The idle state of SPI clock is low level. Data is transmitted with falling edge and received with rising edge. MODE1 = The idle state of SPI clock is low level. Data is transmitted with rising edge and received with falling edge. MODE2 = The idle state of SPI clock is high level. Data is transmitted with rising edge and received with falling edge. MODE3 = The idle state of SPI clock is high level. Data is transmitted with falling edge and received with rising edge.
[5:4]	**Reserved**	Reserved.
[3]	**AUTOSS**	**Automatic Slave Select Function Enable for Master Only** 0 = Slave select signal will be controlled by the setting value of SS (USPI_PROTCTL[2]) bit. 1 = Slave select signal will be generated automatically. The slave select signal will be asserted by the SPI controller when transmit/receive is started, and will be de-asserted after each transmit/receive is finished.
[2]	**SS**	**Slave Select Control for Master Only** If AUTOSS bit is cleared, setting this bit to 1 will set the slave select signal to active state, and setting this bit to 0 will set the slave select back to inactive state. If the AUTOSS function is enabled (AUTOSS = 1), the setting value of this bit will not affect the current state of slave select signal. **Note:** In SPI protocol, the internal slave select signal is active high.
[1]	**SLV3WIRE**	**Slave 3-wire Mode Selection for Slave Only** The SPI protocol can work with 3-wire interface (without slave select signal) in Slave mode. 0 = 4-wire bi-direction interface. 1 = 3-wire bi-direction interface.
[0]	**SLAVE**	**Slave Mode Selection** 0 = Master mode. 1 = Slave mode.

USCI Protocol Interrupt Enable Register – USPI_PROTIEN (SPI)

Register	Offset	R/W	Description	Reset Value
USPI_PROTIEN	USPIn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**SLVBEIEN**	**SLVTOIEN**	**SSACTIEN**	**SSINAIEN**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[3]	**SLVBEIEN**	**Slave Mode Bit Count Error Interrupt Enable Bit** If data transfer is terminated by slave time-out or slave select inactive event in Slave mode, so that the transmit/receive data bit count does not match the setting of DWIDTH (USPI_LINECTL[11:8]). Bit count error event occurs. 0 = The Slave mode bit count error interrupt Disabled. 1 = The Slave mode bit count error interrupt Enabled.
[2]	**SLVTOIEN**	**Slave Time-out Interrupt Enable Bit** In SPI protocol, this bit enables the interrupt generation in case of a Slave time-out event. 0 = The Slave time-out interrupt Disabled. 1 = The Slave time-out interrupt Enabled.
[1]	**SSACTIEN**	**Slave Select Active Interrupt Enable Bit** This bit enables/disables the generation of a slave select interrupt if the slave select changes to active. 0 = Slave select active interrupt generation Disabled. 1 = Slave select active interrupt generation Enabled.
[0]	**SSINAIEN**	**Slave Select Inactive Interrupt Enable Bit** This bit enables/disables the generation of a slave select interrupt if the slave select changes to inactive. 0 = Slave select inactive interrupt generation Disabled. 1 = Slave select inactive interrupt generation Enabled.

USCI Protocol Status Register – USPI_PROTSTS (SPI)

Register	Offset	R/W	Description	Reset Value
USPI_PROTSTS	USPIn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**SLVUDR**	**BUSY**	**SSLINE**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**SSACTIF**	**SSINAIF**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**SLVBEIF**	**SLVTOIF**	**RXENDIF**	**RXSTIF**	**TXENDIF**	**TXSTIF**	**Reserved**

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**SLVUDR**	**Slave Mode Transmit Under-run Status (Read Only)** In Slave mode, if there is no available transmit data in buffer while transmit data shift out caused by input serial bus clock, this status flag will be set to 1. This bit indicates whether the current shift-out data of word transmission is switched to TXUDRPOL (USPI_PROTCTL[28]) or not. 0 = Slave transmit under run event does not occur. 1 = Slave transmit under run event occurs.
[17]	**BUSY**	**Busy Status (Read Only)** 0 = SPI is in idle state. 1 = SPI is in busy state. The following listing are the bus busy conditions: a. USPI_PROTCTL[31] = 1 and the TXEMPTY = 0. b. For SPI Master mode, the TXEMPTY = 1 but the current transaction is not finished yet. c. For SPI Slave mode, the USPI_PROTCTL[31] = 1 and there is serial clock input into the SPI core logic when slave select is active. d. For SPI Slave mode, the USPI_PROTCTL[31] = 1 and the transmit buffer or transmit shift register is not empty even if the slave select is inactive.
[16]	**SSLINE**	**Slave Select Line Bus Status (Read Only)** This bit is only available in Slave mode. It used to monitor the current status of the input slave select signal on the bus. 0 = The slave select line status is 0. 1 = The slave select line status is 1.
[15:10]	**Reserved**	Reserved.
[9]	**SSACTIF**	**Slave Select Active Interrupt Flag for Slave Only** This bit indicates that the internal slave select signal has changed to active. It is cleared by software writes one to this bit 0 = The slave select signal has not changed to active. 1 = The slave select signal has changed to active.

		Note: The internal slave select signal is active high.
[8]	**SSINAIF**	**Slave Select Inactive Interrupt Flag for Slave Only** This bit indicates that the internal slave select signal has changed to inactive. It is cleared by software writes 1 to this bit 0 = The slave select signal has not changed to inactive. 1 = The slave select signal has changed to inactive. **Note:** The internal slave select signal is active high.
[7]	**Reserved**	Reserved.
[6]	**SLVBEIF**	**Slave Bit Count Error Interrupt Flag for Slave Only** 0 = Slave bit count error event did not occur. 1 = Slave bit count error event occurred. **Note:** It is cleared by software write 1 to this bit. If the transmit/receive data bit count does not match the setting of DWIDTH (USPI_LINECTL[11:8]), bit count error event occurs.
[5]	**SLVTOIF**	**Slave Time-out Interrupt Flag for Slave Only** 0 = Slave time-out event did not occur. 1 = Slave time-out event occurred. **Note:** It is cleared by software write 1 to this bit
[4]	**RXENDIF**	**Receive End Interrupt Flag** 0 = Receive end event did not occur. 1 = Receive end event occurred. **Note:** It is cleared by software write 1 to this bit. The receive end event happens when hardware receives the last bit of RX data into shift data unit.
[3]	**RXSTIF**	**Receive Start Interrupt Flag** 0 = Receive start event did not occur. 1 = Receive start event occurred. **Note:** It is cleared by software write 1 to this bit. For SPI master mode, the receive start event happens when SPI master sends slave select active and spi clock to the external SPI slave. For SPI slave mode, the receive start event happens when slave select of SPI slave is active and spi clock of SPI slave is inputed from the external SPI master.
[2]	**TXENDIF**	**Transmit End Interrupt Flag** 0 = Transmit end event did not occur. 1 = Transmit end event occurred. **Note:** It is cleared by software write 1 to this bit. The transmit end event happens when hardware sends the last bit of TX data from shift data unit.
[1]	**TXSTIF**	**Transmit Start Interrupt Flag** 0 = Transmit start event did not occur. 1 = Transmit start event occurred. **Note:** It is cleared by software write 1 to this bit. The transmit start event happens when hardware starts to move TX data from data buffer to shift data unit.
[0]	**Reserved**	Reserved.

6.22 USCI - I²C Mode

6.22.1 Overview

On I²C bus, data is transferred between a Master and a Slave. Data bits transfer on the SCL and SDA lines are synchronously on a byte-by-byte basis. Each data byte is 8-bit. There is one SCL clock pulse for each data bit with the MSB being transmitted first, and an acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP). Please refer to Figure 6.22-1 for more detailed I²C BUS Timing.

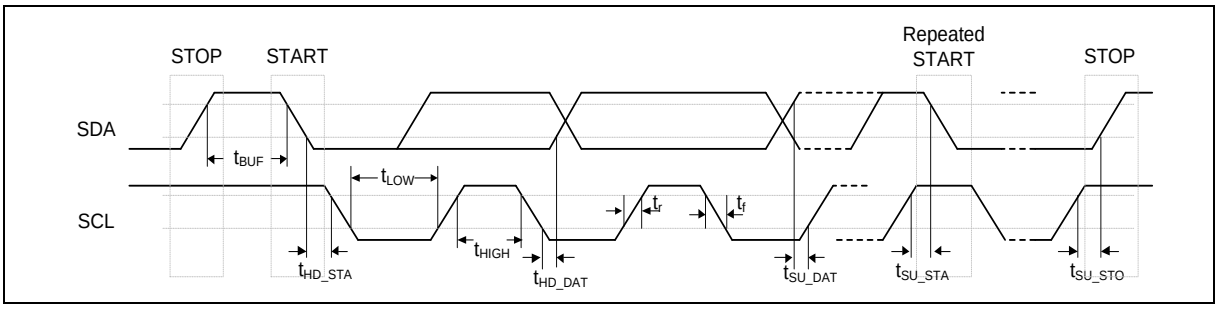


Figure 6.22-1 I²C Bus Timing

The device's on-chip I²C provides the serial interface that meets the I²C bus standard mode specification. The I²C port handles byte transfers autonomously. The I²C mode is selected by FUNMODE (UI2C_CTL [2:0]) = 100B. When enable this port, the USCI interfaces to the I²C bus via two pins: SDA and SCL. When I/O pins are used as I²C ports, user must set the pins function to I²C in advance.

Note: Pull-up resistor is needed for I²C operation because the SDA and SCL are set to open-drain pins when USCI is selected to I²C operation mode.

6.22.2 Features

- Full master and slave device capability
- Supports of 7-bit addressing, as well as 10-bit addressing
- Communication in standard mode (100 kbit/s) or in fast mode (up to 400 kbit/s)
- Supports multi-master bus
- Supports one transmit buffer and two receive buffer for data payload
- Supports 10-bit bus time-out capability
- Supports bus monitor mode.
- Supports Power down wake-up by START signal or address match
- Supports setup/hold time programmable
- Supports multiple address recognition (two slave address with mask option)

6.22.3 Block Diagram

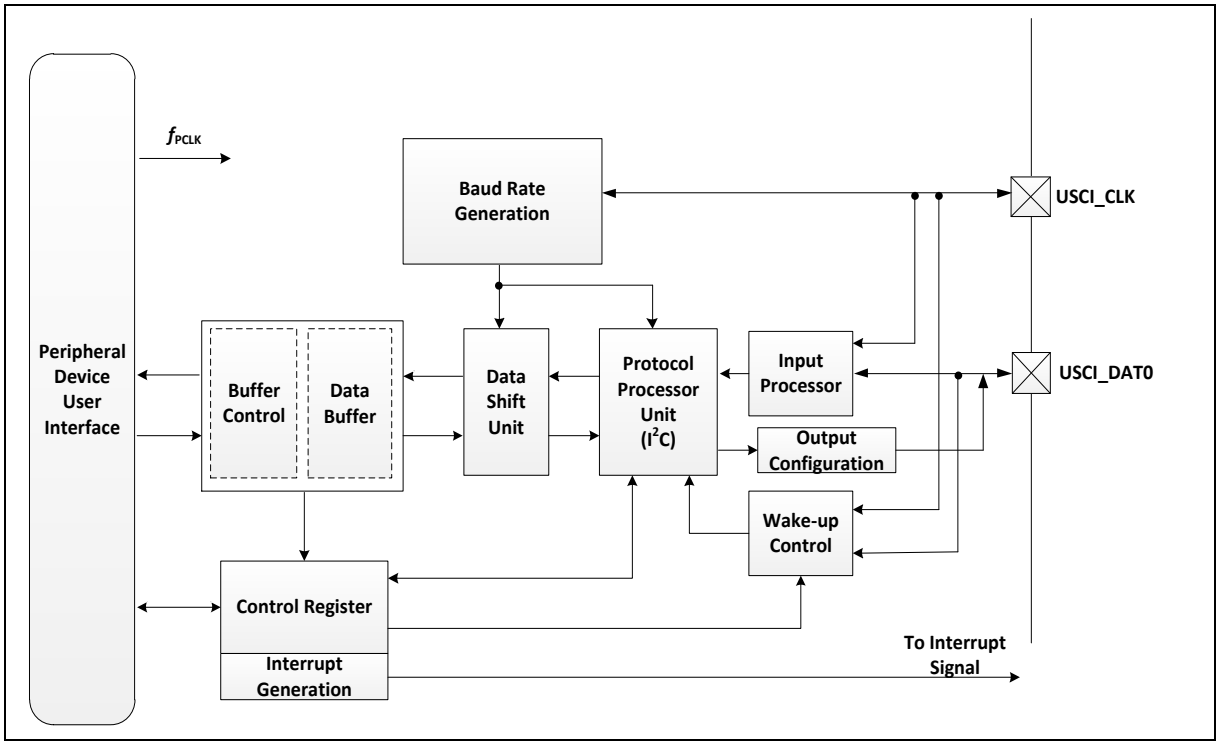


Figure 6.22-2 USCI I²C Mode Block Diagram

6.22.4 Basic Configuration

6.22.4.1 USCI0 I²C Basic Configuration

- Clock Source Configuration
  - Enable USCI0 peripheral clock in USCI0CKEN (CLK_APBCLK1[8]).
  - Enable USCI0_I2C function UI2C_CTL[2:0]) register, UI2C_CTL[2:0]=3'b100
- Reset Configuration
  - Reset USCI0 controller in USCI0RST (SYS_IPRST2[8]).

6.22.5 Functional Description

6.22.5.1 START or Repeated START Signal

Figure 6.22-3 shows the typical I²C protocol. Normally, a standard communication consists of four parts:

- START or Repeated START signal generation
- Slave address and R/W bit transfer
- Data transfer
- STOP signal generation

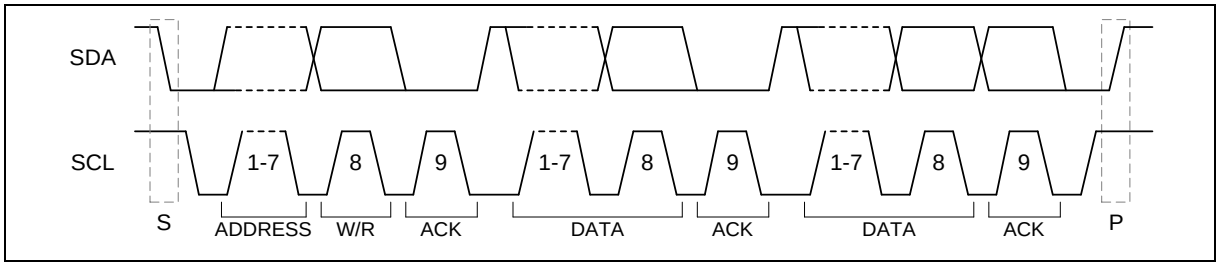


Figure 6.22-3 I²C Protocol

When the bus is free/idle, meaning no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the “S” bit, is defined as a HIGH to LOW transition on the SDA line while SCL is HIGH. The START signal denotes the beginning of a new data transmission.

A Repeated START is not a STOP signal between two START signals and usually referred to as the “Sr” bit. The master uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus idle flag.

6.22.5.2 STOP Signal

The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the “P” bit, is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH. The section between STOP and START is called bus free.

Figure 6.22-4 shows the waveform of START, Repeat START and STOP.

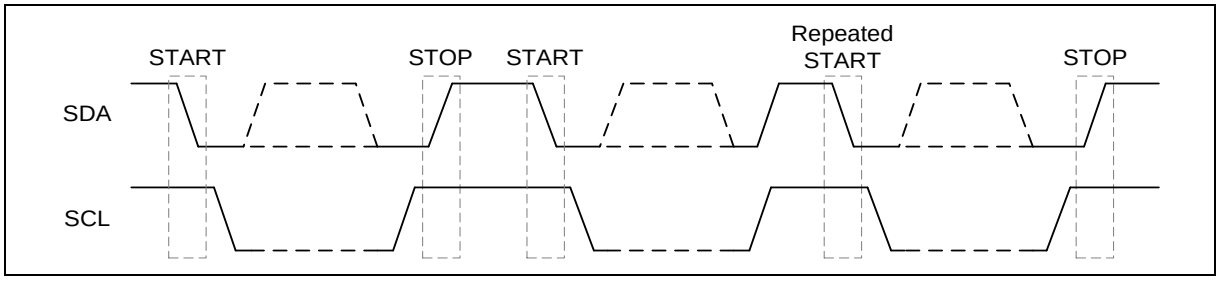


Figure 6.22-4 START and STOP Conditions

6.22.5.3 Slave Address Transfer

After a (Repeated) START condition, the master sends a slave address to identify the target device of the communication. The start address can comprise one or two address bytes (for 7-bit or for 10-bit addressing schemes). After an address byte, a slave sensitive to the transmitted address has to acknowledge the reception.

Therefore, the slave’s address can be programmed in the device, where it is compared to the received address. In case of a match, the slave answers with an acknowledge (SDA = 0). Slaves that are not targeted answer with a non-acknowledge (SDA = 1). In addition to the match of the programmed address, another address byte value has to be answered with an acknowledge if the slave is capable to handle the corresponding requests. The address byte 00H indicates a general call address that can be acknowledged.

In order to allow selective acknowledges for the different values of the address byte(s), the following control mechanism is implemented:

- If the GCFUNC bit (UI2C_PROTCTL [0]) is set the I²C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.
- The I²C port is equipped with one device address registers, UI2C_DEVADDRn (n = 0~1).

In 7-bit address mode, the first 7 bits of a received first address byte are compared to the programmed slave address (UI2C_DEVADDRn [6:0]). If these bits match, the slave sends an acknowledge.

- For 10 bit addressing mode, if the slave address is programmed to 1111 0XXB, the XX bits are compared to the bits UI2C_DEVADDR [9:8] to check for address match and also sends an acknowledge when ADDR10EN (UI2C_PROTCTL [4]) is set. The slave waits for a second address byte compares it with UI2C_DEVADDR [7:0] and sends an acknowledge accordingly to cover the 10 bit addressing mode. The user has to take care about reserved addresses (refer to I²C specification for more detailed description). Only the address 1111 0XXB is supported. Under each of these conditions, bit SLASEL (UI2C_PROTSTS [14]) will be set when the addressing delivered a match. This SLASEL (UI2C_PROTSTS [14]) bit is cleared automatically by a (Repeated) START or STOP condition.
- The I²C port is equipped multiple address recognition with one address mask registers I2C_ADDRMSKn (n = 0~1). When the bit in the address mask register is set to 1, it means the received corresponding address bit is "Don't care". If the bit is set to 0, it means the received corresponding register bit should be exactly the same as address register.

6.22.5.4 Data Transfer

When a slave receives a correct address with an R/W bit, the data will follow R/W bit specified to transfer. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a Not Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

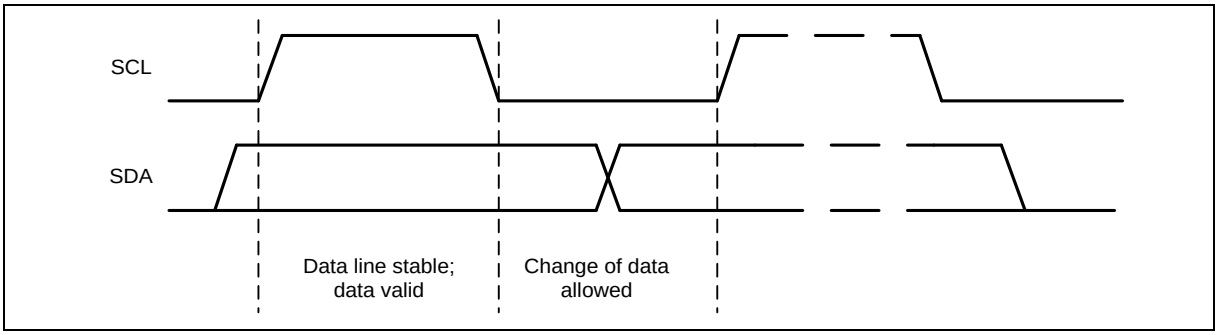


Figure 6.22-5 Bit Transfer on the I²C Bus

If the master received data, does Not Acknowledge (NACK) the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal.

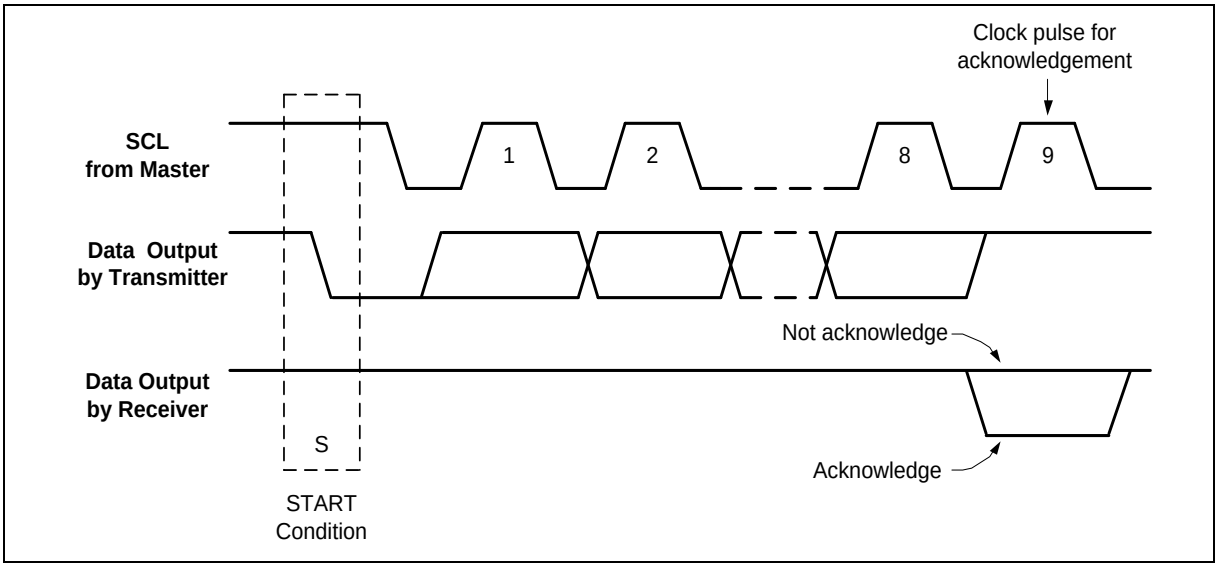


Figure 6.22-6 Acknowledge on the I²C Bus

6.22.5.5 Clock Baud Rate Bits

The data baud rate of I²C is determined by UI2C_BRGEN register when I²C is in Master Mode, and it is not necessary in a Slave mode. In the Slave mode, I²C will automatically synchronize it with any clock frequency from master I²C device. The bits RCLKSEL, SPCLKSEL, PDSCNT, and DSCNT define the baud rate setting:

- RCLKSEL (UI2C_BRGEN [0])

  to define the input frequency f_{REF_CLK}

- SPCLKSEL (UI2C_BRGEN [3:2])

  to define the multiple source of the sample clock f_{SAMP_CLK}

- PDSCNT (UI2C_BRGEN [9:8])

  to define the length of a data sample time (division of f_{REF_CLK} by 1, 2, 3, or 4)

- DSCNT (UI2C_BRGEN [14:10])

  to define the number of data sample time per bit time

  The standard setting is given by RCLKSEL = 0 ($f_{REF_CLK} = f_{PCLK}$), PTCLKSEL = 0 ($f_{PROT_CLK} = f_{REF_CLK}$) and SPCLKSEL = 2'b00 ($f_{SAMP_CLK} = f_{DIV_CLK}$). Under these conditions, the baud rate is given by:

$$f_{I2C} = f_{REF_CLK} \times \frac{1}{\text{CLKDIV} + 1} \times \frac{1}{\text{PDSCNT} + 1} \times \frac{1}{\text{DSCNT} + 1}$$

  In order to generate slower frequencies, additional divide-by-2 stages can be selected by PTCLKSEL = 1 ($f_{PROT_CLK} = f_{REF_CLK2}$), leading to:

$$f_{I2C} = \frac{f_{REF_CLK}}{2} \times \frac{1}{\text{CLKDIV} + 1} \times \frac{1}{\text{PDSCNT} + 1} \times \frac{1}{\text{DSCNT} + 1}$$

  If SPCLKSEL = 2'b10 ($f_{SAMP_CLK} = f_{SCLK}$), and RCLKSEL = 0 ($f_{REF_CLK} = f_{PCLK}$), PTCLKSEL = 0 ($f_{PROT_CLK} = f_{REF_CLK}$). The baud rate is given by:

$$f_{I2C} = f_{REF_CLK} \times \frac{1}{\text{CLKDIV} + 1} \times \frac{1}{2} \times \frac{1}{\text{PDSCNT} + 1} \times \frac{1}{\text{DSCNT} + 1}$$

6.22.5.6 Byte Stretching

If a device is selected as master/slave transmit mode and should transmit a data byte but the transmit buffer TXDAT does not contain valid data to be transmitted, the device ties down SCL = 0 at the end of the previous acknowledge bit. The waiting period is finished if software writes 1 to PTRG (UI2C_PROTCTL [5]).

6.22.5.7 Multi-master Arbitration

In some applications, there are two or more masters on the same I²C bus to access slaves, and the masters may transmit data simultaneously. The I²C supports multi-master by including collision detection and arbitration to prevent data corruption.

If two masters sometimes initiate I²C command at the same time, the arbitration procedure determines which master wins and can continue with the command. Arbitration is performed on the SDA signal while the SCL signal is high. Each master checks if the SDA signal on the bus corresponds to the generated SDA signal. If the SDA signal on the bus is low but it should be high, then this master has lost arbitration. Master I²C device that has lost arbitration can generate SCL pulses until the byte ends and must then release the bus and go into slave mode. The arbitration procedure can continue until all the data is transferred. This means that in multi-master system each I²C master must monitor the I²C bus for collisions and act accordingly. Figure 6.22-7 describes master1 data and master2 data are compete arbitration.

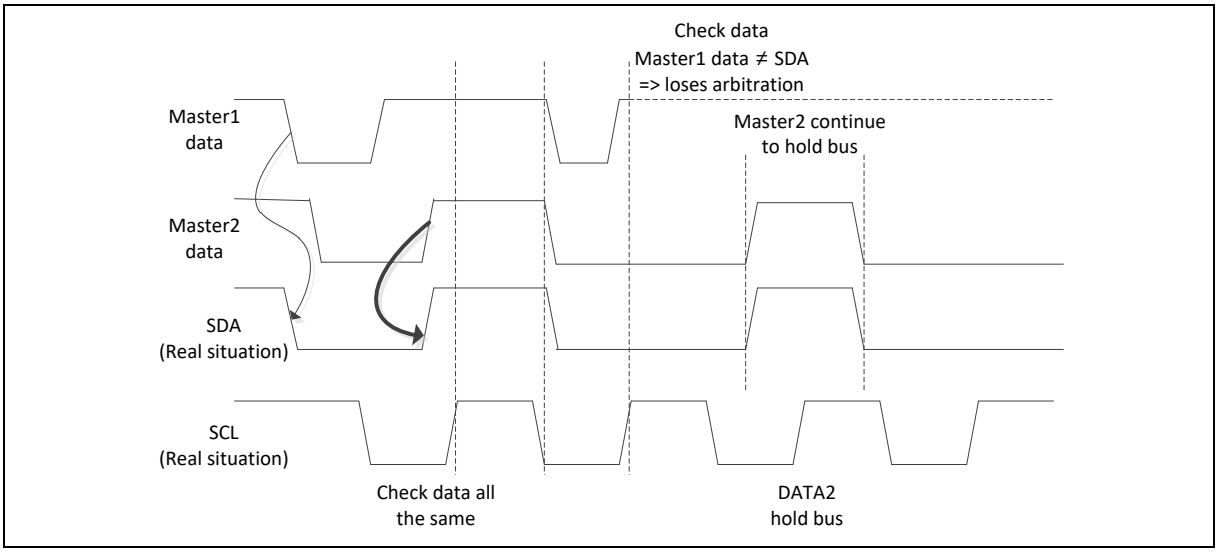


Figure 6.22-7 Arbitration Lost

In this case, during the address and data transmission, the master transmitter checks at the rising edge of SCL for each data bit if the value it is sending is equal to the value read on the SDA line. If yes, master can hold bus continuously. If this is not the case (transmitted value = 1, value read = 0), the master has lost the transmit arbitration. This is indicated by interrupt flag ARBLOIF (UI2C_PROTSTS [11]) and can generate a protocol interrupt if enabled by ARBLOIEN (UI2C_PROTIEN [4]).

When the transmit arbitration has been lost, the software has to initialize the complete frame again, starting with the first address byte together with the START condition for a new master transmit attempt. Arbitration also takes place for the ACK bit. If master arbitration lost and match the device address, then master will turn to slave.

6.22.5.8 Transmission Chain

The I²C bus protocol requiring a kind of in-bit-response during the arbitration phase and while a slave is transmitting, the resulting loop delay of the transmission chain can limit the reachable maximal baud rate, strongly depending on the bus characteristics (bus load, module frequency, etc.).

The shift clock SCL is generated by the master device, output on the wire, then it passes through the

input stage and the input filter. Now, the edges can be detected and the SDA data signal can be generated accordingly. The SDA signal passes through the output stage and the wire to the master receiver part. There, it passes through the input stage and the input filter before it is sampled.

This complete loop has to be finished (including all settling times to obtain stable signal levels) before the SCL signal changes again. The delays in this path have to be taken into account for the calculation of the baud rate as a function of f_{PCLK} and f_{PROT_CLK}. We suggest user adopt f_{PCLK}.

6.22.5.9 Non-Acknowledge and Error Conditions

In case of a non-acknowledge (NACKIF (UI2C_PROTSTS [10])) or an error (ERRIF(UI2C_PROTSTS [12])), no further transmission will take place. User software doesn't invalidate the transmit buffer and disable transmissions, before configuring the transmission (by writing TXDAT) again with appropriate values to react on the previous event.

6.22.5.10 I²C Protocol Interrupt Events

The following protocol-related events are generated in I²C mode and can lead to a protocol interrupt.

Please note that the bits in register UI2C_PROTSTS are not all automatically cleared by hardware and have to be cleared by software in order to monitor new incoming events.

- START condition received at a correct position in a frame (STARIF (UI2C_PROTSTS [8]))
- STOP condition transferred at a correct position in a frame (STORIF (UI2C_PROTSTS [9]))
- Master arbitration lost (ARBLOIF (UI2C_PROTSTS [11]))
- Slave read requested (SLAREAD (UI2C_PROTSTS [15]))
- Acknowledge received (ACKIF (UI2C_PROTSTS [13]))
- Non-acknowledge received (NACKIF (UI2C_PROTSTS [10]))
- START condition not at the expected position in a frame (ERRIF (UI2C_PROTSTS [12]))
- STOP condition not at the expected position in a frame (ERRIF (UI2C_PROTSTS [12]))

6.22.5.11 Operating the I²C

To operate the I²C protocol, the following issues have to be considered:

Select I²C Mode

It is recommended to configure all parameters of the I²C that do not change during run time while FUNMODE (UI2C_CTL [2:0]) = 000B. The I²C control flow has to be done while FUNMODE (UI2C_CTL [2:0]) = 000B to avoid unintended edges of the input signals and the I²C mode can be enabled by FUNMODE (UI2C_CTL [2:0]) = 100B afterwards.

Step 1. Set FUNMODE (UI2C_CTL [2:0]) = 000B

Step 2. Set FUNMODE (UI2C_CTL [2:0]) = 100B

Pin Connections

The pins used for SDA and SCL have to be set to open-drain mode by USCI controller to support the wired-AND structure of the I²C bus lines.

Note: The step to enable the alternate output port functions should only be done after the I²C mode is enabled, to avoided unintended spikes on the output.

Bit Timing Configuration

In standard mode (100 kbit/s) a minimum module frequency of 2 MHz is necessary, whereas in fast mode (400 kbit/s) a minimum of 10 MHz is required. Additionally, if the digital filter stage should be used to eliminate spikes up to 50 ns, a filter frequency of 20 MHz is necessary. There could be an uncertainty

in the SCL high phase timing of maximum $1/f_{PROT_CLK}$ if another I²C participant lengthens the SCL low phase on the bus. Note that the SCL maximum frequency is $f_{SAMP_CLK}/2$ and the SPCLKSEL (UI2C_BRGEN [3:2]) must be set to 0 for selecting $f_{SAMP_CLK} = f_{DIV_CLK}$.

Data Format Configuration

The data format has to be configured for 8 data bits (DWIDTH (UI2C_LINECTL [11:8]) = 8), and MSB shifted first (LSB (UI2C_LINECTL [0]) = 0). As a result, UI2C_LINECTL has to be set to 0x800.

Control Flow

The on-chip I²C ports support three operation modes, Master, Slave, and General Call Mode.

In a given application, I²C port may operate as a master or as a slave. In Slave mode, the I²C port hardware looks for its own slave address and the general call address. If one of these addresses is detected, and if the slave is willing to receive or transmit data from/to master (by setting the AA bit), acknowledge pulse will be transmitted out on the 9th clock, hence an interrupt is requested on both master and slave devices if interrupt is enabled. When the microcontroller wishes to become the bus master, hardware waits until the bus is free before entering Master mode so that a possible slave action is not be interrupted. If address arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer.

To control the I²C bus transfer in each mode, user needs to set UI2C_PROTCTL, UI2C_PROTIEN, TXDAT registers according to current status of UI2C_PROTSTS register. In other words, for each I²C bus action, user needs to check current status by UI2C_PROTSTS register, and then set UI2C_PROTCTL, UI2C_PROTIEN, TXDAT registers to take bus action. Finally, check the response status by UI2C_PROTSTS.

The bits, STA, STO and AA in UI2C_PROTCTL register are used to control the next state of the I²C hardware after interrupt signal is cleared. Upon completion of the new action, a new status will be updated in UI2C_PROTSTS register will be set. If the I²C interrupt control bit of UI2C_PROTIEN is set, appropriate action or software branch of the new status can be performed in the Interrupt service routine.

Figure 6.22-8 shows the current I²C STARIF (UI2C_PROTSTS [8]) is set to 1 by hardware, and then set TXDAT = SLA+W (Slave address + Write bit), (PTRG, STA, STO, AA) = (1, 0, 0, x) to send the address to I²C bus, and write 1 to STARIF (UI2C_PROTSTS [8]) to clear flag. If a slave on the bus matches the address and response ACK, the UI2C_PROTSTS will be updated by ACKIF (UI2C_PROTSTS [13]) setting.

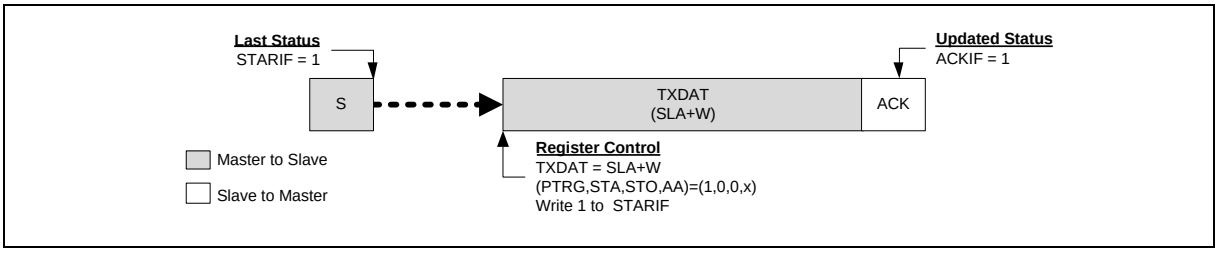


Figure 6.22-8 Control I²C Bus according to Current I²C Status

Data Transfer on the I²C Bus

Figure 6.22-9 shows a master transmits data to slave. A master addresses a slave with a 7-bit address and 1-bit write index to denote that the master wants to transmit data to the slave. The master keeps transmitting data after the slave returns acknowledge to the master.

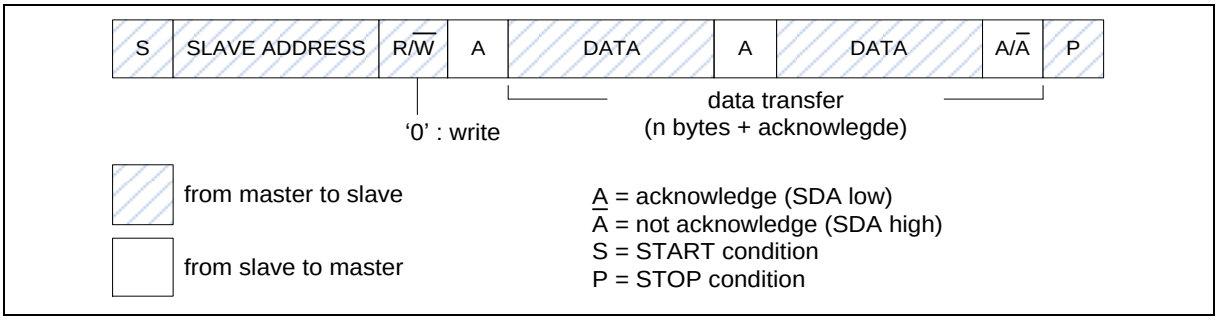


Figure 6.22-9 Master Transmits Data to Slave with a 7-bit Address

Figure 6.22-10 shows a master read data from slave. A master addresses a slave with a 7-bit address and 1-bit read index to denote that the master wants to read data from the slave. The slave will start transmitting data after the slave returns acknowledge to the master.

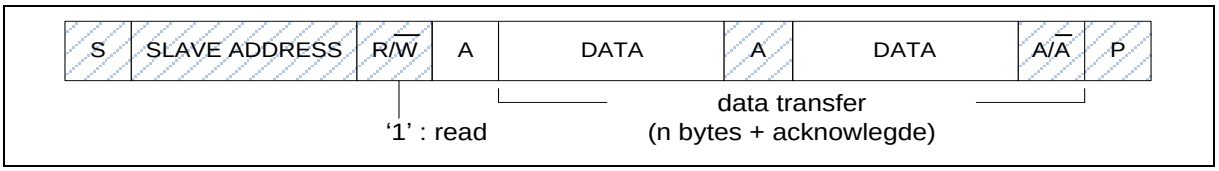


Figure 6.22-10 Master Reads Data from Slave with a 7-bit Address

Figure 6.22-11 shows a master transmits data to slave by 10-bit address. A master addresses a slave with a 10-bit address. First byte contains 10-bit address indicator (5'b11110) and 2-bit address with write index, second byte contains 8-bit address. The master keeps transmitting data after the second byte end. Note that 7-bit and 10-bit address device can work on the same bus.

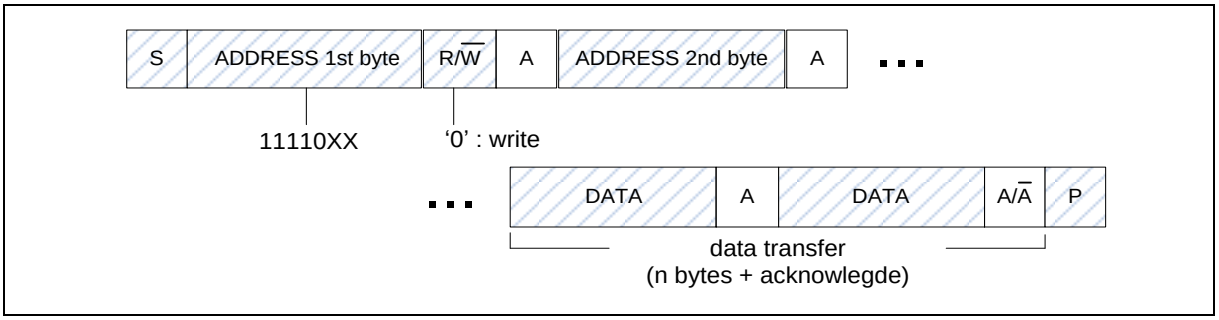


Figure 6.22-11 Master Transmits Data to Slave by 10-bit Address

Figure 6.22-12 shows a master read data from slave by 10-bit address. A master addresses a slave with a 10-bit address. First mater transmits 10-bit address to slave, after that master transmits first byte with read index. The slave will start transmitting data after the first byte with read index.

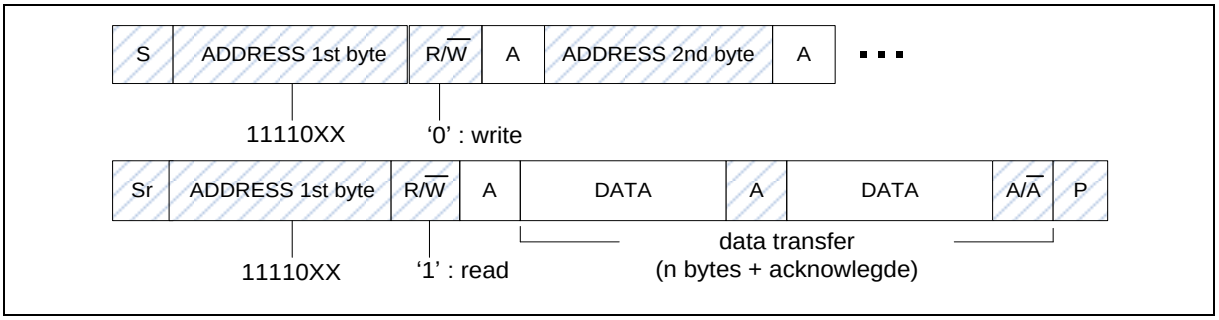


Figure 6.22-12 Master Reads Data from Slave by 10-bit Address

Master Mode

In Figure 6.22-13 and Figure 6.22-14, all possible protocols for I²C master are shown. User needs to follow proper path of the flow to implement required I²C protocol.

In other words, user can send a START signal to bus and I²C will be in Master Transmitter mode (Figure 6.22-13) or Master receiver mode (Figure 6.22-14) after START signal has been sent successfully and new status register would be set STARIF (UI2C_PROTSTS [8]). Followed by START signal, user can send slave address, read/write bit, data and Repeat START, STOP to perform I²C protocol.

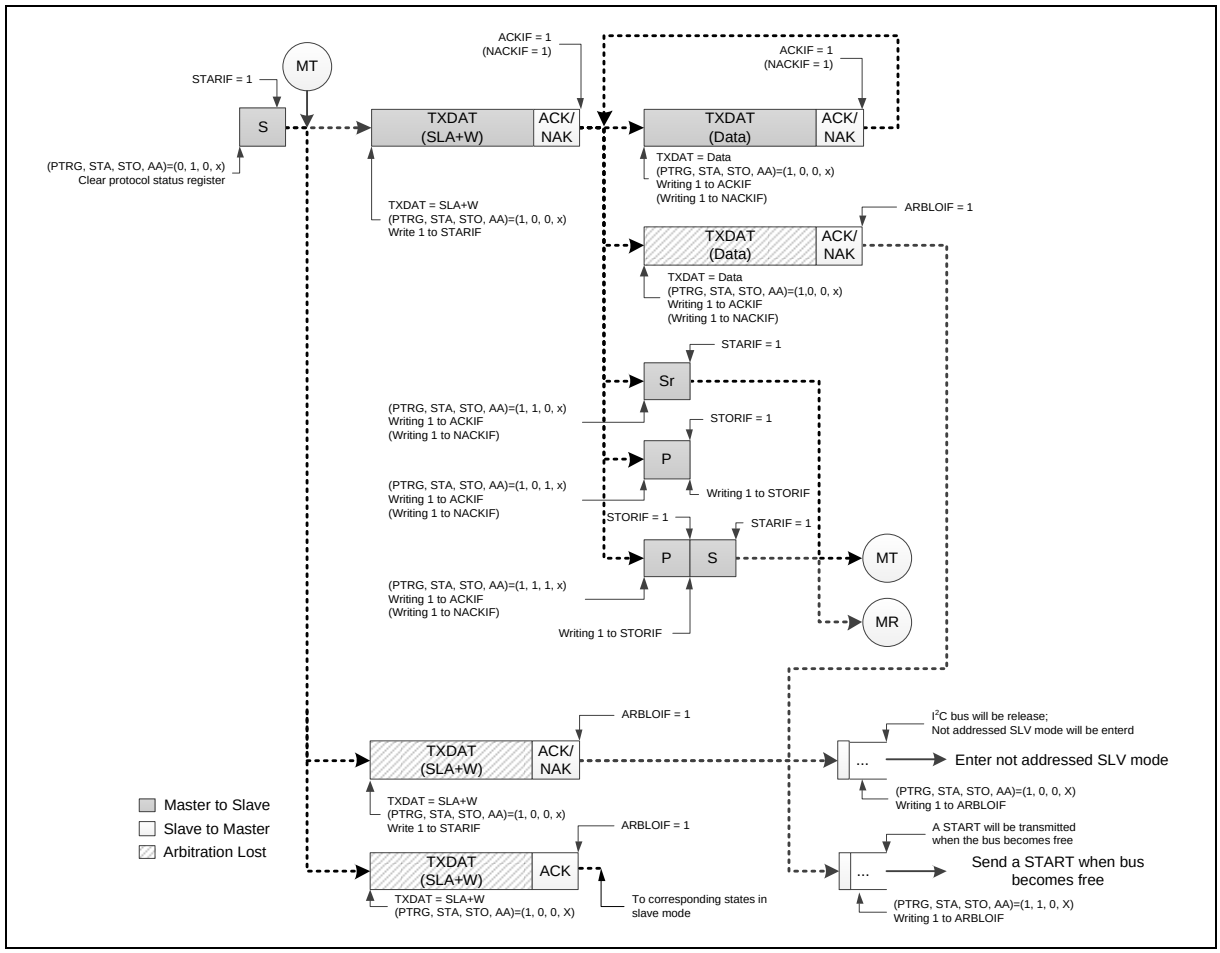


Figure 6.22-13 Master Transmitter Mode Control Flow with 7-bit Address

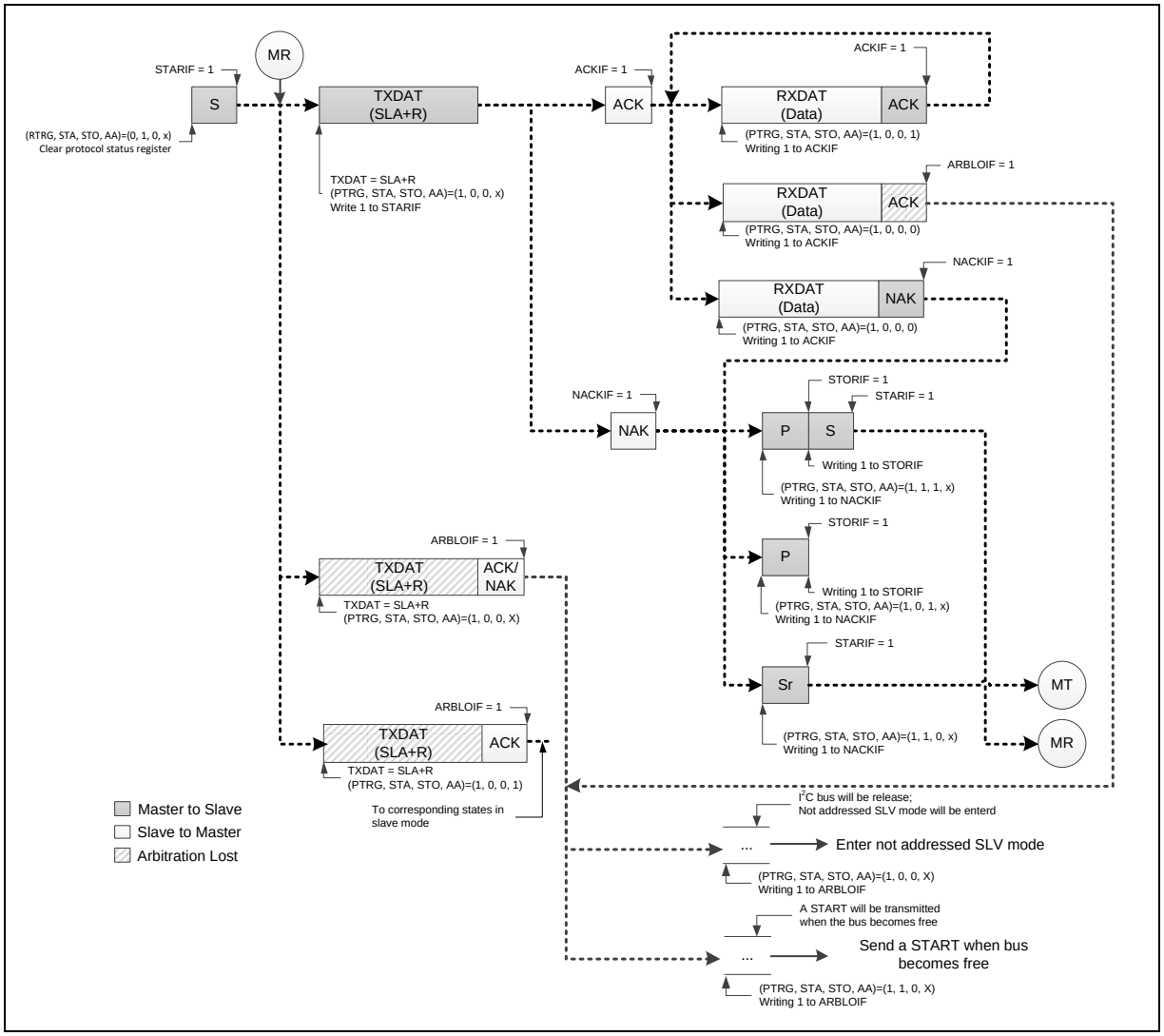


Figure 6.22-14 Master Receiver Mode Control Flow with 7-bit Address

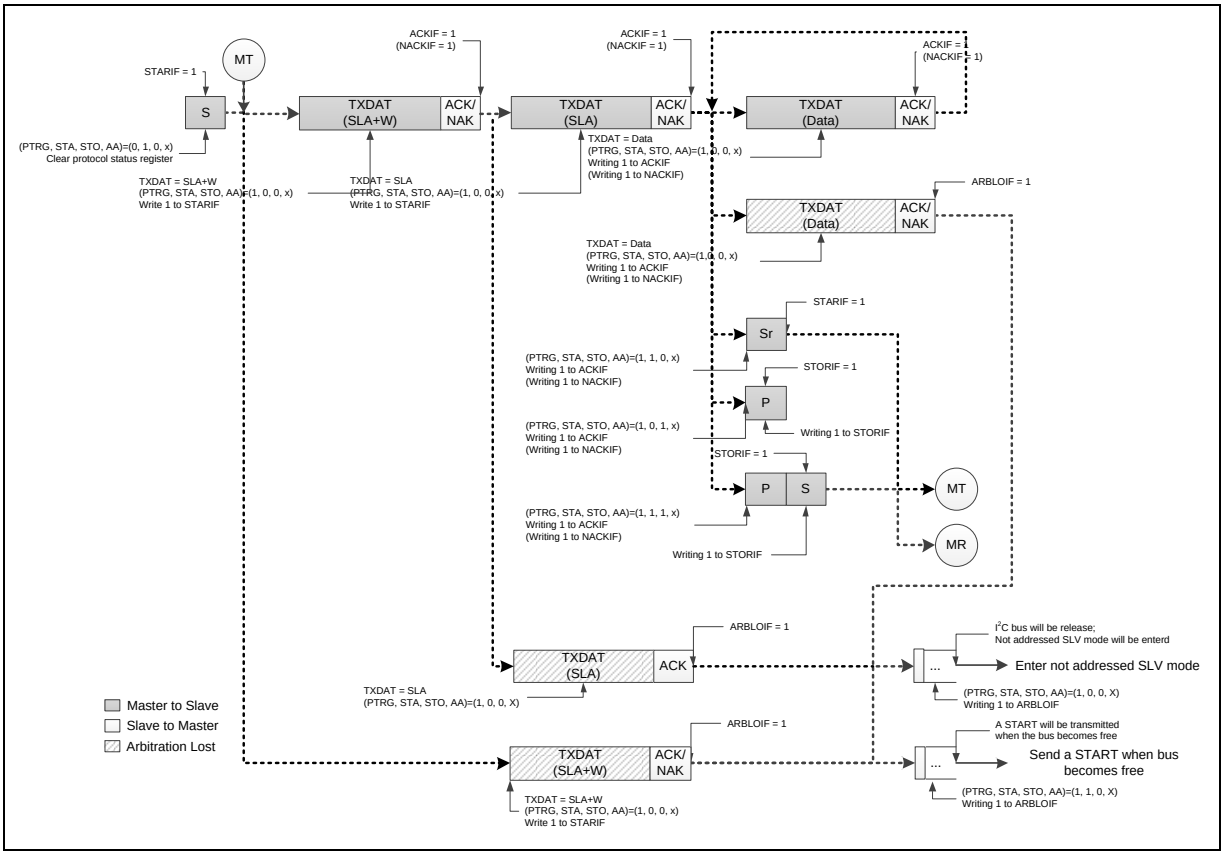


Figure 6.22-15 Master Transmitter Mode Control Flow with 10-bit Address

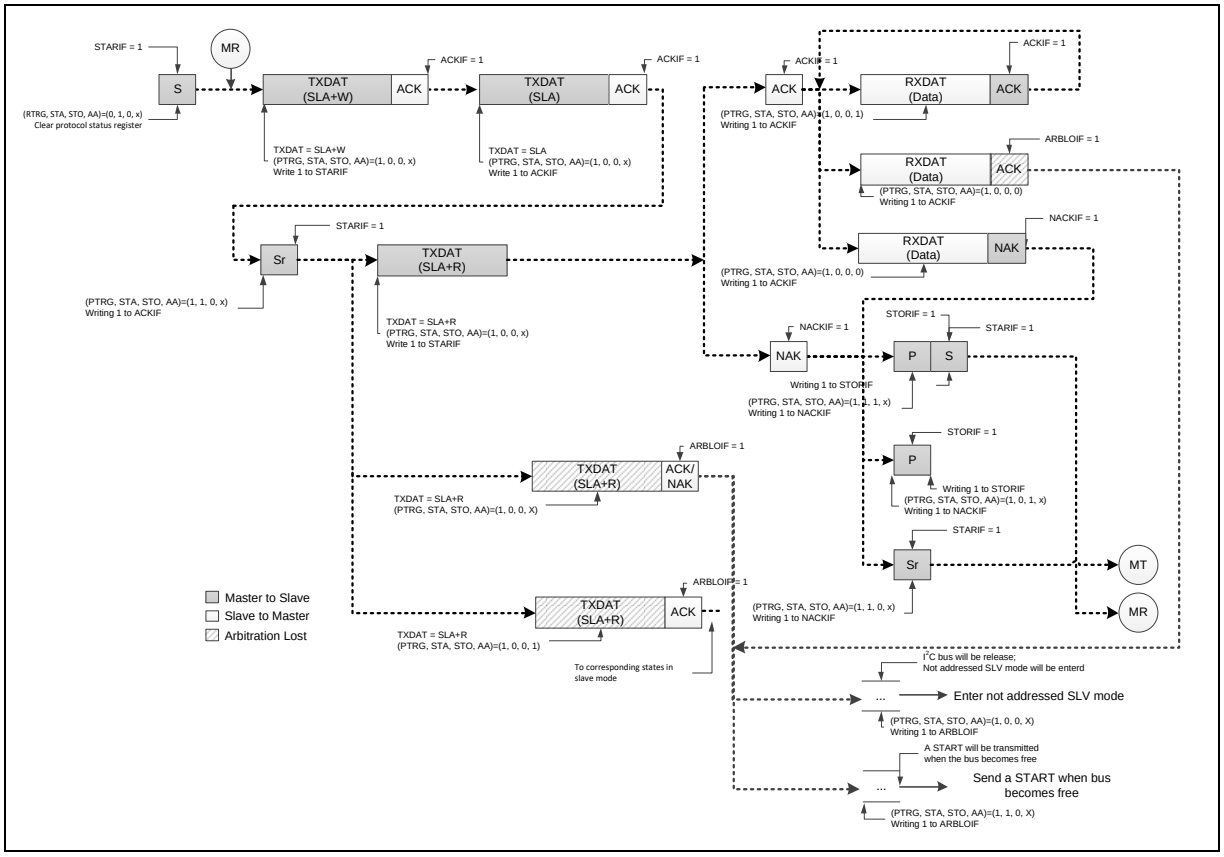


Figure 6.22-16 Master Receiver Mode Control Flow with 10-bit Address

If the I²C is in Master mode and gets arbitration lost, the bit of ARBLOIF (UI2C_PROTSTS [11]) will be set. User may write 1 to ARBLOIF (UI2C_PROTSTS [11]) and set (PTRG, STA, STO, AA) = (1, 1, 0, X) to send START to re-start Master operation when bus become free. Otherwise, user may write 1 to ARBLOIF (UI2C_PROTSTS [11]) and set (PTRG, STA, STO, AA) = (1, 0, 0, X) to release I²C bus and enter not addressed Slave mode.

Slave Mode

When reset, I²C is not addressed and will not recognize the address on I²C bus. User can set device address by UI2C_DEVADDRn and set (PTRG, STA, STO, AA) = (1, 0, 0, 1) to let I²C recognize the address sent by master. Figure 6.22-17 shows all the possible flow for I²C in Slave mode. Users need to follow a proper flow (as shown in Figure 6.22-17 to implement their own I²C protocol.

If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer. If the detected address is SLA+W (Master want to write data to Slave) or SLA+R (Master want to read data from Slave) after arbitration lost, the ARBLOIF will be set to 1.

Note: During I²C communication, the SCL clock will be released when writing '1' to PTRG (UI2C_PROTCTL [5]) in Slave mode.

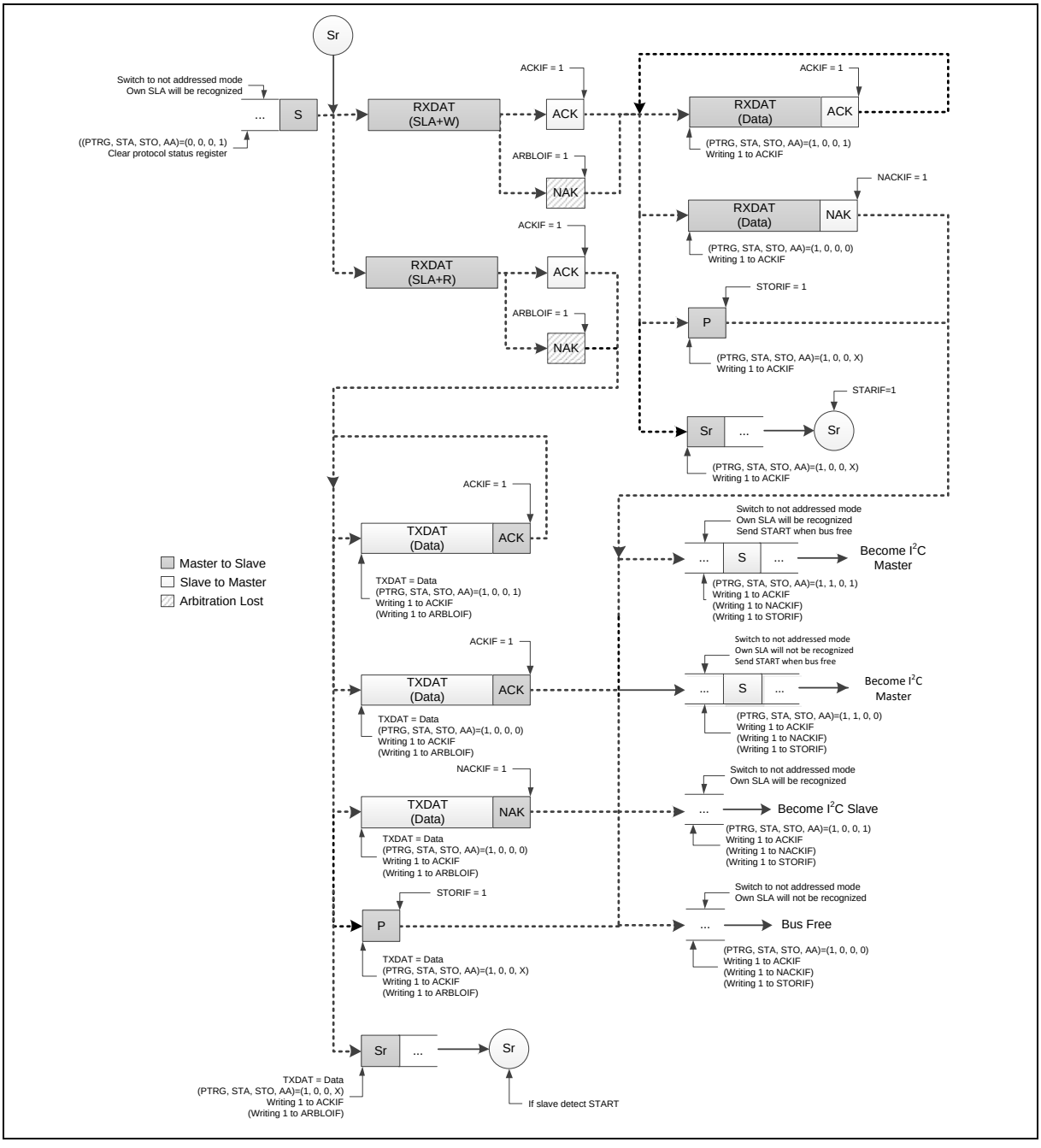


Figure 6.22-17 Save Mode Control Flow with 7-bit Address

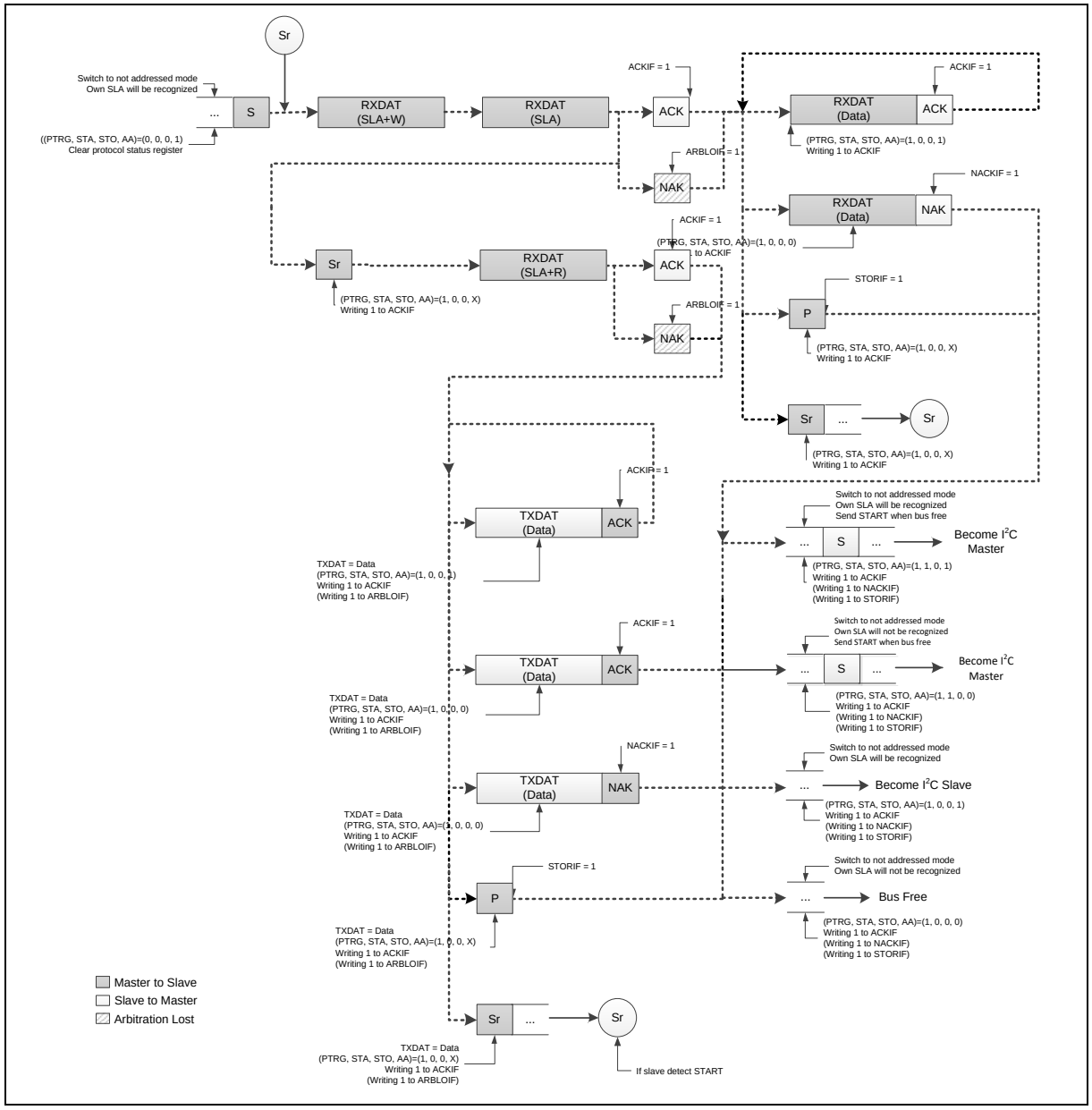


Figure 6.22-18 Save Mode Control Flow with 10-bit Address

If I²C is still transmitting and receiving data in addressed Slave mode but got a STOP or Repeat START, the register STORIF (UI2C_PROTSTS [9]) or STARIF (UI2C_PROTSTS [8]) will be set. User could follow the action for NACKIF (UI2C_PROTSTS [10]) as shown in the above figure when got STARIF (UI2C_PROTSTS [8]) is set.

Note: After slave gets interrupt flag of NACKIF (UI2C_PROTSTS [10]) and start/stop symbol including STARIF (UI2C_PROTSTS [8]) and STORIF (UI2C_PROTSTS [9]), slave can switch to not address mode and own SLA will not be recognized. If setting this interrupt flag, slave will not receive any I²C signal or address from master. At this status, I²C should be reset by setting FUNMODE (UI2C_CTL [2:0]) = 000B to leave this status.

General Call (GC) Mode

If the GCFUNC bit (UI2C_PROTCTL [0]) is set, the I²C port hardware will respond to General Call address (00H). User can clear GC bit to disable general call function. When the GC bit is set and the

I²C in slave mode, it can receive the general call address by 0x00 after master send general call address to I²C bus, and then it also will follow protocol status register.

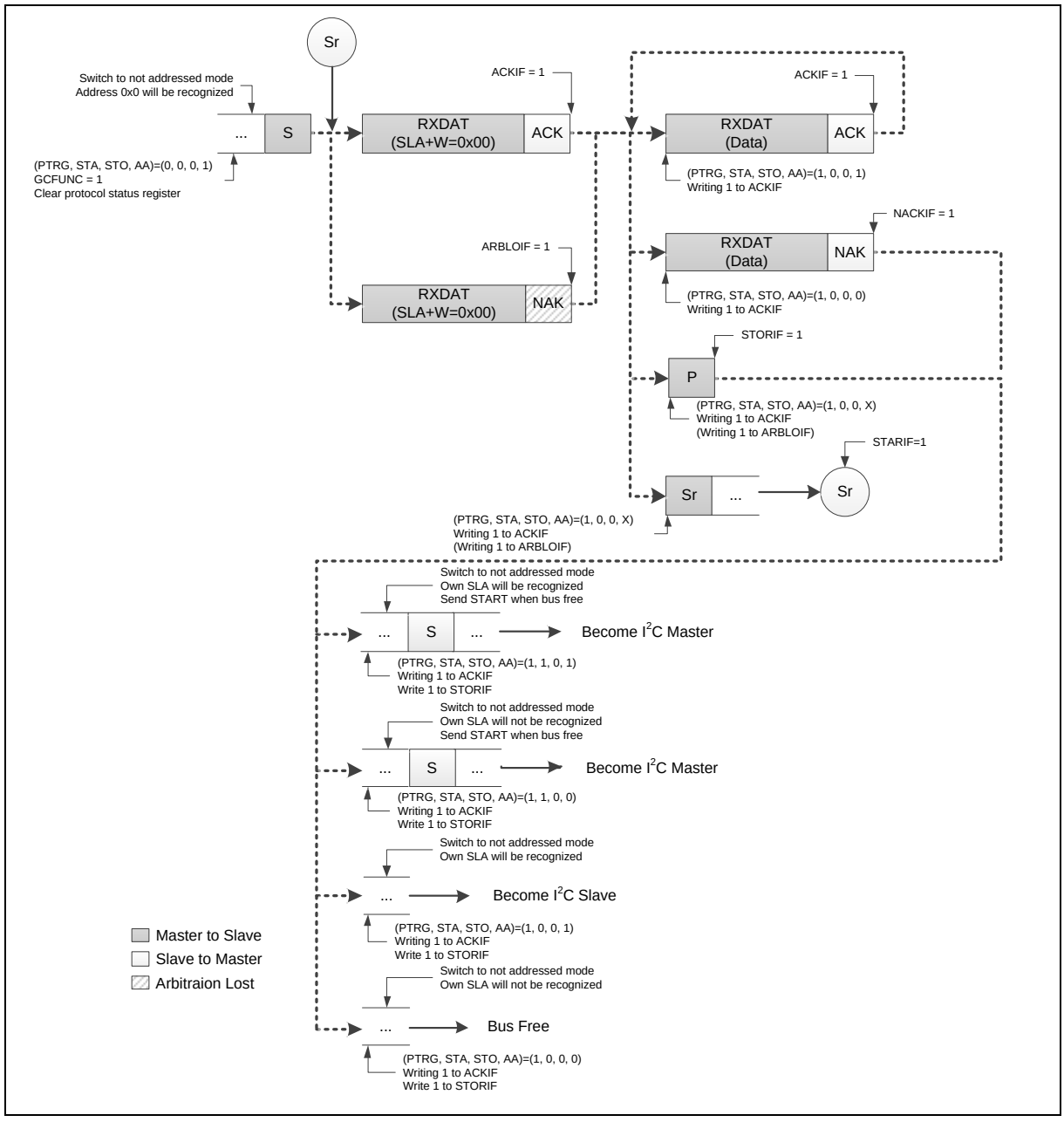


Figure 6.22-19 GC Mode with 7-bit Address

If I²C is still receiving data in GC mode but got a STOP or Repeat START, the STORIF (UI2C_PROTSTS [9]) or STARIF (UI2C_PROTSTS [8]) will be set. User could follow the action for NACKIF (UI2C_PROTSTS [10]) in above figure when got STORIF (UI2C_PROTSTS [9]) or STARIF (UI2C_PROTSTS [8]) is set.

Note: After slave gets interrupt flag of NACKIF (UI2C_PROTSTS [10]) and start/stop symbol including STARIF (UI2C_PROTSTS [8]) and STORIF (UI2C_PROTSTS [9]), slave can switch to not address mode and own SLA will not be recognized. If setting this interrupt flag, slave will not receive any I²C signal or address from master. At this time, the I²C controller should be reset by setting FUNMODE (UI2C_CTL [2:0]) = 000B to leave this status.

Protocol Functional Description

Monitor Mode

When I²C enters monitor mode, this device always returns NACK to master after each frame reception even address matching. Moreover, this device will store any receive data including address, command code, and data.

Interrupt in Monitor Mode

All interrupts will occur as normal process when the MONEN (UI2C_PROTCTL [9]) is set. Note that the first interrupt will occur when initial START, it not the same as I²C slave, but the other interrupts are the same.

Subsequent to the address-match detection, interrupts will be generated after each data byte is received as slave mode control flow, or after each byte that the module believes it has transmitted for a slave-read transfer. In this second case, the data register will actually contain data transmitted by some other slave on the bus which was actually addressed by the master. If user wants to watch other device, user can set address mask and monitor.

If the monitor has not had time to respond to interrupt, the SCL signal will be pulled to low when SCLOUTEN (UI2C_PROTCTL [8]) is set to 1. User must set PTRG (UI2C_PROTCTL [5]) to release bus when SCLOUTEN (UI2C_PROTCTL [8]) is set to 1. If SCLOUTEN (UI2C_PROTCTL [8]) is not set to 1, user doesn't need to set PTRG (UI2C_PROTCTL [5]) to 1.

When device address match, but the device response NACK, this address will be received into buffer and NACK interrupt will be generated.

Following all of these interrupts, the processor may read the data register to see what was actually transmitted on the bus.

Loss of Arbitration in Monitor Mode

In monitor mode, the I²C module will not be able to respond to a request for information by the bus master or issue an ACK. Some other slave on the bus will respond instead. Software should be aware of the fact that the module is in monitor mode and should not respond to any loss of arbitration state that is detected.

Programmable Setup and Hold Time

In order to guarantee a correct data setup and hold time, the timing must be configured. By programming HTCTL (UI2C_TMCTL [24:16]) to configure hold time and STCTL (UI2C_TMCTL[8:0]) to configure setup time.

The delay timing refer to peripheral clock (PCLK). When device stretch master clock, the setup and hold time configuration value will not be affected by stretched.

User should focus the limitation of setup and hold time configuration, the timing setting must follow I²C protocol. Once setup time configuration greater than design limitation, that means if setup time setting make SCL output less than three PCLKs, I²C controller can't work normally due to SCL must sample three times. And once hold time configuration greater than I²C clock limitation, I²C will occur bus error. Suggest that user calculate suitable timing with baud rate and protocol before setting timing. Table 6.22-1 shows the relationship between I²C baud rate and PCLK, the number of table represent one clock duty contain how many PCLKs. Setup and hold time configuration even can program some extreme values in the design, but user should follow I²C protocol standard.

I²C Baud Rate PCLK	100k	200k	400k	800k	1200k
12 MHz	120	60	30	15	10
24 MHz	240	120	60	30	20

48 MHz	480	240	120	60	40
72 MHz	720	360	180	90	60

Table 6.22-1 Relationship between I²C Baud Rate and PCLK

For setup time wrong adjustment example, assuming one SCL cycle contains ten PCLKs and set STCTL (UI2C_TMCTL [8:0]) to 3 that stretch three PCLKs for setup time setting. The setup time setting limitation: ST_{limit} = (UI2C_BRGEN[25:16]+1) - 6.

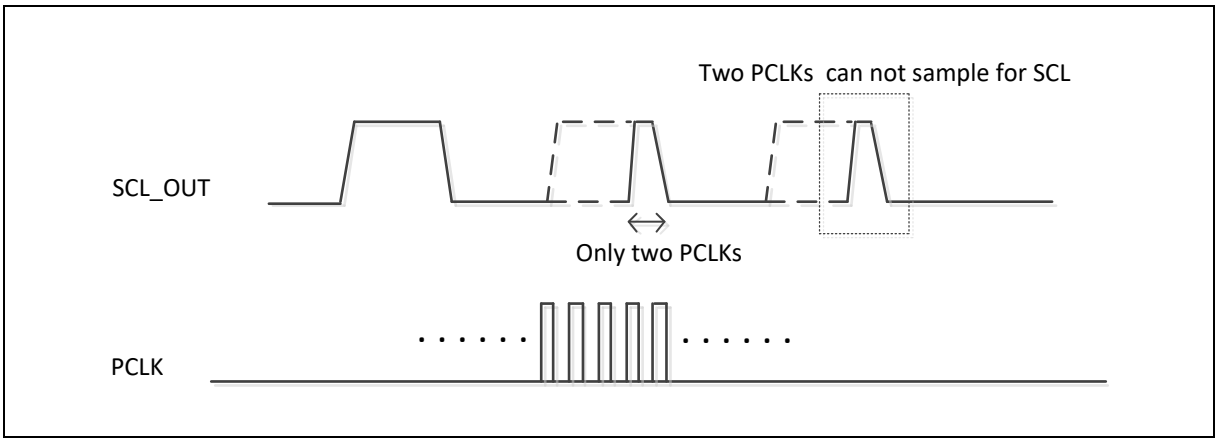


Figure 6.22-20 Setup Time Wrong Adjustment

For hold time wrong adjustment example, use I²C Baud Rate = 1200k and PCLK = 72 MHz, the SCL high/low duty = 60 PCLK. When HTCTL (UI2C_TMCTL [24:16]) is set to 63 and STCTL (UI2C_TMCTL[8:0]) is set to 0, then SDA output delay will over SCL high duty and cause bus error. The hold time setting limitation: HT_{limit} = (UI2C_BRGEN [25:16]+1) - 9.

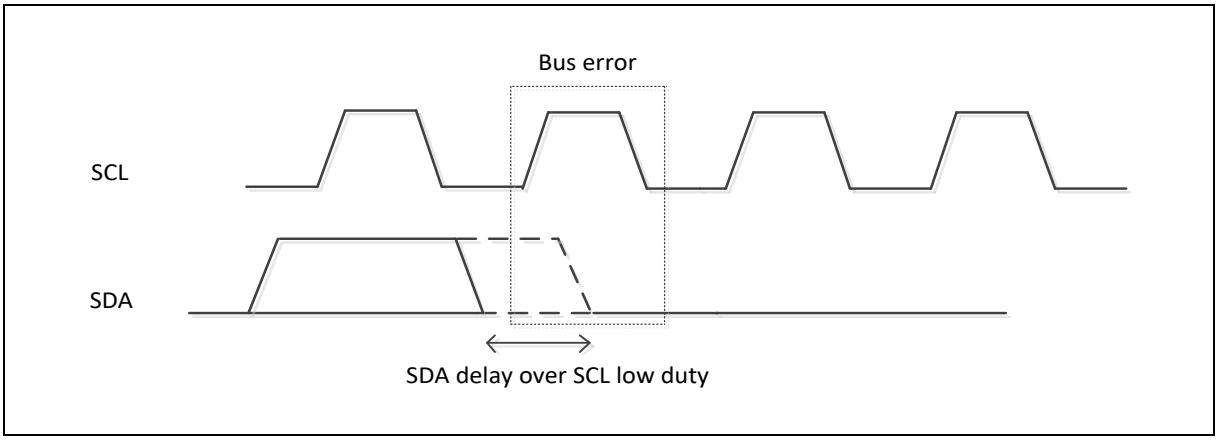


Figure 6.22-21 Hold Time Wrong Adjustment

I²C Time-out Function

There is a 10-bit time-out counter TOCNT (UI2C_PROTCTL [25:16]) which can be used to deal with the I²C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it equals TOCNT (UI2C_PROTCTL [25:16]) and generates I²C interrupt to CPU or stops counting by clearing TOIEN (UI2C_PROTIEN [0]) to 0 or setting all I²C interrupt signal (ACKIF, ERRIF, ARBLOIF, NACKIF, STORIF, STARIF). User may write 1 to clear TOIF(UI2C_PROTSTS [5]) to 0. When time-out counter is enabled, writing 1 to the TOIF will reset counter and re-start up counting after TOIF is cleared. Refer to Figure 6.22-22 for the time-out counter TOCNT (UI2C_PROTCTL [25:16]). T_{TOCNT} = (TOCNT (UI2C_PROTCTL

[25:16]) +1) x32 (5-bit) x T_{PCLK}. Note that the time counter clock source TMCNTSRC (UI2C_ BRGEN [5]) must be set as 0.

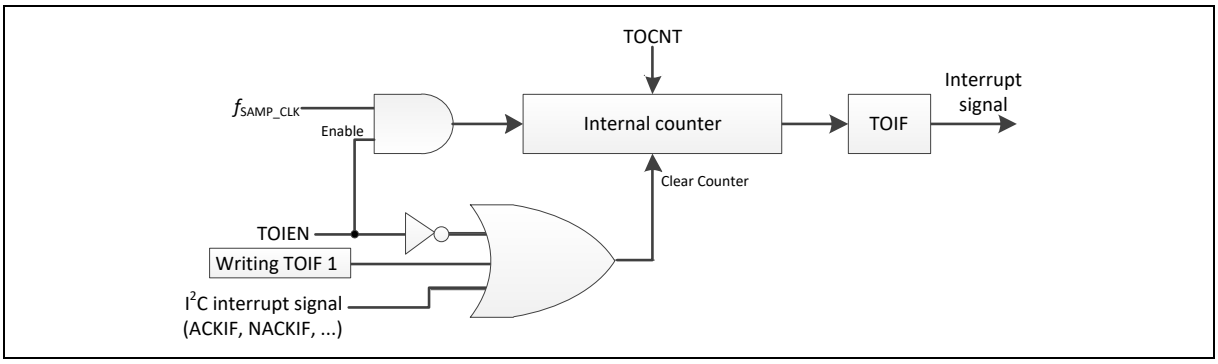


Figure 6.22-22 I²C Time-out Count Block Diagram

Wake-up Function

When chip enters Power-down mode and sets WKEN (WKCTL [0]) to 1, other I²C master can wake up the chip by addressing the I²C device. User must configure the related setting before entering sleep mode. The ACK bit cycle of address match frame is done in power-down. The controller will stretch the SCL to low when the address is matched the device's address and the ACK cycle done. The SCL is stretched until the bit is clear by user. If the frequency of SCL is low speed and the system has wakeup from address match frame, the user shall check this bit to confirm this frame has transaction done and then to do the wake-up procedure. Therefore, when the chip is woken up by address match with one of the device address register (UI2C_DEVADDRn), the user shall check the WKAKDONE (UI2C_PROTSTS [16]) bit is set to 1 to confirm if the address wakeup frame has been done. The WKAKDONE bit indicates that the ACK bit cycle of address match frame is done in power-down. The controller will stretch the SCL to low when the address is matched the device's slave address and the ACK cycle done. The SCL is stretched until the WKAKDONE bit is clear by user. If the frequency of SCL is low speed and the system has wakeup from address match frame, the user shall check this bit to confirm this frame has transaction done and then to do the wake-up procedure. Note that user must clear WKUPIF after clearing the WKAKDONE bit to 0.

The WRSTSWK (UI2C_PROTSTS [17]) bit records the Read/Write command on the address match wake-up frame. The user can use read this bit's status to prepare the next transmitted data (WRSTSWK = 0) or to wait the incoming data (WRSTSWK = 1) can be stored in time after the system is wake-up by the address match frame.

When system is woken up by other I²C master device, WKF (UI2C_WKSTS [0]) is set to indicate this event. User needs write "1" to clear this bit.

Example for Random Read on EEPROM

The following steps are used to configure the USCI0_I2C related registers when using I²C protocol to read data from EEPROM.

1. Set USCI0_I2C the multi-function pin as SCL and SDA pins. Refer to the Basic Configuration section for the multi-function configuration.
2. Enable USCI0 APB clock. Refer to the Basic Configuration section for the multi-function configuration.
3. Set USCI0RST=1 to reset USCI controller then set USCI0RST=0 let USCI controller to normal operation. Refer to the Basic Configuration section for the reset controller configuration.
4. Set FUNMODE = 3'b100 to enable USCI0_I2C controller in the "UI2C_CTL" register.
5. Give USCI0_I2C clock a divided register value for USCI0_I2C clock rate in the "UI2C_BRGEN".

6. Enable system I I2C0 IRQ in system “NVIC” control register.
7. Set ACKIEN, ERRIEN, ARBLOIEN, NACKIEN, STORIEN, STARIEN, and TOIEN to enable I²C Interrupt in the “UI2C_PROTIEN” register.
8. Set USCI address registers “UI2C_ADDR0 ~ UI2C_ADDR1”.

Random read operation is one of the methods of access EEPROM. The method allows the master to access any address of EEPROM space. Figure 6.22-23 shows the EEPROM random read operation.

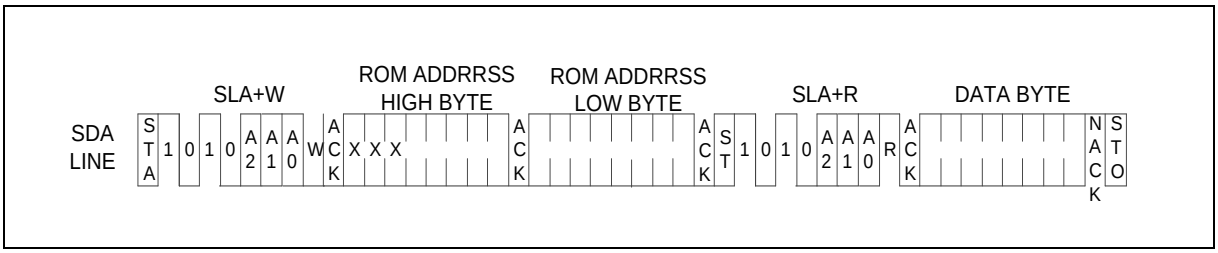


Figure 6.22-23 EEPROM Random Read

Figure 6.22-24 shows how to use I²C controller to implement the protocol of EEPROM random read.

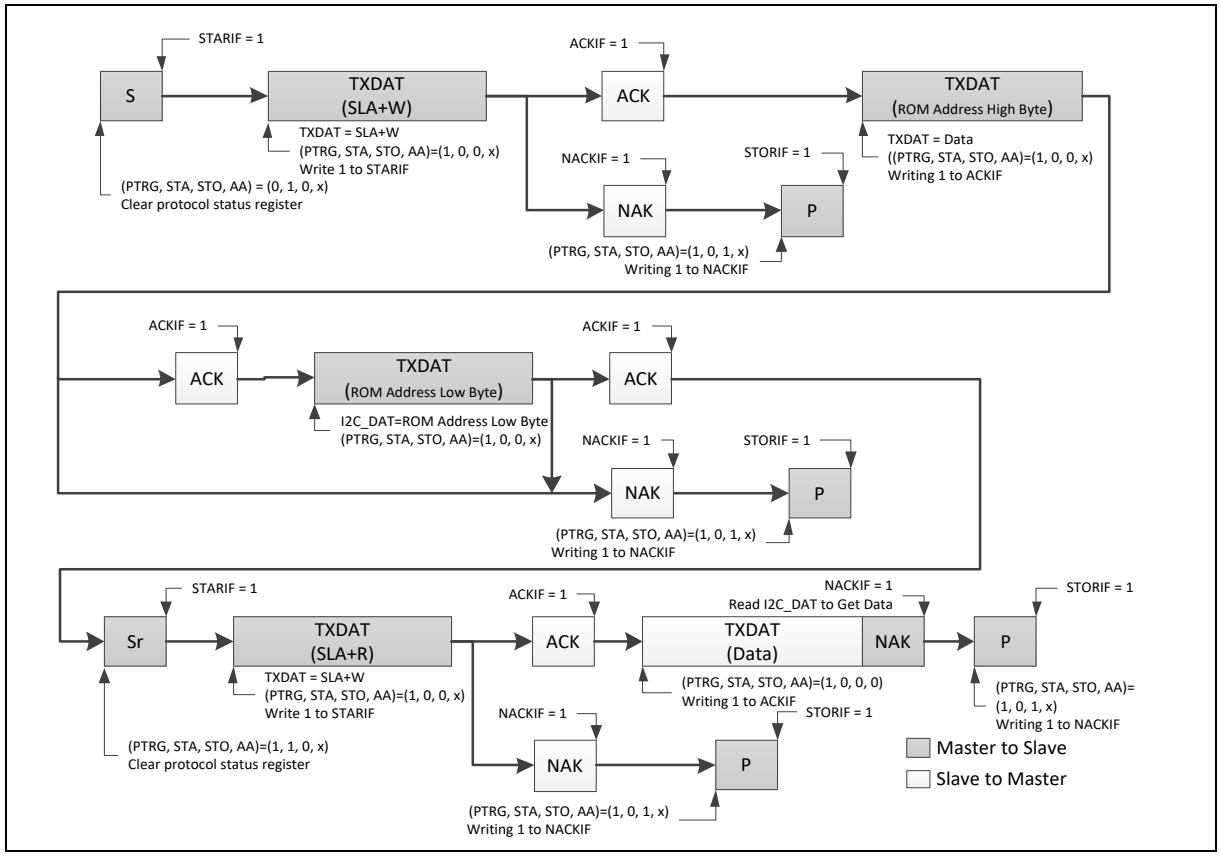


Figure 6.22-24 Protocol of EEPROM Random Read

The I²C controller, which is a master, sends START to bus. Then, it sends a SLA+W (Slave address + Write bit) to EERPOM followed by two bytes data address to set the EEPROM address to read. Finally, a Repeat START followed by SLA+R is sent to read the data from EEPROM.

6.22.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
USCI_I2C Base Address: **UI2Cn_BA = 0x400D_0000 + (0x1000 * n)** **n= 0**				
UI2C_CTL	UI2Cn_BA+0x00	R/W	USCI Control Register	0x0000_0000
UI2C_BRGEN	UI2Cn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00
UI2C_LINECTL	UI2Cn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000
UI2C_TXDAT	UI2Cn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000
UI2C_RXDAT	UI2Cn_BA+0x34	R	USCI Receive Data Register	0x0000_0000
UI2C_DEVADDR0	UI2Cn_BA+0x44	R/W	USCI Device Address Register 0	0x0000_0000
UI2C_DEVADDR1	UI2Cn_BA+0x48	R/W	USCI Device Address Register 1	0x0000_0000
UI2C_ADDRMSK0	UI2Cn_BA+0x4C	R/W	USCI Device Address Mask Register 0	0x0000_0000
UI2C_ADDRMSK1	UI2Cn_BA+0x50	R/W	USCI Device Address Mask Register 1	0x0000_0000
UI2C_WKCTL	UI2Cn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000
UI2C_WKSTS	UI2Cn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000
UI2C_PROTCTL	UI2Cn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0000
UI2C_PROTIEN	UI2Cn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000
UI2C_PROTSTS	UI2Cn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000
UI2C_TMCTL	UI2Cn_BA+0x8C	R/W	I^2C Timing Configure Control Register	0x0000_0000

6.22.7 Register Description

USCI Control Register (UI2C_CTL)

Register	Offset	R/W	Description	Reset Value
UI2C_CTL	UI2Cn_BA+0x00	R/W	USCI Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**FUNMODE**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**FUNMODE**	**Function Mode** This bit field selects the protocol for this USCI controller. Selecting a protocol that is not available or a reserved combination disables the USCI. When switching between two protocols, the USCI has to be disabled before selecting a new protocol. Simultaneously, the USCI will be reset when user write 000 to FUNMODE. 000 = The USCI is disabled. All protocol related state machines are set to idle state. 001 = The SPI protocol is selected. 010 = The UART protocol is selected. 100 = The I^2C protocol is selected. **Note:** Other bit combinations are reserved.

USCI Baud Rate Generator Register (UI2C_BRGEN)

Register	Offset	R/W	Description	Reset Value
UI2C_BRGEN	UI2Cn_BA+0x08	R/W	USCI Baud Rate Generator Register	0x0000_3C00

31	30	29	28	27	26	25	24
Reserved						CLKDIV	
23	22	21	20	19	18	17	16
CLKDIV							
15	14	13	12	11	10	9	8
Reserved	DSCNT					PDSCNT	
7	6	5	4	3	2	1	0
Reserved		TMCNTSRC	TMCNTEN	SPCLKSEL		PTCLKSEL	RCLKSEL

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25:16]	**CLKDIV**	**Clock Divider** This bit field defines the ratio between the protocol clock frequency f_{PROT_CLK} and the clock divider frequency f_{DIV_CLK} ($f_{DIV_CLK} = f_{PROT_CLK} / (CLKDIV+1)$).
[15]	**Reserved**	Reserved.
[14:10]	**DSCNT**	**Denominator for Sample Counter** This bit field defines the divide ratio of the sample clock f_{SAMP_CLK}. The divided frequency $f_{DS_CNT} = f_{PDS_CNT} / (DSCNT+1)$. **Note:** The maximum value of DSCNT is 0xF on UART mode and suggest to set over 4 to confirm the receiver data is sampled in right value.
[9:8]	**PDSCNT**	**Pre-divider for Sample Counter** This bit field defines the divide ratio of the clock division from sample clock f_{SAMP_CLK}. The divided frequency $f_{PDS_CNT} = f_{SAMP_CLK} / (PDSCNT+1)$.
[7:6]	**Reserved**	Reserved.
[5]	**TMCNTSRC**	**Time Measurement Counter Clock Source Selection** 0 = Time measurement counter with f_{PROT_CLK}. 1 = Time measurement counter with f_{DIV_CLK}.
[4]	**TMCNTEN**	**Time Measurement Counter Enable Bit** This bit enables the 10-bit timing measurement counter. 0 = Time measurement counter is Disabled. 1 = Time measurement counter is Enabled.
[3:2]	**SPCLKSEL**	**Sample Clock Source Selection** This bit field used for the clock source selection of a sample clock (f_{SAMP_CLK}) for the protocol processor. 00 = $f_{SAMP_CLK} = f_{DIV_CLK}$. 01 = $f_{SAMP_CLK} = f_{PROT_CLK}$. 10 = $f_{SAMP_CLK} = f_{SCLK}$.

		11 = f_{SAMP_CLK} = f_{REF_CLK}.
[1]	**PTCLKSEL**	**Protocol Clock Source Selection** This bit selects the source signal of protocol clock (f_{PROT_CLK}). 0 = Reference clock f_{REF_CLK}. 1 = f_{REF_CLK2} (its frequency is half of f_{REF_CLK}).
[0]	**RCLKSEL**	**Reference Clock Source Selection** This bit selects the source signal of reference clock (f_{REF_CLK}). 0 = Peripheral device clock f_{PCLK}. 1 = Reserved.

USCI Line Control Register (UI2C_LINECTL)

Register	Offset	R/W	Description	Reset Value
UI2C_LINECTL	UI2Cn_BA+0x2C	R/W	USCI Line Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**DWIDTH**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**LSB**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**DWIDTH**	**Word Length of Transmission** This bit field defines the data word length (amount of bits) for reception and transmission. The data word is always right-aligned in the data buffer. USCI support word length from 4 to 16 bits. 0x0: The data word contains 16 bits located at bit positions [15:0]. 0x1: Reserved. 0x2: Reserved. 0x3: Reserved. 0x4: The data word contains 4 bits located at bit positions [3:0]. 0x5: The data word contains 5 bits located at bit positions [4:0]. ... 0xF: The data word contains 15 bits located at bit positions [14:0].
[7:1]	**Reserved**	Reserved.
[0]	**LSB**	**LSB First Transmission Selection** 0 = The MSB, which bit of transmit/receive data buffer depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, the bit 0 of data buffer, will be transmitted/received first.

USCI Transmit Data Register (UI2C_TXDAT)

Register	Offset	R/W	Description	Reset Value
UI2C_TXDAT	UI2Cn_BA+0x30	W	USCI Transmit Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TXDAT**	**Transmit Data** Software can use this bit field to write 16-bit transmit data for transmission.

USCI Receive Data Register (UI2C_RXDAT)

Register	Offset	R/W	Description	Reset Value
UI2C_RXDAT	UI2Cn_BA+0x34	R	USCI Receive Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RXDAT**	**Received Data** This bit field monitors the received data which stored in receive data buffer. **Note:** In I^2C protocol, RXDAT[12:8] indicate the different transmission conditions which defined in I^2C.

USCI Device Address Register (UI2C_DEVADDR)

Register	Offset	R/W	Description	Reset Value
UI2C_DEVADDR0	UI2Cn_BA+0x44	R/W	USCI Device Address Register 0	0x0000_0000
UI2C_DEVADDR1	UI2Cn_BA+0x48	R/W	USCI Device Address Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						DEVADDR	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DEVADDR							

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**DEVADDR**	**Device Address** In I^2C protocol, this bit field contains the programmed slave address. If the first received address byte are 1111 $0AAX_B$, the AA bits are compared to the bits DEVADDR [9:8] to check for address match, where the X is R/W bit. Then the second address byte is also compared to DEVADDR [7:0]. **Note 1:** The DEVADDR [9:7] must be set 3'b000 when I^2C operating in 7-bit address mode. **Note 2:** When software set 10'h000, the address cannot be used.

USCI Device Address Mask Register (UI2C_ADDRMSK) – for I²C Only

Register	Offset	R/W	Description	Reset Value
UI2C_ADDRMSK0	UI2Cn_BA+0x4C	R/W	USCI Device Address Mask Register 0	0x0000_0000
UI2C_ADDRMSK1	UI2Cn_BA+0x50	R/W	USCI Device Address Mask Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						ADDRMSK	
7	6	5	4	3	2	1	0
ADDRMSK							

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**ADDRMSK**	**USCI Device Address Mask** 0 = Mask Disabled (the received corresponding register bit should be exact the same as address register.). 1 = Mask Enabled (the received corresponding address bit is don't care.). USCI support multiple address recognition with two address mask register. When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register. **Note:** The wake-up function cannot use address mask.

USCI Wake-up Control Register (UI2C_WKCTL)

Register	Offset	R/W	Description	Reset Value
UI2C_WKCTL	UI2Cn_BA+0x54	R/W	USCI Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						WKADDREN	WKEN

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WKADDREN**	**Wake-up Address Match Enable Bit** 0 = The chip is woken up according to START signal. 1 = The chip is woken up according to address match.
[0]	**WKEN**	**Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.

USCI Wake-up Status Register (UI2C_WKSTS)

Register	Offset	R/W	Description	Reset Value
UI2C_WKSTS	UI2Cn_BA+0x58	R/W	USCI Wake-up Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**WKF**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKF**	**Wake-up Flag** When chip is woken up from Power-down mode, this bit is set to 1. Software can write 1 to clear this bit.

USCI Protocol Control Register – I²C (UI2C_PROTCTL)

Register	Offset	R/W	Description	Reset Value
UI2C_PROTCTL	UI2Cn_BA+0x5C	R/W	USCI Protocol Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PROTEN	Reserved					TOCNT	
23	22	21	20	19	18	17	16
TOCNT							
15	14	13	12	11	10	9	8
Reserved						MONEN	SCLOUTEN
7	6	5	4	3	2	1	0
Reserved		PTRG	ADDR10EN	STA	STO	AA	GCFUNC

Bits	Description	
[31]	**PROTEN**	**I²C Protocol Enable Bit** 0 = I²C Protocol Disabled. 1 = I²C Protocol Enabled.
[30:26]	**Reserved**	Reserved.
[25:16]	**TOCNT**	**Time-out Clock Cycle** This bit field indicates how many clock cycle selected by TMCNTSRC (UI2C_BRGEN [5]) when each interrupt flags are clear. The time-out is enable when TOCNT bigger than 0. **Note:** The TMCNTSRC (UI2C_BRGEN [5]) must be set zero on I²C mode.
[15:10]	**Reserved**	Reserved.
[9]	**MONEN**	**Monitor Mode Enable Bit** This bit enables monitor mode. In monitor mode the SDA output will be put in high impedance mode. This prevents the I²C module from outputting data of any kind (including ACK) onto the I²C data bus. 0 = The monitor mode Disabled. 1 = The monitor mode Enabled. **Note:** Depending on the state of the SCLOUTEN bit, the SCL output may be also forced high, preventing the module from having control over the I²C clock line.
[8]	**SCLOUTEN**	**SCL Output Enable Bit** This bit enables monitor pulling SCL to low. This monitor will pull SCL to low until it has had time to respond to an I²C interrupt. 0 = SCL output will be forced high due to open drain mechanism. 1 = I²C module may act as a slave peripheral just like in normal operation, the I²C holds the clock line low until it has had time to clear I²C interrupt.
[7:6]	**Reserved**	Reserved.

[5]	**PTRG**	**I²C Protocol Trigger (Write Only)** When a new state is present in the UI2C_PROTSTS register, if the related interrupt enable bits are set, the I²C interrupt is requested. It must write one by software to this bit after the related interrupt flags are set to 1 and the I²C protocol function will go ahead until the STOP is active or the PROTEN is disabled. 0 = I²C 's stretch disabled and the I²C protocol function will go ahead. 1 = I²C 's stretch is active.
[4]	**ADDR10EN**	**Address 10-bit Function Enable Bit** 0 = Address match 10 bit function Disabled. 1 = Address match 10 bit function Enabled.
[3]	**STA**	**I²C START Control** Setting STA to logic 1 to enter Master mode, the I²C hardware sends a START or repeat START condition to bus when the bus is free.
[2]	**STO**	**I²C STOP Control** In Master mode, setting STO to transmit a STOP condition to bus then I²C hardware will check the bus condition if a STOP condition is detected this bit will be cleared by hardware automatically. In a slave mode, setting STO resets I²C hardware to the defined "not addressed" slave mode when bus error (UI2C_PROTSTS.ERRIF = 1).
[1]	**AA**	**Assert Acknowledge Control** When AA =1 prior to address or data received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when 1.) A slave is acknowledging the address sent from master, 2.) The receiver devices are acknowledging the data sent by transmitter. When AA=0 prior to address or data received, a Not acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line.
[0]	**GCFUNC**	**General Call Function** 0 = General Call Function Disabled. 1 = General Call Function Enabled. **Note:** When ADDR10EN (UI2C_PROTCTL [4]) is set, don't set this bit.

USCI Protocol Interrupt Enable Register – I²C (UI2C_PROTIEN)

Register	Offset	R/W	Description	Reset Value
UI2C_PROTIEN	UI2Cn_BA+0x60	R/W	USCI Protocol Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	ACKIEN	ERRIEN	ARBLOIEN	NACKIEN	STORIEN	STARIEN	TOIEN

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**ACKIEN**	**Acknowledge Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if an acknowledge is detected by a master. 0 = The acknowledge interrupt Disabled. 1 = The acknowledge interrupt Enabled.
[5]	**ERRIEN**	**Error Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if an I²C error condition is detected (indicated by ERR (UI2C_PROTSTS [16])). 0 = The error interrupt Disabled. 1 = The error interrupt Enabled.
[4]	**ARBLOIEN**	**Arbitration Lost Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if an arbitration lost event is detected. 0 = The arbitration lost interrupt Disabled. 1 = The arbitration lost interrupt Enabled.
[3]	**NACKIEN**	**Non - Acknowledge Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if a Non - acknowledge is detected by a master. 0 = The non - acknowledge interrupt Disabled. 1 = The non - acknowledge interrupt Enabled.
[2]	**STORIEN**	**STOP Condition Received Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if a STOP condition is detected. 0 = The stop condition interrupt Disabled. 1 = The stop condition interrupt Enabled.
[1]	**STARIEN**	**START Condition Received Interrupt Enable Bit** This bit enables the generation of a protocol interrupt if a START condition is detected. 0 = The start condition interrupt Disabled.

		1 = The start condition interrupt Enabled.
[0]	**TOIEN**	**Time-out Interrupt Enable Bit** In I^2C protocol, this bit enables the interrupt generation in case of a time-out event. 0 = The time-out interrupt Disabled. 1 = The time-out interrupt Enabled.

USCI Protocol Status Register – I²C (UI2C_PROTSTS)

Register	Offset	R/W	Description	Reset Value
UI2C_PROTSTS	UI2Cn_BA+0x64	R/W	USCI Protocol Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				ERRARBLO	BUSHANG	WRSTSWK	WKAKDONE
15	14	13	12	11	10	9	8
SLAREAD	SLASEL	ACKIF	ERRIF	ARBLOIF	NACKIF	STORIF	STARIF
7	6	5	4	3	2	1	0
Reserved	ONBUSY	TOIF	Reserved				

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19]	**ERRARBLO**	**Error Arbitration Lost** This bit indicates bus arbitration lost due to bigger noise which is can't be filtered by input processor. The I²C can send start condition when ERRARBLO is set. Thus this bit doesn't be cared on slave mode. 0 = The bus is normal status for transmission. 1 = The bus is error arbitration lost status for transmission. **Note:** This bit has no interrupt signal, and it will be cleared automatically by hardware when a START condition is present.
[18]	**BUSHANG**	**Bus Hang-up** This bit indicates bus hang-up status. There is 4-bit counter count when SCL hold high and refer f_{SAMP_CLK}. The hang-up counter will count to overflow and set this bit when SDA is low. The counter will be reset by falling edge of SCL signal. 0 = The bus is normal status for transmission. 1 = The bus is hang-up status for transmission. **Note:** This bit has no interrupt signal, and it will be cleared automatically by hardware when a START condition is present.
[17]	**WRSTSWK**	**Read/Write Status Bit in Address Wake-up Frame** 0 = Write command be record on the address match wake-up frame. 1 = Read command be record on the address match wake-up frame.
[16]	**WKAKDONE**	**Wake-up Address Frame Acknowledge Bit Done** 0 = The ACK bit cycle of address match frame isn't done. 1 = The ACK bit cycle of address match frame is done in power-down. **Note:** This bit can't release when WKUPIF is set.
[15]	**SLAREAD**	**Slave Read Request Status** This bit indicates that a slave read request has been detected. 0 = A slave R/W bit is 1 has not been detected. 1 = A slave R/W bit is 1 has been detected. **Note:** This bit has no interrupt signal, and it will be cleared automatically by hardware.

[14]	**SLASEL**	**Slave Select Status** This bit indicates that this device has been selected as slave. 0 = The device is not selected as slave. 1 = The device is selected as slave. **Note:** This bit has no interrupt signal, and it will be cleared automatically by hardware.
[13]	**ACKIF**	**Acknowledge Received Interrupt Flag** This bit indicates that an acknowledge has been received in master mode. A protocol interrupt can be generated if UI2C_PROTCTL.ACKIEN = 1. 0 = An acknowledge has not been received. 1 = An acknowledge has been received. **Note:** This bit is cleared by software writing 1 into this bit
[12]	**ERRIF**	**Error Interrupt Flag** This bit indicates that a Bus Error occurs when a START or STOP condition is present at an illegal position in the formation frame. Example of illegal position are during the serial transfer of an address byte, a data byte or an acknowledge bit. A protocol interrupt can be generated if UI2C_PROTCTL.ERRIEN = 1. 0 = An I^2C error has not been detected. 1 = An I^2C error has been detected. **Note 1:** This bit is cleared by software writing 1 into this bit. **Note 2:** This bit is set for slave mode, and user must write 1 into STO register to the defined “not addressed” slave mode.
[11]	**ARBLOIF**	**Arbitration Lost Interrupt Flag** This bit indicates that an arbitration has been lost. A protocol interrupt can be generated if UI2C_PROTCTL.ARBLOIEN = 1. 0 = An arbitration has not been lost. 1 = An arbitration has been lost. **Note:** This bit is cleared by software writing 1 into this bit.
[10]	**NACKIF**	**Non - Acknowledge Received Interrupt Flag** This bit indicates that a non - acknowledge has been received in master mode. A protocol interrupt can be generated if UI2C_PROTCTL.NACKIEN = 1. 0 = A non - acknowledge has not been received. 1 = A non - acknowledge has been received. **Note:** This bit is cleared by software writing 1 into this bit.
[9]	**STORIF**	**Stop Condition Received Interrupt Flag** This bit indicates that a stop condition has been detected on the I^2C bus lines. A protocol interrupt can be generated if UI2C_PROTCTL.STORIEN = 1. 0 = A stop condition has not yet been detected. 1 = A stop condition has been detected. **Note:** This bit is cleared by software writing 1 into this bit.
[8]	**STARIF**	**Start Condition Received Interrupt Flag** This bit indicates that a start condition or repeated start condition has been detected on master mode. However, this bit also indicates that a repeated start condition has been detected on slave mode. A protocol interrupt can be generated if UI2C_PROTCTL.STARIEN = 1. 0 = A start condition has not yet been detected. 1 = A start condition has been detected. **Note:** This bit is cleared by software writing 1 into this bit.
[7]	**Reserved**	Reserved.
[6]	**ONBUSY**	**On Bus Busy**

		Indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected 0 = The bus is IDLE (both SCLK and SDA High). 1 = The bus is busy.
[5]	**TOIF**	**Time-out Interrupt Flag** 0 = A time-out interrupt status has not occurred. 1 = A time-out interrupt status has occurred. **Note:** This bit is cleared by software writing 1 into this bit.
[4:0]	**Reserved**	Reserved.

USCI Timing Configure Control Register (UI2C_TMCTL)

Register	Offset	R/W	Description	Reset Value
UI2C_TMCTL	UI2Cn_BA+0x8C	R/W	I^2C Timing Configure Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							**HTCTL**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
HTCTL							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**STCTL**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STCTL							

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24:16]	**HTCTL**	**Hold Time Configure Control** This field is used to generate the delay timing between SCL falling edge SDA edge in transmission mode. The delay hold time is numbers of peripheral clock = HTCTL x f_{PCLK}.
[15:9]	**Reserved**	Reserved.
[8:0]	**STCTL**	**Setup Time Configure Control** This field is used to generate a delay timing between SDA edge and SCL rising edge in transmission mode.. The delay setup time is numbers of peripheral clock = STCTL x f_{PCLK}.

6.23 UART Interface Controller (UART)

6.23.1 Overview

The chip provides ten channels of Universal Asynchronous Receiver/Transmitters (UART). The UART controller performs serial-to-parallel conversion on data received from the peripheral and parallel-to-serial conversion on data transmitted from the CPU. Each UART controller channel supports ten types of interrupts. The UART controller supports flow control function. The UART controller also supports IrDA SIR, LIN, RS-485 and Single-wire function modes and auto-baud rate measuring function.

6.23.2 Features

- Full-duplex asynchronous communications
- Separates receive and transmit 16/16 bytes entry FIFO for data payloads
- Supports hardware auto-flow control
- Programmable receiver buffer trigger level
- Supports programmable baud rate generator for each channel individually
- Supports nCTS, incoming data, Received Data FIFO reached threshold and RS-485 Address Match (AAD mode) wake-up function
- Supports 8-bit receiver buffer time-out detection function
- Programmable transmitting data delay time between the last stop and the next START bit by setting DLY (UART_TOUT [15:8])
- Supports Auto-Baud Rate measurement and baud rate compensation function
  - 9600 bps for UART_CLK is selected LXT.
- Supports break error, frame error, parity error and receive/transmit buffer overflow detection function
- Fully programmable serial-interface characteristics
  - Programmable number of data bit, 5-, 6-, 7-, 8- bit character
  - Programmable PARITY bit, even, odd, no parity or stick PARITY bit generation and detection
  - Programmable STOP bit, 1, 1.5, or 2 STOP bit generation
- Supports IrDA SIR function mode
  - 3/16 bit duration for normal mode
- Supports LIN function mode (Only UART0/UART1 with LIN function)
  - LIN master/slave mode
  - Programmable break generation function for transmitter
  - Break detection function for receiver
- Supports RS-485 function mode
  - RS-485 9-bit mode
  - Hardware or software enables to program nRTS pin to control RS-485 transmission direction
- Supports PDMA transfer function
- Supports Single-wire function mode.

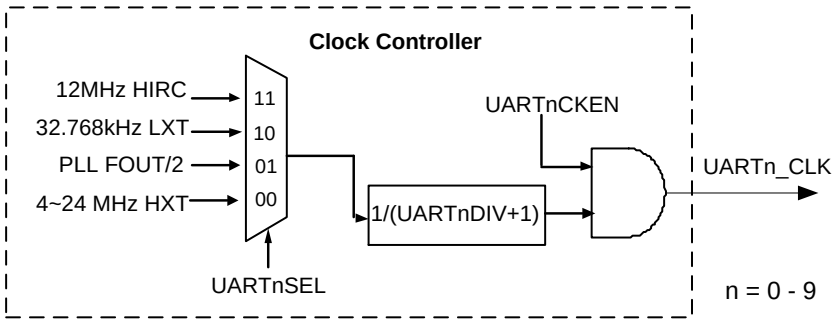
UART Feature	UART0/ UART1	UART2 ~ UART9	USCI-UART
FIFO	16 Bytes	16 Bytes	TX: 1byte RX: 2byte
Auto Flow Control (CTS/RTS)	√	√	√
IrDA	√	√	-
LIN	√	-	-
RS-485 Function Mode	√	√	√
nCTS Wake-up	√	√	√
Incoming Data Wake-up	√	√	√
Received Data FIFO reached threshold Wake-up	√	√	-
RS-485 Address Match (AAD mode) Wake-up	√	√	-
Received Data FIFO reached threshold Time-out Wake-up	√	√	-
Baud Rate Compensation	√	√	-
Auto-Baud Rate Measurement	√	√	√
STOP bit Length	1, 1.5, 2 bit	1, 1.5, 2 bit	1, 2 bit
Word Length	5, 6, 7, 8 bits	5, 6, 7, 8 bits	6~13 bits
Even / Odd Parity	√	√	√
Stick Bit	√	√	-

Table 6.23-1 M463/M467 Series UART Features

6.23.3 Block Diagram

The UART clock control and block diagram are shown in Figure 6.23-1 and Figure 6.23-2 respectively.

Note: The frequency of UARTx_CLK should not be greater than 30 times HCLK.



UART0SEL	UART0DIV	UART0CKEN
CLK_CLKSEL1[25:24]	CLK_CLKDIV0[11:8]	CLK_APBCLK0[16]

UART1SEL	UART1DIV	UART1CKEN
CLK_CLKSEL1[27:26]	CLK_CLKDIV0[15:12]	CLK_APBCLK0[17]

UART2SEL	UART2DIV	UART2CKEN
CLK_CLKSEL3[25:24]	CLK_CLKDIV4[3:0]	CLK_APBCLK0[18]

UART3SEL	UART3DIV	UART3CKEN
CLK_CLKSEL3[27:26]	CLK_CLKDIV4[7:4]	CLK_APBCLK0[19]

UART4SEL	UART4DIV	UART4CKEN
CLK_CLKSEL3[29:28]	CLK_CLKDIV4[11:8]	CLK_APBCLK0[20]

UART5SEL	UART5DIV	UART5CKEN
CLK_CLKSEL3[31:30]	CLK_CLKDIV4[15:12]	CLK_APBCLK0[21]

UART6SEL	UART6DIV	UART6CKEN
CLK_CLKSEL3[21:20]	CLK_CLKDIV4[19:16]	CLK_APBCLK0[22]

UART7SEL	UART7DIV	UART7CKEN
CLK_CLKSEL3[23:22]	CLK_CLKDIV4[23:20]	CLK_APBCLK0[23]

UART8SEL	UART8DIV	UART8CKEN
CLK_CLKSEL2[21:20]	CLK_CLKDIV5[19:16]	CLK_APBCLK2[16]

UART9SEL	UART9DIV	UART9CKEN
CLK_CLKSEL2[23:22]	CLK_CLKDIV5[23:20]	CLK_APBCLK2[17]

Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.23-1 UART Clock Control Diagram

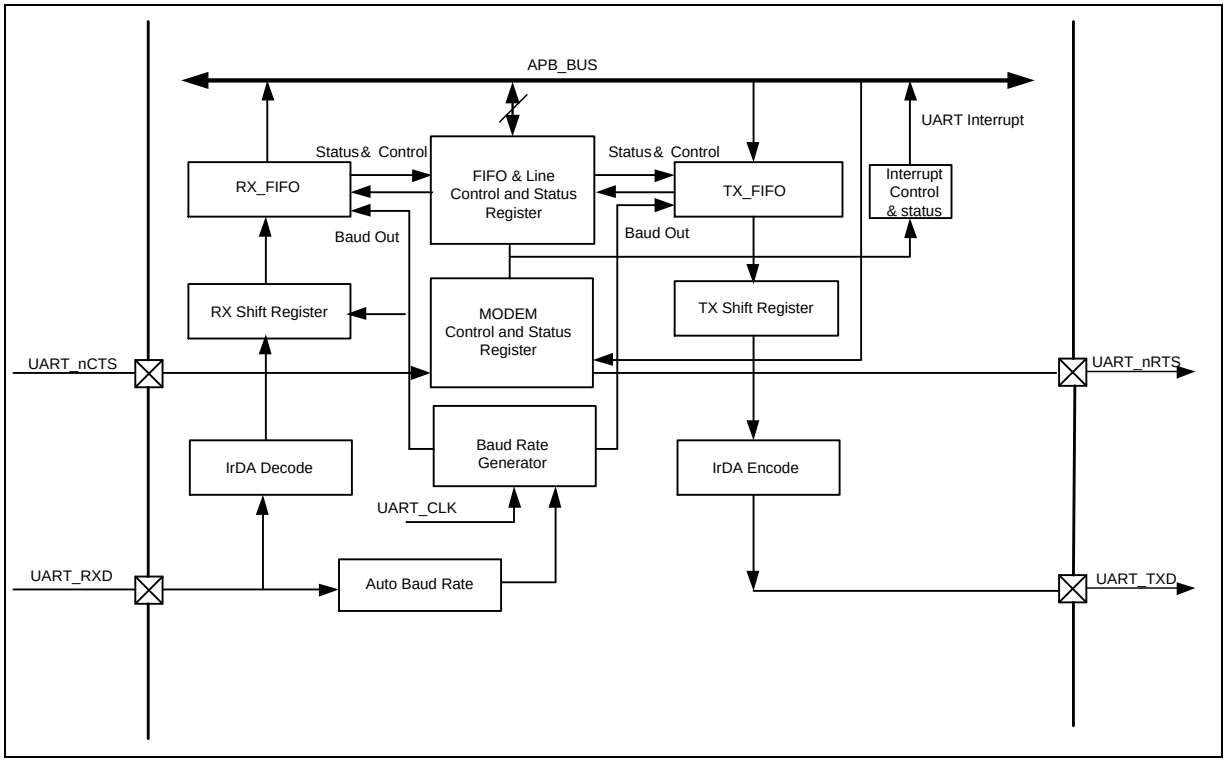


Figure 6.23-2 UART Block Diagram

Each block is described in detail as follows:

TX_FIFO

The transmitter is buffered with a 16 bytes FIFO to reduce the number of interrupts presented to the CPU.

RX_FIFO

The receiver is buffered with a 16 bytes FIFO (plus three error bits, BIF (UART_FIFOSTS[6]), FEF (UART_FIFOSTS[5]), PEF (UART_FIFOSTS[4])) to reduce the number of interrupts presented to the CPU.

TX Shift Register

This block is responsible for shifting out the transmitting data serially.

RX Shift Register

This block is responsible for shifting in the receiving data serially.

Modem Control and Status Register

This register controls the interface to the MODEM or data set (or a peripheral device emulating a MODEM).

Baud Rate Generator

Divide the external clock by the divisor to get the desired baud rate clock. Refer to baud rate equation.

IrDA Encode

This block is IrDA encoding control block.

IrDA Decode

This block is IrDA decoding control block.

FIFO & Line Control and Status Register

This field is register set including the FIFO control register (UART_FIFO), FIFO status register (UART_FIFOSTS), and line control register (UART_LINE) for transmitter and receiver. The time-out register (UART_TOUT) identifies the condition of time-out interrupt.

Auto-Baud Rate Measurement

This block is responsible for auto-baud rate measurement.

Interrupt Control and Status Register

There are eleven types of interrupts. Interrupt enable register (UART_INTEN) enable or disable the responding interrupt and interrupt status register (UART_INTSTS) identifying the occurrence of the responding interrupt.

Interrupt	Description
RDAINT	Receive Data Available Interrupt.
THREINT	Transmit Holding Register Empty Interrupt.
TXENDINT	Transmitter Empty Interrupt.
RLSINT	Receive Line Status Interrupt (parity error or frame error or break error).
MODEMINT	MODEM Status Interrupt.
RXTOINT	Receiver Buffer Time-out Interrupt.
BUFERRINT	Buffer Error Interrupt.
LININT	LIN Bus Interrupt.
WKINT	Wake-up Interrupt.
ABRINT	Auto-Baud Rate Interrupt.
SWBEINT	Single-wire Bit Error Detect Interrupt.

Table 6.23-2 UART Interrupt

6.23.4 Basic Configuration

6.23.4.1 UART0 Basic Configuration

- Clock Source Configuration
  - Select the source of UART0 peripheral clock on UART0SEL (CLK_CLKSEL1[25:24]).
  - Select the clock divider number of UART0 peripheral clock on UART0DIV (CLK_CLKDIV0[11:8]).
  - Enable UART0 peripheral clock in UART0CKEN (CLK_APBCLK0[16]).
- Reset UART0 controller in UART0RST (SYS_IPRST1[16]).

6.23.4.2 UART1 Basic Configuration

- Clock Source Configuration
  - Select the source of UART1 peripheral clock on UART1SEL (CLK_CLKSEL1[27:26]).
  - Select the clock divider number of UART1 peripheral clock on UART1DIV (CLK_CLKDIV0[15:12]).
  - Enable UART1 peripheral clock in UART1CKEN (CLK_APBCLK0[17]).

- Reset UART1 controller in UART1RST (SYS_IPRST1[17]).

6.23.4.3 UART2 Basic Configuration

- Clock Source Configuration
  - Select the source of UART2 peripheral clock on UART2SEL (CLK_CLKSEL3[25:24]).
  - Select clock divider number of UART2 peripheral clock on UART2DIV (CLK_CLKDIV4[3:0]).
  - Enable UART2 peripheral clock in UART2CKEN (CLK_APBCLK0[18]).
- Reset UART2 controller in UART2RST (SYS_IPRST1[18]).

6.23.4.4 UART3 Basic Configuration

- Clock Source Configuration
  - Select the source of UART3 peripheral clock on UART3SEL (CLK_CLKSEL3[27:26]).
  - Select the clock divider number of UART3 peripheral clock on UART3DIV (CLK_CLKDIV4[7:4]).
  - Enable UART3 peripheral clock in UART3CKEN (CLK_APBCLK0[19]).
- Reset UART3 controller in UART3RST (SYS_IPRST1[19]).

6.23.4.5 UART4 Basic Configuration

- Clock Source Configuration
  - Select the source of UART4 peripheral clock on UART4SEL (CLK_CLKSEL3[29:28]).
  - Select the clock divider number of UART4 peripheral clock on UART4DIV (CLK_CLKDIV4[11:8]).
  - Enable UART4 peripheral clock in UART4CKEN (CLK_APBCLK0[20]).
- Reset UART4 controller in UART4RST (SYS_IPRST1[20]).

6.23.4.6 UART5 Basic Configuration

- Clock Source Configuration
  - Select the source of UART5 peripheral clock on UART5SEL (CLK_CLKSEL3[31:30]).
  - Select the clock divider number of UART5 peripheral clock on UART5DIV (CLK_CLKDIV4[15:12]).
  - Enable UART5 peripheral clock in UART5CKEN (CLK_APBCLK0[21]).
- Reset UART5 controller in UART5RST (SYS_IPRST1[21]).

6.23.4.7 UART6 Basic Configuration

- Clock Source Configuration
  - Select the source of UART6 peripheral clock on UART6SEL (CLK_CLKSEL3[21:20]).
  - Select the clock divider number of UART6 peripheral clock on UART6DIV (CLK_CLKDIV4[19:16]).
  - Enable UART6 peripheral clock in UART6CKEN (CLK_APBCLK0[22]).
- Reset UART6 controller in UART6RST (SYS_IPRST1[22]).

6.23.4.8 UART7 Basic Configuration

- Clock Source Configuration
  - Select the source of UART7 peripheral clock on UART7SEL (CLK_CLKSEL3[23:22]).

- Select the clock divider number of UART7 peripheral clock on UART7DIV (CLK_CLKDIV4[23:20]).
- Enable UART7 peripheral clock in UART7CKEN (CLK_APBCLK0[23]).

- Reset UART7 controller in UART7RST (SYS_IPRST1[23]).

6.23.4.9 UART8 Basic Configuration

- Clock Source Configuration
  - Select the source of UART8 peripheral clock on UART8SEL (CLK_CLKSEL2[21:20]).
  - Select the clock divider number of UART8 peripheral clock on UART8DIV (CLK_CLKDIV5[19:16]).
  - Enable UART8 peripheral clock in UART8CKEN (CLK_APBCLK2[16]).
- Reset UART8 controller in UART8RST (SYS_IPRST3[16]).

6.23.4.10 UART9 Basic Configuration

- Clock Source Configuration
  - Select the source of UART9 peripheral clock on UART9SEL (CLK_CLKSEL2[23:22]).
  - Select the clock divider number of UART9 peripheral clock on UART9DIV (CLK_CLKDIV5[23:20]).
  - Enable UART9 peripheral clock in UART9CKEN (CLK_APBCLK2[17]).
- Reset UART9 controller in UART9RST (SYS_IPRST3[17]).

UART Interface Controller Pin description is shown in Table 6.23-3:

Pin	Type	Description
UARTx_TXD	Output	UARTx transmit
UARTx_RXD	Input	UARTx receive
UARTx_nCTS	Input	UARTx modem clear to send
UARTx_nRTS	Output	UARTx modem request to send

Table 6.23-3 UART Interface Controller Pin

6.23.5 Functional Description

The UART controller supports five function modes including UART, IrDA, LIN, RS-485 and Single-wire mode. User can select a function by setting the UART_FUNCSEL register. The five function modes will be described in following section.

6.23.5.1 UART Controller Baud Rate Generator

The UART controller includes a programmable baud rate generator capable of dividing clock input by divisors to produce the serial clock that transmitter and receiver need. Table 6.23-4 list the UART baud rate equations in the various conditions. Table 6.23-5 and Table 6.23-6 list the UART baud rate parameter and register setting example. In IrDA function mode, the baud rate generator must be set in mode 0. More detailed register description is shown in UART_BAUD register. There are three setting modes. Mode 0 is set by UART_BAUD[29:28] with 00. Mode 1 is set by UART_BAUD[29:28] with 10. Mode 2 is set by UART_BAUD[29:28] with 11.

Mode	BAUDM1	BAUDM0	Baud Rate Equation
Mode 0	0	0	UART_CLK / [16 * (BRD+2)].

Mode 1	1	0	UART_CLK / [(EDIVM1+1) * (BRD+2)], EDIVM1 must >= 8.
Mode 2	1	1	UART_CLK / (BRD+2) If UART_CLK <= 3*HCLK, BRD must >= 8. If UART_CLK > 3*HCLK, BRD must >= 3*N – 1. N is the smallest integer larger than or equal to the ratio of UART_CLK /HCLK. For example, if 2*HCLK < UART_CLK =< 3*HCLK, BRD must >=11. if 4*HCLK < UART_CLK =< 5*HCLK, BRD must >=14. (If the UART_CLK is selected from LXT, BRD can be greater than or equal to 1)

Table 6.23-4 UART Controller Baud Rate Equation Table

UART Peripheral Clock = 12 MHz			
Baud Rate	Mode 0	Mode 1	Mode 2
921600	Not support	Not recommended	BRD=11
460800	Not recommended	BRD=0, EDIVM1 =12	BRD=24
230400	Not recommended	BRD =2, EDIVM1 =12	BRD =50
115200	Not recommended	BRD =6, EDIVM1 =12	BRD =102
57600	BRD =11	BRD =14, EDIVM1 =12	BRD =206
38400	BRD =18	BRD =22, EDIVM1 =12	BRD =311
19200	BRD =37	BRD =61, EDIVM1 =9	BRD =623
9600	BRD =76	BRD =123, EDIVM1 =9	BRD =1248
4800	BRD =154	BRD =248, EDIVM1 =9	BRD =2498

Table 6.23-5 UART Controller Baud Rate Parameter Setting Example Table

UART Peripheral Clock = 12 MHz			
Baud Rate	UART_BAUD Value		
	Mode 0	Mode 1	Mode 2
921600	Not support	Not recommended	0x3000_000B
460800	Not recommended	0x2C00_0000	0x3000_0018
230400	Not recommended	0x2C00_0002	0x3000_0032
115200	Not recommended	0x2C00_0006	0x3000_0066
57600	0x0000_000B	0x2C00_000E	0x3000_00CE
38400	0x0000_0012	0x2C00_0016	0x3000_0137
19200	0x0000_0025	0x2900_003D	0x3000_026F
9600	0x0000_004C	0x2900_007B	0x3000_04E0
4800	0x0000_009A	0x2900_00F8	0x3000_09C2

Table 6.23-6 UART Controller Baud Rate Register Setting Example Table

6.23.5.2 UART Controller Baud Rate Compensation

The UART controller supports baud rate compensation function. It is used to optimize the precision in each bit. The precision of the compensation is half of UART module clock because there is BRCOMDEC (UART_BRCOMP[31]) to define the positive or negative compensation in each bit. If the BRCOMDEC (UART_BRCOMP[31]) = 0, it is positive compensation for each bit, one more module clock will be append in the compensated bit. If the BRCOMPDEC (UART_BRCOMP[31]) = 1, it is negative compensation for each bit, decrease one module clock in the compensated bit.

There is 9-bits location, BRCOMP[8:0] (UART_BRCOMP[8:0]), can be configured by user to define the relative bit is compensated or not. BRCOMP[7:0] is used to define the compensation of DAT (UART_DAT[7:0]) and BRCOMP[8] is used to define PARITY (UART_DAT[8]).

Example:

1. UART's peripheral clock = 32.768 kHz and baud rate is 9600 bps

Baud rate is 9600 bps, UART peripheral clock is 32.768 kHz → 3.413 peripheral clock/bit

If the baud divider is set 1 (3 peripheral clock/bit), the inaccuracy of each bit is -0.413 peripheral clock and BRCOMPDEC (UART_BRCOMP[31]) = 0, so that the BRCOMP (UART_BRCOMP[8:0]) can be set as 9'b010100101 = 0xa5.

Bit	Name	Total INACCURACY	BRCOMP Compensated	Final Inaccuracy
0	START	-0.413	x	-0.413
1	UART_DAT[0]	-0.826(-0.413-0.413)	1	0.174
2	UART_DAT[1]	-0.239(0.174-0.413)	0	-0.239
3	UART_DAT[2]	-0.652(-0.239-0.413)	1	0.348
4	UART_DAT[3]	-0.065(0.348-0.413)	0	-0.065
5	UART_DAT[4]	-0.478(-0.065-0.413)	0	-0.478
6	UART_DAT[5]	-0.891(-0.478-0.413)	1	0.109
7	UART_DAT[6]	-0.304(0.109-0.413)	0	-0.304
8	UART_DAT[7]	-0.717(-0.304-0.413)	1	0.283
9	PARITY	-0.130(0.283-0.413)	0	-0.13

Table 6.23-7 Baud Rate Compensation Example Table 1

2. UART's peripheral clock = 32.768 kHz and baud rate is 4800

Baud rate is 4800, UART peripheral clock is 32.768 kHz → 6.827 peripheral clock/bit

If the baud divider is set 5 (7 peripheral clock/bit), the inaccuracy of each bit is 0.173 peripheral clock and BRCOMPDEC (UART_BRCOMP[31]) = 1, so that the BRCOMP (UART_BRCOMP[8:0]) can be set as 9'b010000010 = 0x82.

Bit	Name	Total INACCURACY	BRCOMP Compensated	Final Inaccuracy
0	START	0.173	x	0.173
1	UART_DAT[0]	0.346(0.173+0.173)	0	0.346
2	UART_DAT[1]	0.519(0.346+0.173)	1	-0.481
3	UART_DAT[2]	-0.308(-0.481+0.173)	0	-0.308

4	UART_DAT[3]	-0.135(-0.308+0.173)	0	-0.135
5	UART_DAT[4]	-0.038(-0.135+0.173)	0	0.038
6	UART_DAT[5]	0.211(0.038+0.173)	0	0.211
7	UART_DAT[6]	0.384(0.211+0.173)	0	0.384
8	UART_DAT[7]	0.557(0.384+0.173)	1	-0.443
9	PARITY	-0.270(-0.443+0.173)	0	-0.270

Table 6.23-8 Baud Rate Compensation Example Table 2

6.23.5.3 UART Controller Auto-Baud Rate Function Mode:

Auto-Baud Rate function can measure baud rate of receiving data from UART RX pin automatically. When the Auto-Baud Rate measurement is finished, the measuring baud rate is loaded to BRD (UART_BAUD[15:0]). Both of the BAUDM1 (UART_BAUD[29]) and BAUDM0 (UART_BAUD[28]) are set to 1 automatically. UART RX data from START bit to 1st rising edge time is set by $2^{ABRDBITS}$ bit time in Auto-Baud Rate function detection frame.

The $2^{ABRDBITS}$ bit time from START bit to the 1st rising edge is calculated by setting ABRDBITS (UART_ALTCTL[20:19]). Set ABRDEN (UART_ALTCTL[18]) to enable auto-baud rate function. In the beginning when there is no data transfer, the UART RX is held high. Once the falling edge is detected, START bit is received. The auto-baud rate counter will be reset and then start counting. The auto-baud rate counter will be stopoed when the 1st rising edge is detected. Then, the auto-baud rate counter value divided by ABRDBITS (UART_ALTCTL[20:19]) is loaded to BRD (UART_BAUD[15:0]) automatically. ABRDEN (UART_ALTCTL[18]) is cleared. The Auto-Baud measurement is shown in Figure 6.23-3. Once the auto-baud rate measurement is finished, the ABRDIF (UART_FIFOSTS[1]) is set. When auto-baud rate counter is overflow, ABRDTOIF (UART_FIFOSTS[2]) is set. If ABRDIF (UART_FIFOSTS[1]) or ABRDTOIF (UART_FIFOSTS[2]) is set, the auto-baud rate flag ABRIF (UART_ALTCTL[17]) generated. If ABRIEN (UART_INTEN[18]) is enabled, the auto-baud rate interrupt ABRINT (UART_INTSTS[31]) is generated when ABRIF (UART_ALTCTL[17]) is set.

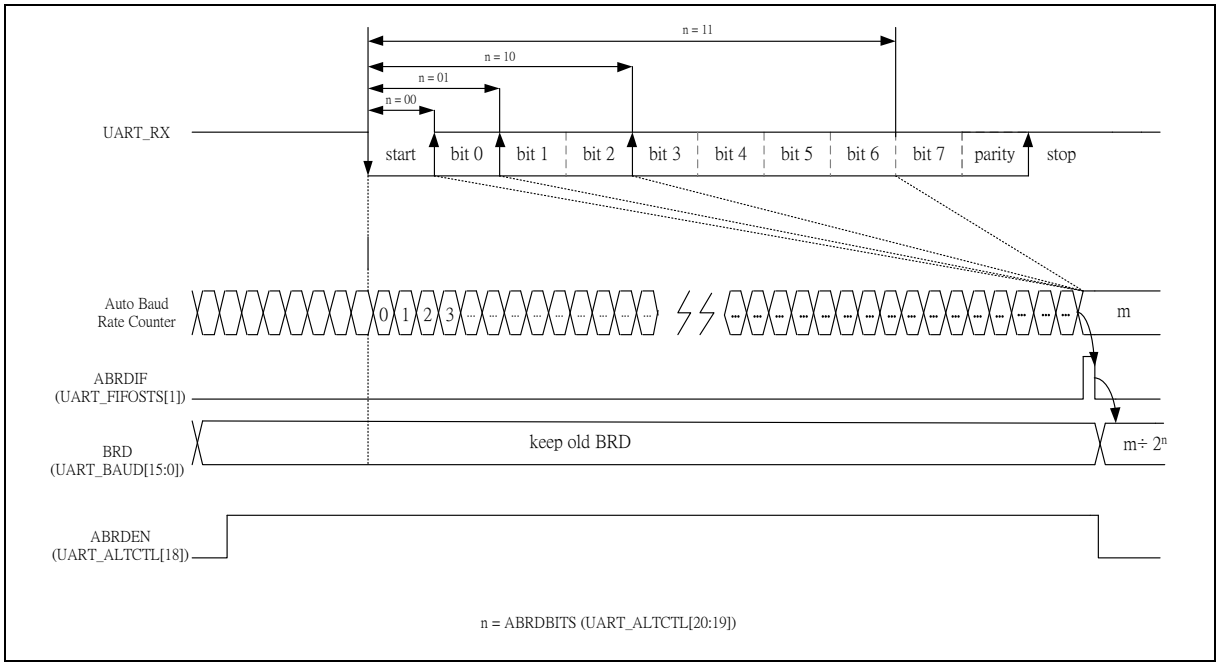


Figure 6.23-3 Auto-Baud Rate Measurement

Programming Sequence Example:

1. Program ABRDBITS (UART_ALTCTL[20:19]) to determine UART RX data 1st rising edge time from Start by $2^{ABRDBITS}$ bit time.
2. Set ABRIEN (UART_INTEN[18]) to enable auto-baud rate function interrupt.
3. Set ABRDEN (UART_ALTCTL[18]) to enable auto-baud rate function.
4. ABRDIF (UART_FIFOSTS[1]) is set, the auto-baud rate measurement is finished.
5. Operate UART transmit and receive action.
6. ABRDTOIF (UART_FIFOSTS[2]) is set, if auto-baud rate counter is overflow.
7. Go to Step 3.

6.23.5.4 UART Controller Transmit Delay Time Value

The UART controller programs DLY (UART_TOUT [15:8]) to control the transfer delay time between the last STOP bit and next START bit in transmission. The unit is baud. The operation is shown in Figure 6.23-4.

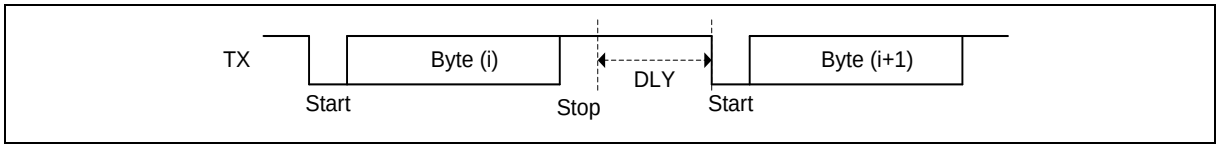


Figure 6.23-4 Transmit Delay Time Operation

6.23.5.5 UART Controller FIFO Control and Status

The UART controller is built-in with a 16 bytes transmitter FIFO (TX_FIFO) and a 16 bytes receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU. The CPU can read the status of the UART at any time during operation. The reported status information includes condition of the transfer operations being performed by the UART, as well as 3 error conditions (parity error, framing error, break interrupt) occur if receiving data has parity, frame or break error. UART, IrDA, LIN and RS-485 mode support FIFO control and status function.

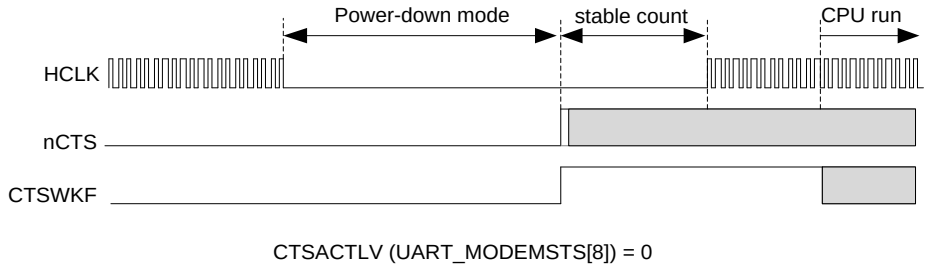
6.23.5.6 UART Controller Wake-up Function

The UART controller supports wake-up system function. The wake-up function includes nCTS pin, incoming data wake-up, Received Data FIFO reached threshold wake-up, RS-485 Address Match (AAD mode) wake-up and Received Data FIFO threshold time-out wake-up function. CTSWKF (UART_WKSTS[0]), DATWKF (UART_WKSTS[1]), RFRTWKF (UART_WKSTS[2]), RS485WKF (UART_WKSTS[3]) or TOUTWKF (UART_WKSTS[4]) cause the wake-up interrupt flag WKIF(UART_INTSTS[6]) generated. If the WKIEN (UART_INTEN[6]) is enabled, the wake-up interrupt flag WKIF(UART_INTSTS[6]) cause the wake-up interrupt WKINT (UART_INTSTS[14]) generated.

nCTS pin wake-up :

When the system is in Power-down mode and WKCTSEN (UART_WKCTL[0]) is set, the toggle of nCTS pin can wake-up system. If the WKCTSEN (UART_WKCTL[0]) is enabled, the toggle of nCTS pin cause the nCTS wake-up flag CTSWKF (UART_WKSTS[0]) generated. The nCTS wake-up is shown in Figure 6.23-5 and Figure 6.23-6.

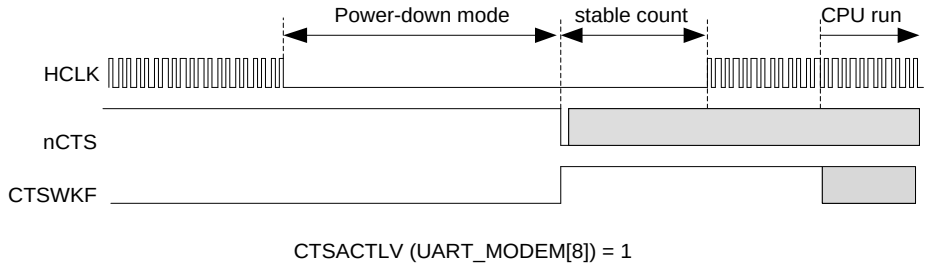
nCTS Wake-up Case 1 (nCTS transition from low to high)



Note: Stable count means HCLK source recovery stable count.

Figure 6.23-5 UART nCTS Wake-up Case1

nCTS Wake-up Case 2 (nCTS transition from high to low)



Note: Stable count means HCLK source recovery stable count.

Figure 6.23-6 UART nCTS Wake-up Case2

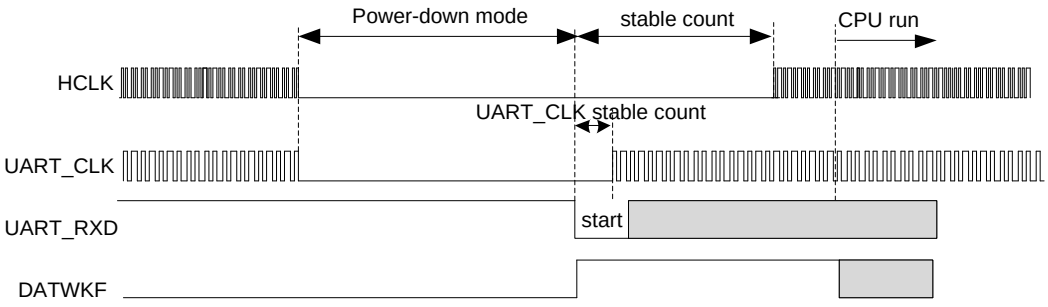
Incoming Data Wake-up

When system is in Power-down mode and the WKDATEN (UART_WKCTL [1]) is set, the toggle of incoming data (UART_RXD) pin can wake-up the system. In order to receive the incoming data after the system wake-up, the STCOMP (UART_DWKCOMP[15:0]) shall be set. These bits field of STCOMP indicate how many clock cycle selected by UART_CLK do the UART controller can get the 1st bit (START bit) when the system is wakeup from Power-down mode.

When incoming data wakes system up, the incoming data will be received and stored in FIFO. If the WKDATEN (UART_WKCTL[1]) is enabled, the toggle of incoming data (UART_RXD) pin cause the incoming data wake-up flag DATWKF (UART_WKSTS[1]) generated. The incoming data wake-up is shown in Figure 6.23-7.

Note 1: The UART controller clock source should be selected as HIRC, and the compensation time for START bit that refer to the M463/M467 series Datasheet for detailed information about wakeup time electrical characteristics. The STCOMP (UART_DWKCOMP[15:0]) = (wake-up stable time) * (HIRC frequency).

Note 2: The value of BRD(UART_BAUD[15:0]) should be greater than STCOMP (UART_DWKCOMP[15:0]).



Note 1: Stable count means HCLK source recovery stable count.

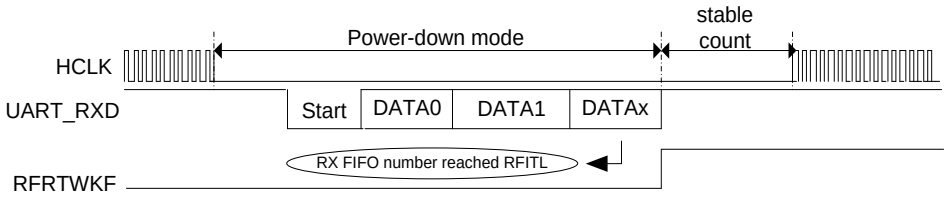
Note 2: UART_CLK stable count means UART clock source recovery stable count.

Figure 6.23-7 UART Data Wake-up

Received Data FIFO Reached Threshold Wake-up

The received data FIFO threshold reached wake-up function is enabled by setting WKRFRTEN (UART_WKCTL[2]). In Power-down mode, when the number of received data in RX FIFO reaches the threshold value RFITL (UART_FIFO[7:4]), it can wake-up the system. If the WKRFRTEN (UART_WKCTL[2]) is enabled, the number of received data in RX FIFO reaches the threshold value RFITL (UART_FIFO[7:4]) cause the received data FIFO reached threshold wake-up flag RFRTWKF (UART_WKSTS[2]) generated. The Received Data FIFO reached threshold wake-up is shown in Figure 6.23-8.

Note: The UART controller clock source should be selected as LXT in Power-down mode to receive data.



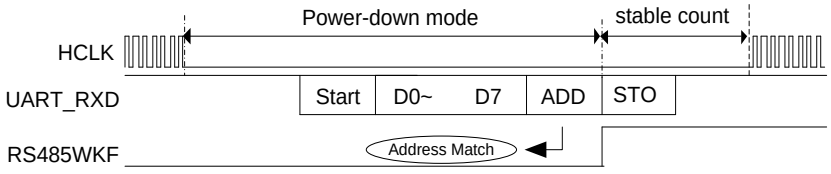
Note: Stable count means HCLK source recovery stable count.

Figure 6.23-8 UART Received Data FIFO reached threshold wake-up

RS-485 Address Match (AAD Mode) Wake-up

The RS-485 address match wake-up function is enabled by setting WKRFRTEN (UART_WKCTL[2]) and WKRS485EN (UART_WKCTL[3]). This function is used for RS-485 Auto Address Detection (AAD) mode in RS-485 function mode and ADDRDEN (UART_ALTCTL[15]) is set to 1. In Power-down mode, when an address byte is detected and matches the ADDRMV (UART_ALTCTL[31:24]) or the number of received data in RX FIFO reaches the threshold value RFITL (UART_FIFO[7:4]), it can wake-up the system. If the WKRS485EN (UART_WKCTL[3]) is enabled, when an address byte is detected and matches the ADDRMV (UART_ALTCTL[31:24]), the RS485 address match (AAD mode) wake-up flag RS485WKF (UART_WKSTS[3]) generated. The RS-485 Address Match (AAD mode) wake-up is shown in Figure 6.23-9.

Note: The UART controller clock source should be selected as LXT in Power-down mode to receive data.



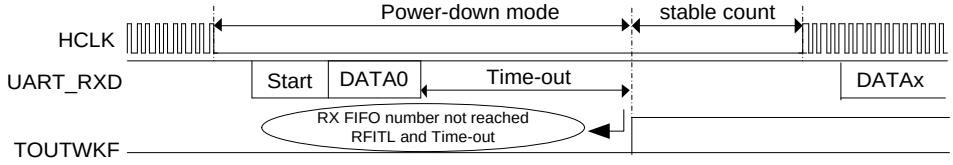
Note: Stable count means HCLK source recovery stable count.

Figure 6.23-9 UART RS-485 AAD Mode Address Match Wake-up

Received Data FIFO Threshold Time-out Wake-up

The received data FIFO threshold time-out wake-up function is enabled by setting WKRFRTEN (UART_WKCTL[2]) and WKTOUTEN (UART_WKCTL[4]). Setting TOCNTEN (UART_INTEN[11]) to enable receiver buffer time-out counter. In Power-down mode, when the number of received data in RX FIFO does not reach the threshold value RFITL (UART_FIFO[7:4]) and the time-out counter equals to the time-out value TOIC (UART_TOUT[7:0]), it can wake-up the system. If the WKTOUTEN (UART_WKCTL[4]) is enabled, when the time-out counter equals to the time-out value TOIC (UART_TOUT[7:0]), the Received Data FIFO threshold time-out wake-up flag TOUTWKF (UART_WKSTS[4]) generated. The Received Data FIFO threshold time-out wake-up is shown in Figure 6.23-10.

Note: The UART controller clock source should be selected as LXT in Power-down mode to receive data.



Note: Stable count means HCLK source recovery stable count.

Figure 6.23-10 UART Received Data FIFO threshold time-out wake-up

6.23.5.7 UART Controller Interrupt and Status

Each UART controller supports eleven types of interrupts including:

- Receive Data Available Interrupt (RDAINT)
- Transmit Holding Register Empty Interrupt (THREINT)
- Transmitter Empty Interrupt (TXENDIF)
- Receive Line Status Interrupt (RLSINT)
  - Break Interrupt Flag (BIF)
  - Framing Error Flag (FEF)
  - Parity Error Flag (PEF)
  - RS-485 Address Byte Detect Flag (ADDRDETF)
- MODEM Status Interrupt (MODEMINT)
  - Detect nCTS State Change Flag (CTSDETF)
- Receiver Buffer Time-out Interrupt (RXTOINT)
- Buffer Error Interrupt (BUFERRINT)

- TX Overflow Error Interrupt Flag (TXOVIF)
- RX Overflow Error Interrupt Flag (RXOVIF)

- LIN Bus Interrupt (LININT)
  - LIN Break Detection Flag (BRKDETF)
  - Bit Error Detect Status Flag (BITEF)
  - LIN Slave ID Parity Error Flag (SLVIDPEF)
  - LIN Slave Header Error Flag (SLVHEF)
  - LIN Slave Header Detection Flag (SLVHDETF)
- Wake-up Interrupt (WKINT)
  - nCTS Wake-up Flag (CTSWKF)
  - Incoming Data Wake-up Flag (DATWKF)
  - Received Data FIFO Reached Threshold Wake-up Flag (RFRTWKF)
  - RS-485 Address Match (AAD mode) Wake-up Flag (RS485WKF)
  - Received Data FIFO Threshold Time-out Wake-up Flag (TOUTWKF)
- Auto-Baud Rate Interrupt (ABRINT)
  - Auto-baud Rate Detect Interrupt Flag (ABRDIF)
  - Auto-baud Rate Detect Time-out Interrupt Flag (ABRDTOIF)
- Single-wire Bit Error Detect Interrupt (SWBEINT)

Table 6.23-9 describes the interrupt sources and flags. The interrupt is generated when the interrupt flag is generated and the interrupt enable bit is set. User must clear the interrupt flag after the interrupt is generated.

Interrupt Source	Interrupt Indicator	Interrupt Enable Bit	Interrupt Flag	Flag Caused By	Flag Cleared By
Receive Data Available Interrupt	RDAINT	RDAIEN	RDAIF	N/A	Read UART_DAT
Transmit Holding Register Empty Interrupt	THREINT	THREIEN	THREIF	N/A	Write UART_DAT
Transmitter Empty Interrupt	TXENDINT	TXENDIEN	TXENDIF	N/A	Write UART_DAT
Receive Line Status Interrupt	RLSINT	RLSIEN	RLSIF	RLSIF = BIF	Write '1' to BIF
				RLSIF = FEF	Write '1' to FEF
				RLSIF = PEF	Write '1' to PEF
				RLSIF = ADDRDETF	Write '1' to ADDRDETF
Modem Status Interrupt	MODEMINT	MODEMIEN	MODEMIF	MODEMIF = CTSDETF	Write '1' to CTSDETF
Receiver Buffer Time-out Interrupt	RXTOINT	RXTOIEN	RXTOIF	N/A	Read UART_DAT
Buffer Error Interrupt	BUFERRINT	BUFERRIEN	BUFERRIF	BUFERRIF = TXOVIF	Write '1' to TXOVIF
				BUFERRIF =	Write '1' to RXOVIF

				RXOVIF	
LIN Bus Interrupt	LININT	LINIEN	LINIF	LINIF = BRKDETF	Write '1' to LINIF and Write '1' to BRKDETF
				LINIF = BITEF	Write '1' to LINIF and Write '1' to BITEF
				LINIF = SLVIDPEF	Write '1' to LINIF and Write '1' to SLVIDPEF
				LINIF = SLVHEF	Write '1' to LINIF and Write '1' to SLVHEF
				LINIF = SLVHDETF	Write '1' to LINIF and Write '1' to SLVHDETF
Wake-up Interrupt	WKINT	WKIEN	WKIF	WKIF = CTSWKF	Write '1' to CTSWKF
				WKIF = DATWKF	Write '1' to DATWKF
				WKIF = RFRTWKF	Write '1' to RFRTWKF
				WKIF = RS485WKF	Write '1' to RS485WKF
				WKIF = TOUTWKF	Write '1' to TOUTWKF
Auto-Baud Rate Interrupt	ABRINT	ABRIEN	ABRIF	ABRIF = ABRDIF	Write '1' to ABRDIF
				ABRIF = ABRDTOIF	Write '1' to ABRDTOIF
Single-wire Bit Error Detect Interrupt	SWBEINT	SWBEIEN	SWBEIF	N/A	Writing '1' to SWBEIF

Table 6.23-9 UART Controller Interrupt Source and Flag List

6.23.5.8 UART Function Mode

The UART controller provides UART function (Setting FUNCSEL (UART_FUNCSEL [2:0]) to '000' to enable UART function mode). The UART baud rate is up to 10 Mbps.

The UART provides full-duplex and asynchronous communications. The transmitter and receiver contain 16 bytes FIFO for payloads. User can program receiver buffer trigger level and receiver buffer time-out detection for receiver. The transmitting data delay time between the last stop and the next START bit can be programed by setting DLY (UART_TOUT [15:8]) register. The UART supports hardware auto-flow control that provides programmable nRTS flow control trigger level. The number of data bytes in RX FIFO is equal to or greater than RTSTRGLV (UART_FIFO[19:16]), the nRTS is de-asserted.

UART Line Control Function

The UART controller supports fully programmable serial-interface characteristics by setting the UART_LINE register. User can program UART_LINE register for the word length, STOP bit and PARITY bit setting. Table 6.23-10 and Table 6.23-11 list the UART word, STOP bit length and the PARITY bit

settings.

NSB (UART_LINE[2])	WLS (UART_LINE[1:0])	Word Length (Bit)	Stop Length (Bit)
0	00	5	1
0	01	6	1
0	10	7	1
0	11	8	1
1	00	5	1.5
1	01	6	2
1	10	7	2
1	11	8	2

Table 6.23-10 UART Line Control of Word and Stop Length Setting

Parity Type	SPE (UART_LINE[5])	EPE (UART_LINE[4])	PSS (UART_LINE[7])	PBE (UART_LINE[3])	Description
No Parity	x	x	x	0	No PARITY bit output.
Parity source from UART_DAT	x	x	1	1	PARITY bit is generated and checked by software.
Odd Parity	0	0	0	1	Odd Parity is calculated by adding all the “1’s” in a data stream and adding a PARITY bit to the total bits, to make the total count an odd number.
Even Parity	0	1	0	1	Even Parity is calculated by adding all the “1’s” in a data stream and adding a PARITY bit to the total bits, to make the count an even number.
Forced Mask Parity	1	0	0	1	PARITY bit always logic 1. PARITY bit on the serial byte is set to “1” regardless of total number of “1’s” (even or odd counts).
Forced Space Parity	1	1	0	1	PARITY bit always logic 0. PARITY bit on the serial byte is set to “0” regardless of total number of “1’s” (even or odd counts).

Table 6.23-11 UART Line Control of PARITY bit Setting

UART Auto-Flow Control Function

The UART supports auto-flow control function that uses two signals, nCTS (clear-to-send) and nRTS (request-to-send), to control the flow of data transfer between the UART and external devices (e.g. Modem). When auto-flow is enabled, the UART is not allowed to receive data until the UART asserts nRTS to external device. When the number of bytes stored in the RX FIFO equals the value of RTSTRGLV (UART_FIFO [19:16]), the nRTS is de-asserted. The UART sends data out when UART detects nCTS is asserted from external device. If the valid asserted nCTS is not detected, the UART will not send data out. The auto flow control block diagram is shown in Figure 6.23-11.

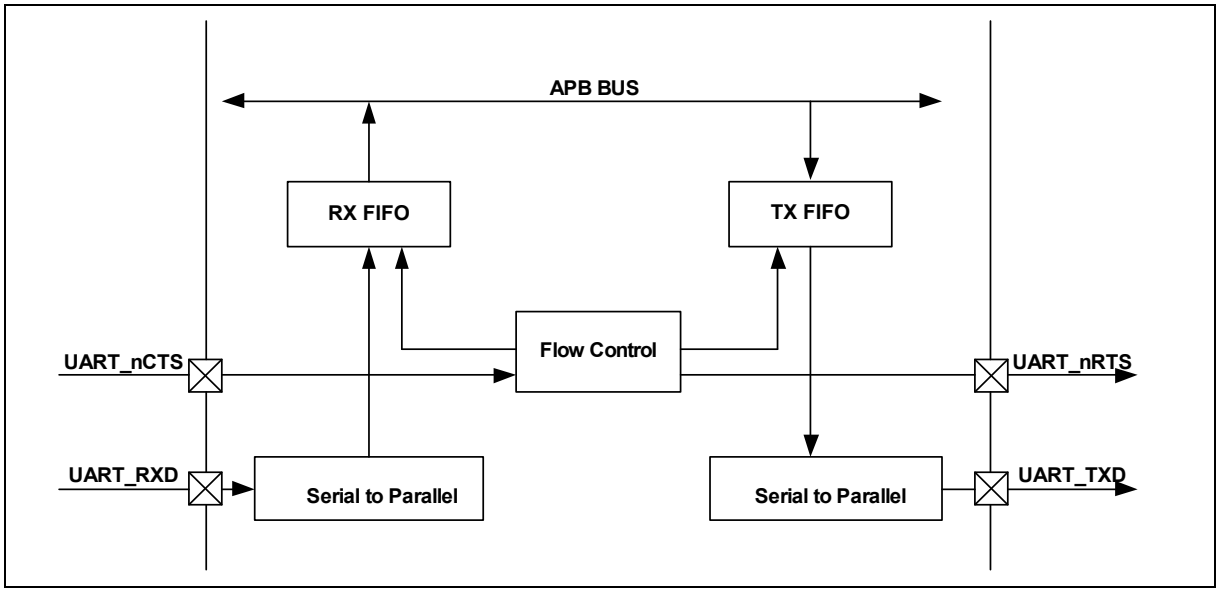


Figure 6.23-11 Auto-Flow Control Block Diagram

Figure 6.23-12 demonstrates the nCTS auto-flow control of UART function mode. User must set ATOCTSEN (UART_INTEN [13]) to enable nCTS auto-flow control function. The CTSACTLV (UART_MODEMSTS [8]) can set nCTS pin input active state. The CTSDETF (UART_MODEMSTS[0]) is set when any state change of nCTS pin input has occurred, and then TX data will be automatically transmitted from TX FIFO.

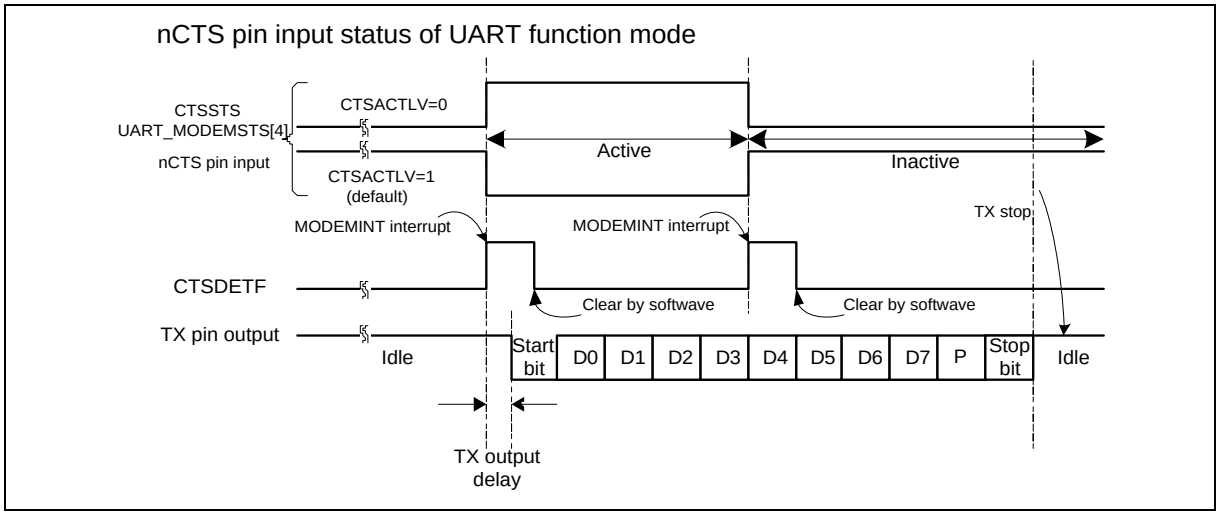


Figure 6.23-12 UART nCTS Auto-Flow Control Enabled

As shown in Figure 6.23-13, in UART nRTS auto-flow control mode (ATORTSEN(UART_INTEN[12])=1), the nRTS internal signal is controlled by UART FIFO controller with RTSTRGLV(UART_FIFO[19:16]) trigger level.

Setting RTSACTLV(UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from nRTS signal. User can read the RTSSTS (UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

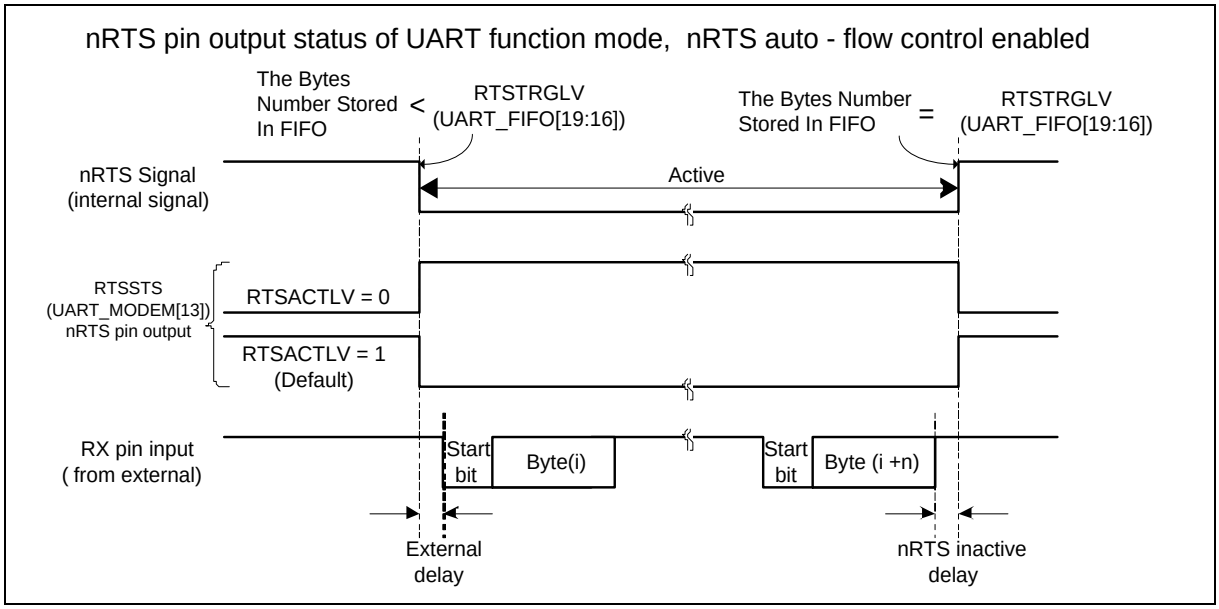


Figure 6.23-13 UART nRTS Auto-Flow Control Enabled

As shown in Figure 6.23-14, in software mode (ATORTSEN(UART_INTEN[12])=0), the nRTS flow is directly controlled by software programming of RTS(UART_MODEM[1]) control bit.

Setting RTSACTLV(UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UART_MODEM[1]) control bit. User can read the RTSSTS(UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

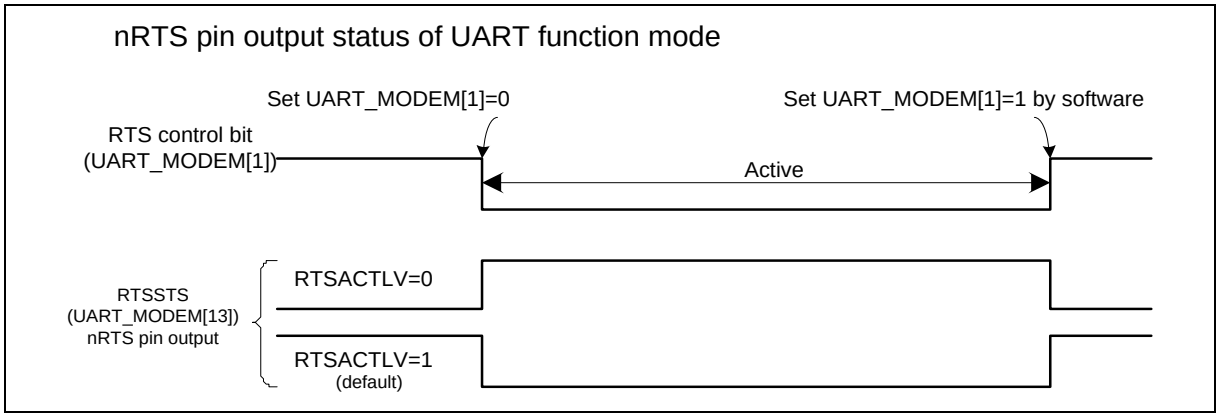


Figure 6.23-14 UART nRTS Auto-Flow with Software Control

6.23.5.9 IrDA Function Mode

The UART controller also provides Serial IrDA (SIR, Serial Infrared) function (Setting FUNCSEL (UART_FUNCSEL[2:0]) to '010' to enable the IrDA function). The SIR specification defines a short-range infrared asynchronous serial transmission mode with one START bit, 8 data bits, and 1 STOP bit. The maximum data rate is 115.2 kbps. The IrDA SIR block contains an IrDA SIR protocol encoder/decoder. The IrDA SIR protocol is half-duplex only. So, it cannot transmit and receive data at the same time. The IrDA SIR physical layer specifies a minimum 10 ms transfer delay between transmission and reception, and this delay feature must be implemented by software.

In IrDA mode, the BAUDM1 (UART_BAUD [29]) must be cleared.

Baud Rate = Clock / (16 * (BRD +2)), where BRD (UART_BAUD[15:0]) is Baud Rate Divider in UART_BAUD register.

Note: The tolerance of baud-rate is ±5% between IrDA master and IrDA slave.

The IrDA control block diagram is shown in Figure 6.23-15.

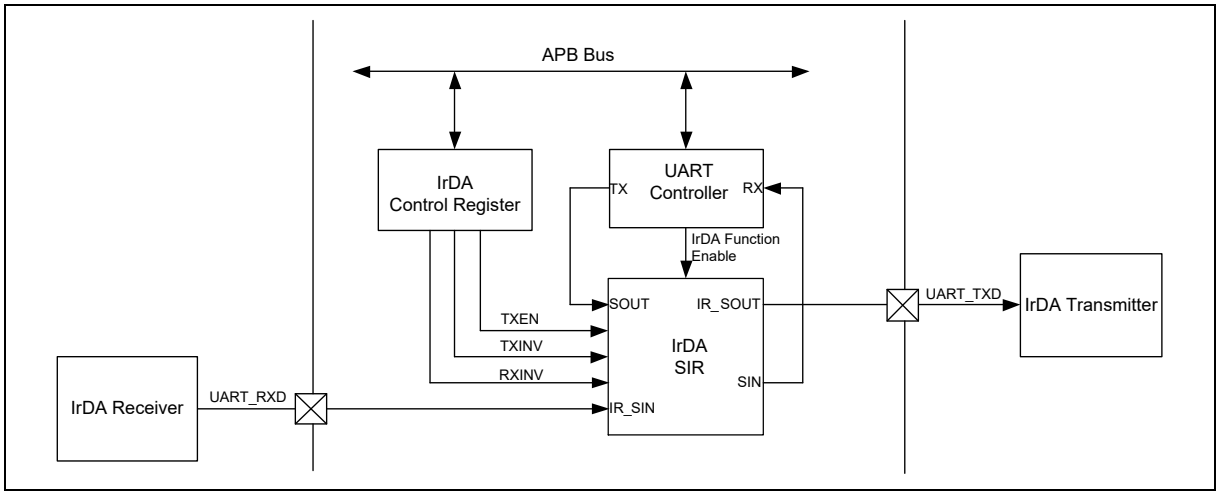


Figure 6.23-15 IrDA Control Block Diagram

IrDA SIR Transmit Encoder

The IrDA SIR Transmit Encoder modulates Non-Return-to-Zero (NRZ) transmit bit stream output from UART. The IrDA SIR physical layer specifies the use of Return-to-Zero, Inverted (RZI) modulation scheme which represents logic 0 as an infra light pulse. The modulated output pulse stream is transmitted to an external output driver and infrared light emitting diode.

The transmitted pulse width is specified as 3/16 period of baud rate.

IrDA SIR Receive Decoder

The IrDA SIR Receive Decoder demodulates the Return-to-Zero bit stream from the input detector and outputs the NRZ serial bits stream to the UART received data input.

In idle state, the decoder input is high. A START bit is detected when the decoder input is LOW. In normal operation, the RXINV (UART_IRDA[6]) is set to ‘1’ and TXINV (UART_IRDA[5]) is set to ‘0’.

IrDA SIR Operation

The IrDA SIR encoder/decoder provides functionality which converts between UART data stream and half-duplex serial SIR interface. Figure 6.23-16 is IrDA encoder/decoder waveform.

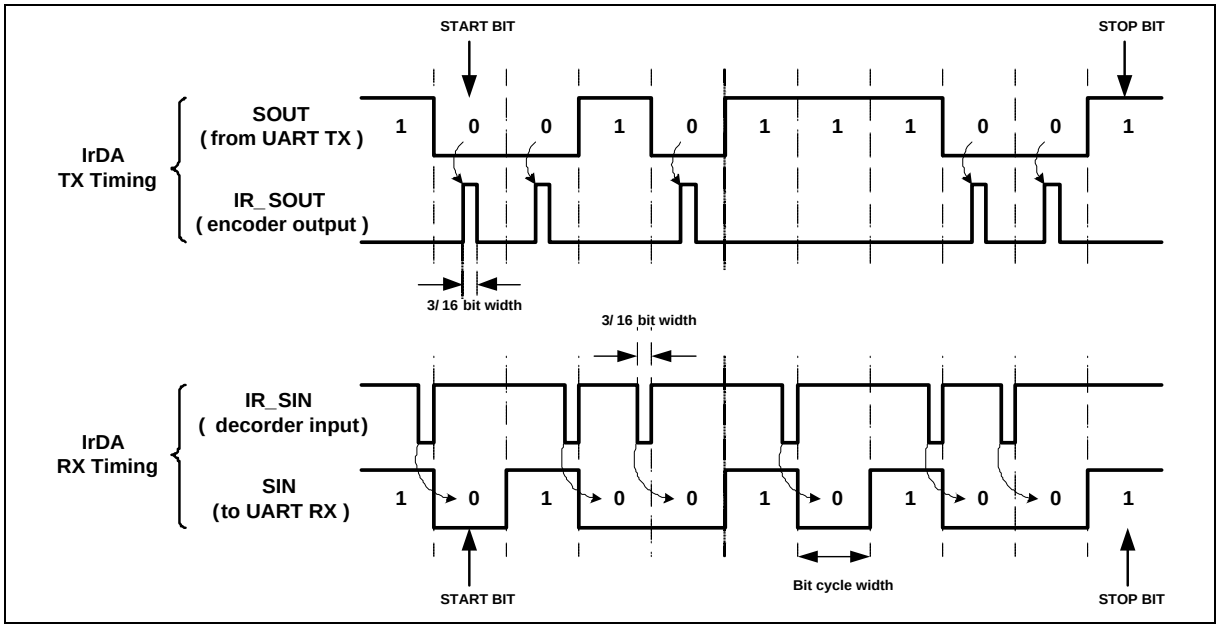


Figure 6.23-16 IrDA TX/RX Timing Diagram

6.23.5.10 LIN Function Mode (Local Interconnection Network)

The UART Controller supports LIN function. Setting FUNCSEL (UART_FUNCSEL[2:0]) to '001' to select LIN mode operation. The UART Controller supports LIN break/delimiter generation and break/delimiter detection in LIN master mode, and supports header detection and automatic resynchronization in LIN Slave mode.

Structure of LIN Frame

According to the LIN protocol, all information transmitted is packed as frames; a frame consists of a header (provided by the master task) and a response (provided by a slave task). The header (provided by the master task) consists of a break field and a sync field followed by a frame identifier (frame ID). The frame identifier uniquely defines the purpose of the frame. The slave task is appointed for providing the response associated with the frame ID. The response consists of a data field and a checksum field. Figure 6.23-17 is the structure of LIN Frame.

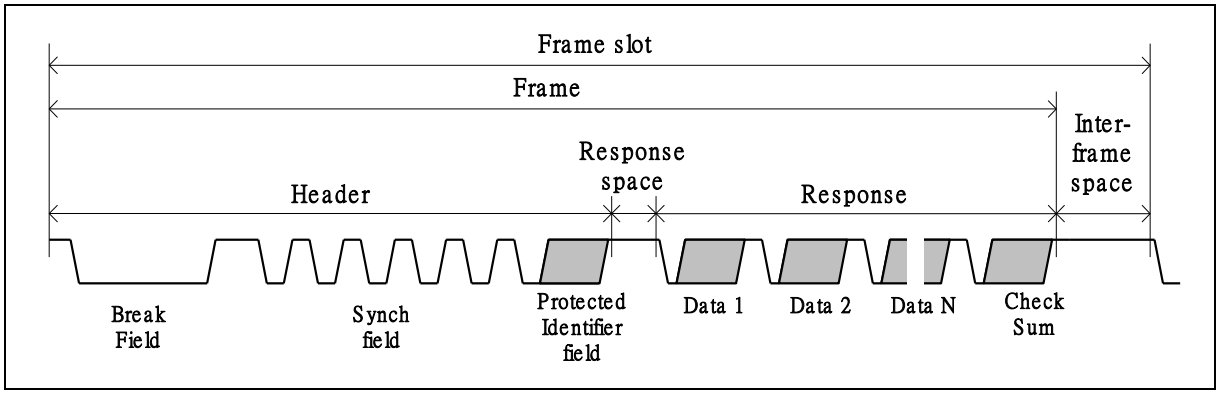


Figure 6.23-17 Structure of LIN Frame

Structure of LIN Byte

In LIN mode, each byte field is initiated by a START bit with value 0 (dominant), followed by 8 data bits and no PARITY bit, LSB is first and ended by 1 STOP bit with value 1 (recessive) in accordance with the LIN standard. The structure of Byte is shown in Figure 6.23-18.

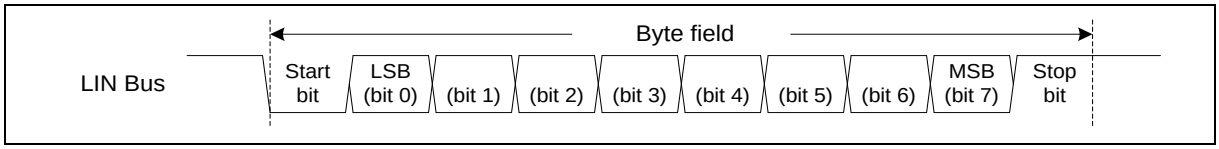


Figure 6.23-18 Structure of LIN Byte

LIN Master Mode

The UART Controller supports LIN Master mode. To enable and initialize the LIN Master mode, the following steps are necessary:

1. Set the UART_BAUD register to select the desired baud rate.
2. Set WLS (UART_LINE[1:0]) to '11' to configure the word length with 8 bits, clearing PBE (UART_LINE[3]) bit to disable parity check and clearing NSB (UART_LINE[2]) bit to configure with one STOP bit.
3. Set FUNCSEL (UART_FUNCSEL[2:0]) to '001' to select LIN function mode operation.

A complete header consists of a break field and sync field followed by a frame identifier (frame ID). The UART controller can be selected header sending by three header selected modes. The header selected mode can be "break field" or "break field and sync field" or "break field, sync field and frame ID field" by setting HSEL (UART_LINCTL[23:22]). If the selected header is "break field", software must handle the following sequence to send a complete header to bus by filling sync data (0x55) and frame ID data to the UART_DAT register. If the selected header is "break field and sync field", software must handle the sequence to send a complete header to bus by filling the frame ID data to UART_DAT register, and if the selected header is "break field, sync field and frame ID field", hardware will control the header sending sequence automatically but software must filled frame ID data to PID (UART_LINCTL [31:24]). When operating in header selected mode in which the selected header is "break field, sync field and frame ID field", the frame ID PARITY bit can be calculated by software or hardware depending whether the IDPEN (UART_LINCTL[9]) bit is set or not.

HSEL	Break Field	Sync Field	ID Field
0	Generated by Hardware	Handled by Software	Handled by Software
1	Generated by Hardware	Generated by Hardware	Handled by Software
2	Generated by Hardware	Generated by Hardware	Generated by Hardware (But Software needs to fill ID to PID (UART_LINCTL[31:24]) first)

Table 6.23-12 LIN Header Selection in Master Mode

When UART is operated in LIN data transmission, LIN bus transfer state can be monitored by hardware or software. User can enable hardware monitoring by setting BITERREN (UART_LINCTL [12]) to "1", if the input pin (UART_RX) state is not equal to the output pin (UART_TX) state in LIN transmitter state that hardware will generate an interrupt to CPU. Software can also monitor the LIN bus transfer state by checking the read back data in UART_DAT register. The following sequence is a program sequence example.

The procedure without software error monitoring in Master mode:

1. Fill Protected Identifier to PID (UART_LINCTL[31:24]).
2. Select the hardware transmission header field including "break field + sync field + protected identifier field" by setting HSEL (UART_LINCTL [23:22]) to "10".
3. Set SENDH (UART_LINCTL[8]) bit to 1 for requesting header transmission.
4. Wait until SENDH (UART_LINCTL[8]) bit cleared by hardware.

5. Wait until TXEMPTYF (UART_FIFOSTS[28]) set to 1 by hardware.

Note 1: The default setting of break field is 12 dominant bits (break field) and 1 recessive bit break/sync delimiter. Setting BRKFL (UART_LINCTL [19:16]) and BSL (UART_LINCTL[21:20]) to change the LIN break field length and break/sync delimiter length.

Note 2: The default setting of break/sync delimiter length is 1-bit time and the inter-byte spaces default setting is also 1-bit time. Setting BSL (UART_LINCTL[21:20]) and DLY(UART_TOUT[15:8]) can change break/sync delimiter length and inter-byte spaces.

Note 3: If the header includes the "break field, sync field and frame ID field", software must fill frame ID to PID (UART_LINCTL[31:24]) before trigger header transmission (setting the SENDH (UART_LINCTL[8])). The frame ID parity can be generated by software or hardware depending on IDPEN (UART_LINCTL[9]) setting. If the parity generated by software with IDPEN (UART_LINCTL[9]) is set to '0', software must fill 8 bit data (include 2 bit parity) in this field. If the parity generated by hardware with IDPEN (UART_LINCTL[9]) is set to '1', software fills ID0~ID5 and hardware calculates P0 and P1.

Procedure with software error monitoring in Master mode:

1. Choose the hardware transmission header field to only include "break field" by setting HSEL (UART_LINCTL [23:22])] to '00'.
2. Enable break detection function by setting BRKDETEN (UART_LINCTL[10]).
3. Request break + break/sync delimiter transmission by setting the SENDH (UART_LINCTL[8]).
4. Wait until the BRKDETF (UART_LINSTS[8]) flag is set to "1" by hardware.
5. Request sync field transmission by writing 0x55 into UART_DAT register.
6. Wait until the RDAIF (UART_INTSTS[0]) is set to "1" by hardware and then read back the UART_DAT register.
7. Request header frame ID transmission by writing the protected identifier value to UART_DAT register.
8. Wait until the RDAIF (UART_INTSTS[0]) is set to "1" by hardware and then read back the UART_DAT register.

LIN Break and Delimiter Detection

When software enables the break detection function by setting BRKDETEN (UART_LINCTL[10]), the break detection circuit is activated. The break detection circuit is totally independent from the UART receiver.

When the break detection function is enabled, the circuit looks at the input UART_RX pin for a start signal. If UART LIN controller detects consecutive dominant is greater than 11 bits dominant followed by a recessive bit (delimiter), the BRKDETF (UART_LINSTS[8]) flag is set at the end of break field. If the LINIEN (UART_INTEN[8]) bit is set to 1, an interrupt LININT (UART_INTSTS[15]) will be generated. The behavior of the break detection and break flag are shown in Figure 6.23-19.

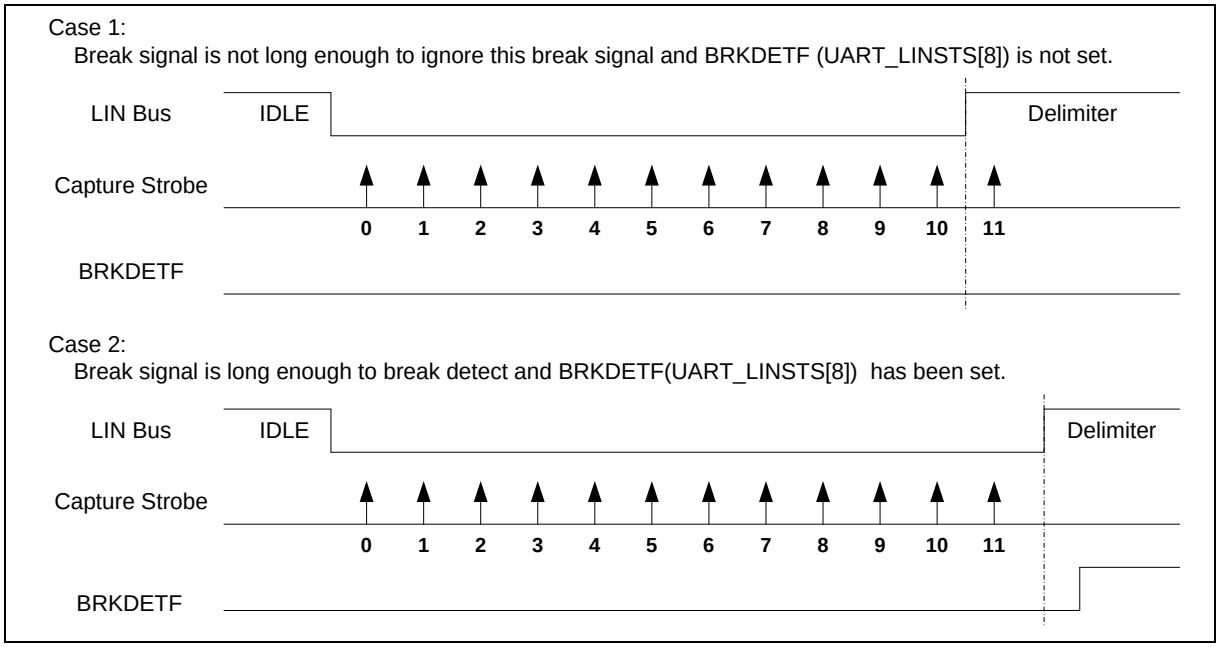


Figure 6.23-19 Break Detection in LIN Mode

LIN Frame ID and Parity Format

The LIN frame ID value in LIN function mode is shown, the frame ID parity can be generated by software or hardware depends on IDPEN (UART_LINCTL[9]).

If the parity generated by hardware (IDPEN (UART_LINCTL[9])=1), user fill ID0~ID5 (UART_LINCTL [29:24]) hardware will calculate P0 (UART_LINCTL[30]) and P1 (UART_LINCTL[31]) otherwise user must filled frame ID and parity in this field.

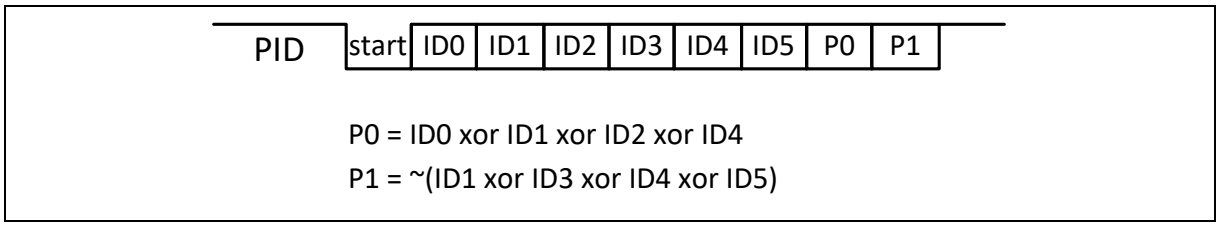


Figure 6.23-20 LIN Frame ID and Parity Format

LIN Slave Mode

The UART Controller supports LIN Slave mode. To enable and initialize the LIN Slave mode, the following steps are necessary:

1. Set the UART_BAUD register to select the desired baud rate.
2. Configure the data length to 8 bits by setting WLS (UART_LINE[1:0]) to '11' and disable parity check by clearing PBE (UART_LINE[3]) bit and configure with one STOP bit by clearing NSB (UART_LINE[2]) bit.
3. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[2:0]) to '001'.
4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) to 1.

LIN Header Reception

According to the LIN protocol, a slave node must wait for a valid header which comes from the master node. Next the slave task will take one of following actions (depend on the master header frame ID value).

- Receive the response.
- Transmit the response.
- Ignore the response and wait for next header.

In LIN Slave mode, user can enable the slave header detection function by setting the SLVHDEN (UART_LINCTL[1]) to detect complete frame header (receive “break field”, “sync field” and “frame ID field”). When a LIN header is received, the SLVHDETF (UART_LINSTS[0]) flag will be set. If the LINIEN (UART_INTEN[8]) bit is set to 1, an interrupt will be generated. User can enable the frame ID parity check function by setting IDPEN (UART_LINCTL[9]). If only received frame ID parity is not correct (break and sync filed are correct), the SLVIDPEF (UART_LINSTS[2]) flag is set to ‘1’. If the LINIEN(UART_INTEN[8]) is set to 1, an interrupt will be generated and SLVHDETF (UART_LINSTS[0]) is set to ‘1’. User can also put LIN in mute mode by setting MUTE (UART_LINCTL[4]) to ‘1’. This mode allows detection of headers only (break + sync + frame ID) and prevents the reception of any other characters. In order to avoid bit rate tolerance, the controller supports automatic resynchronization function to avoid clock deviation error, user can enable this feature by setting SLVAREN (UART_LINCTL[2]).

LIN Response Transmission

The LIN slave node can transmit response and receive response. When slave node is the publisher of the response, the slave node sends response by filling data to the UART_DAT register. If the slave node is the subscriber of the response, the slave node receives data from LIN bus.

LIN Header Time-out Error

The LIN slave controller contains a header time-out counter. If the entire header is not received within the maximum time limit of 57 bit times, the header error flag SLVHEF (UART_LINSTS [1]) will be set. The time-out counter is enabled at each break detect edge and stopped in the following conditions.

- A LIN frame ID field has been received.
- The header error flag asserts.
- Writing 1 to the SLVSYNCF (UART_LINSTS[3]) to re-search a new frame header.

Mute Mode and LIN Exit from Mute Mode Condition

In Mute mode, a LIN slave node will not receive any data until specified condition occurred. It allows header detection only and prevents the reception of any other characters. User can enable Mute mode by setting the MUTE (UART_LINCTL[4]) and exiting from Mute mode condition can be selected by HSEL (UART_LINCTL[23:22]).

Note: It is recommended to set LIN slave node to Mute mode after checksum transmission.

The LIN slave controller exiting from Mute mode is described as follows: If HSEL (UART_LINCTL[23:22]) is set to “break field”, when LIN slave controller detects a valid LIN break and delimiter, the controller will enable the receiver (exit from Mute mode) and subsequent data (sync data, frame ID data, response data) are received in RX FIFO.

If HSEL (UART_LINCTL[23:22]) is set to “break field and sync field”, when the LIN slave controller detects a valid LIN break and delimiter followed by a valid sync field without frame error, the controller will enable the receiver (exit from mute mode) and subsequent data(ID data, response data) are received in RX FIFO. If HSEL (UART_LINCTL[23:22]) is set to “break field, sync field and ID field”, when the LIN slave controller detects a valid LIN break and delimiter and valid sync field without frame error followed by ID data without frame error and received ID data matched PID (UART_LINCTL[31:24]) value. The controller will enable the receiver (exit from mute mode) and subsequent data (response data) are received in RX FIFO.

Slave Mode Non-automatic Resynchronization (NAR)

User can disable the automatic resynchronization function to fix the communication baud rate. When operating in Non-Automatic Resynchronization mode, software needs some initial process, and the

initialization process flow of Non-Automatic Resynchronization mode is shown as follows:

1. Select the desired baud rate by setting the UART_BAUD register.
2. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[2:0]) to '001'.
3. Disable automatic resynchronization function by setting SLVAREN (UART_LINCTL[2]) is set to 0.
4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) is set to 1.

Slave Mode with Automatic Resynchronization (AR)

In Automatic Resynchronization (AR) mode, the controller will adjust the baud rate generator after each sync field reception. The initialization process flow of Automatic Resynchronization mode is shown as follows:

1. Select the desired baud rate by setting the UART_BAUD register.
2. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[2:0]) to '001'.
3. Enable automatic resynchronization function by setting SLVAREN (UART_LINCTL[2]) to '1'.
4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) is set to '1'.

When the automatic resynchronization function is enabled, after each LIN break field, the time duration between five falling edges is sampled on peripheral clock and the result of this measurement is stored in an internal 13-bit register and the UART_BAUD register value will be automatically updated at the end of the fifth falling edge. If the measure timer (13-bit) overflows before five falling edges, then the header error flag SLVHEF (UART_LINSTS [1]) will be set.

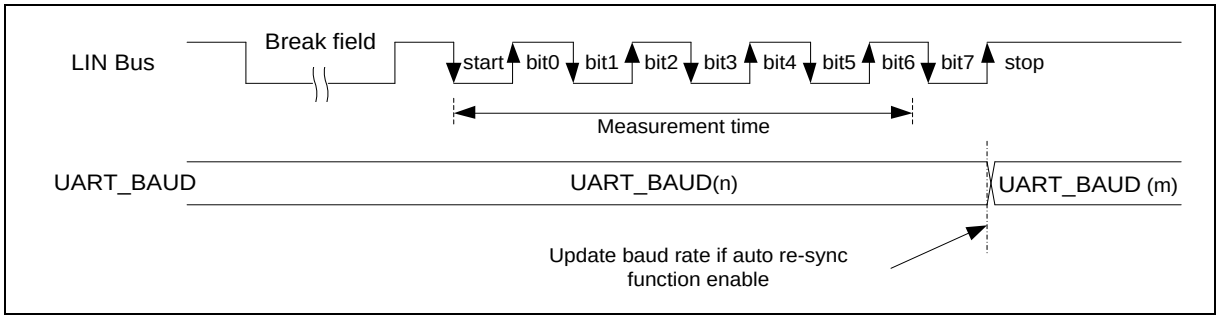


Figure 6.23-21 LIN Sync Field Measurement

When operating in Automatic Resynchronization (AR) mode, software must select the desired baud rate by setting the UART_BAUD register and hardware will store it at internal TEMP_REG register, after each LIN break field, the time duration between five falling edges is sampled on peripheral clock and the result of this measurement is stored in an internal 13-bit register BAUD_LIN and the result will be updated to UART_BAUD register automatically.

To guarantee the transmission baud rate, the baud rate generator must reload the initial value before each new break reception. The initial value is programmed by the application during initialization (TEMP_REG). User can set SLVDUEN (UART_LINCTL [3]) to enable auto reload initial baud rate value function. If the SLVDUEN (UART_LINCTL [3]) is set, when received the next character, hardware will auto reload the initial value to UART_BAUD, and when the UART_BAUD be updated, the SLVDUEN (UART_LINCTL [3]) will be cleared automatically. The behavior of LIN updated method as shown in Figure 6.23-22.

Note 1: It is recommended to set the SLVDUEN bit before every checksum reception.

Note 2: When a header error is detected, user must write 1 to SLVSYNCF (UART_LINSTS[3]) to re-search new frame header. When writing 1 to it, hardware will reload the initial baud rate TEMP_REG and re-search new frame header.

Note 3: When operating in Automatic Resynchronization mode, the baud rate setting must be operated at mode2 (BAUDM1 (UART_BAUD [29]) and BAUDM0 (UART_BAUD[28]) must be 1).

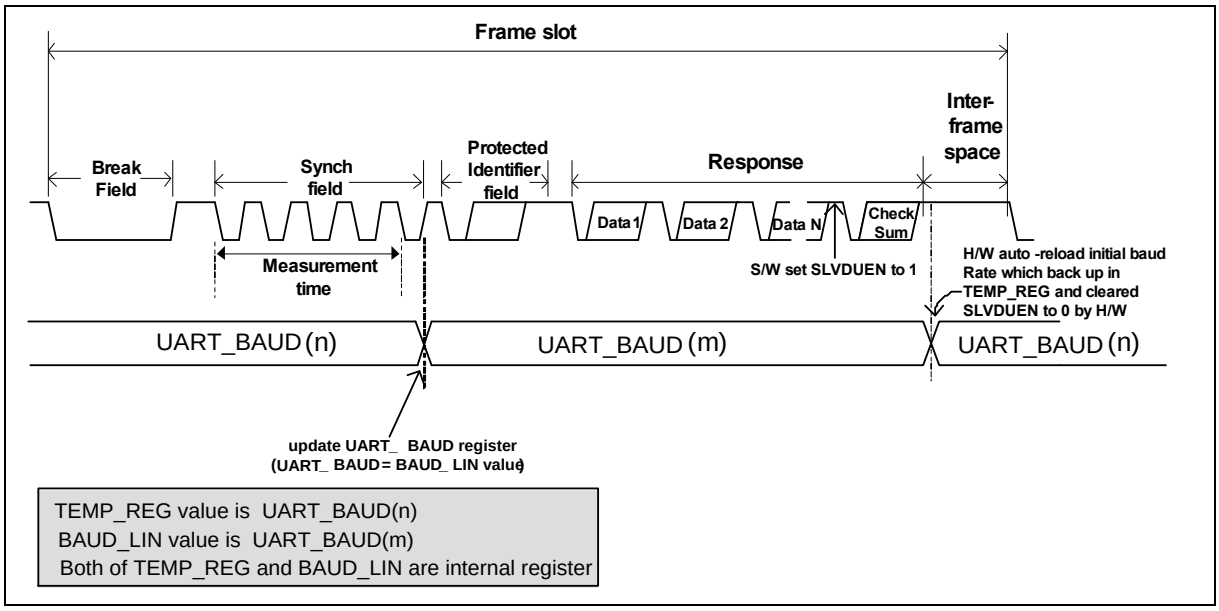


Figure 6.23-22 UART_BAUD Update Sequence in AR mode if SLVDUEN is 1

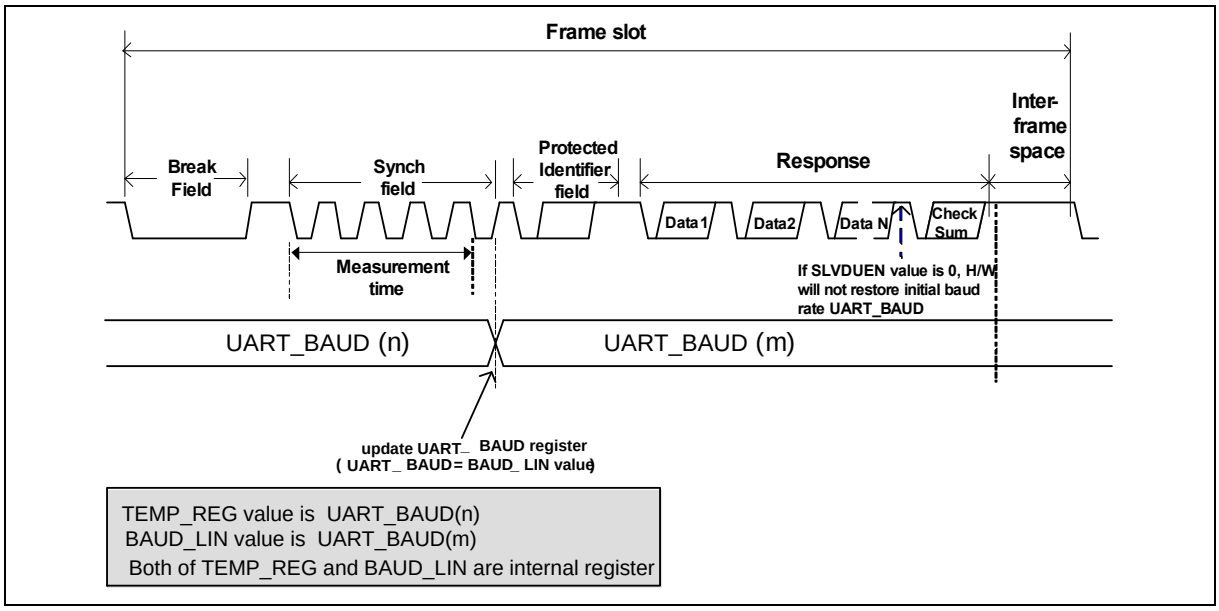


Figure 6.23-23 UART_BAUD Update Sequence in AR mode if SLVDUEN is 0

Deviation Error on the Sync Field

When operating in Automatic Resynchronization mode, the controller will check the deviation error on the sync field. The deviation error is checked by comparing the current baud rate with the received sync field. Two checks are performed in parallel.

Check1: Based on measurement between the first falling edge and the last falling edge of the sync field.

- If the difference is more than 14.84%, the header error flag SLVHEF (UART_LINSTS[1]) will be set.
- If the difference is less than 14.06%, the header error flag SLVHEF (UART_LINSTS[1])

will not be set.

- If the difference is between 14.84% and 14.06%, the header error flag SLVHEF (UART_LINSTS[1]) may either set or not.

Check2: Based on measurement of time between each falling edge of the sync field.

- If the difference is more than 18.75%, the header error flag SLVHEF (UART_LINSTS[1]) will be set.
- If the difference is less than 15.62%, the header error flag SLVHEF (UART_LINSTS[1]) will not be set.
- If the difference is between 18.75% and 15.62%, the header error flag SLVHEF (UART_LINSTS[1]) may either set or not.

Note: The deviation check is based on the current baud rate clock. Therefore, in order to guarantee correct deviation checking, the baud rate must reload the nominal value before each new break reception by setting SLVDUEN (UART_LINCTL[3]) register (It is recommend setting the SLVDUEN (UART_LINCTL[3]) bit before every checksum reception).

LIN Header Error Detection

In LIN Slave function mode, when user enables the header detection function by setting the SLVHDEN (UART_LINCTL[1]), hardware will handle the header detect flow. If the header has an error, the LIN header error flag SLVHEF (UART_LINSTS[1]) will be set and an interrupt is generated if the LINIEN (UART_INTEN[8]) bit is set. When header error is detected, user must reset the detect circuit to re-search a new frame header by writing 1 to SLVSYNCF (UART_LINSTS[3]) to re-search a new frame header.

The LIN header error flag SLVHEF (UART_LINSTS[1]) is set if one of the following conditions occurs:

- Break Delimiter is too short (less than 0.5-bit time).
- Frame error in sync field or Identifier field.
- The sync field data is not 0x55 (Non-Automatic Resynchronization mode).
- The sync field deviation error (With Automatic Resynchronization mode).
- The sync field measure time-out (With Automatic Resynchronization mode).
- LIN header reception time-out.

6.23.5.11 RS-485 Function Mode

Another alternate function of UART controller is RS-485 function (user must set FUNCSEL (UART_FUNCSEL[2:0]) to '011' to enable RS-485 function), and direction control provided by nRTS pin from an asynchronous serial port. The RS-485 transceiver control is implemented by using the nRTS control signal to enable the RS-485 driver. Many characteristics of the RX and TX are the same as UART in RS-485 mode.

The UART controller can be configured as a RS-485 addressable slave or a RS-485 master. RS-485 master transmitter will identify an address character by setting the parity (9^{th} bit) to 1. For data characters, the parity is set to 0. Software can use UART_LINE register to control the 9^{th} bit (When the PBE, EPE and SPE are set, the 9^{th} bit is transmitted 0 and when PBE and SPE are set and EPE is cleared, the 9^{th} bit is transmitted 1).

The controller supports three operation modes: RS-485 Normal Multidrop Operation Mode (NMM), RS-485 Auto Address Detection Operation Mode (AAD) and RS-485 Auto Direction Control Operation Mode (AUD). Software can choose any operation mode by programming the UART_ALTCTL register, and drive the transfer delay time between the last STOP bit leaving the TX FIFO and the de-assertion of by setting DLY (UART_TOUT [15:8]) register.

RS-485 Normal Multidrop Operation Mode (NMM)

In RS-485 Normal Multidrop Operation Mode (RS485NMM (UART_ALTCTL[8]) = 1), software must first decide the data which before the address byte be detected will be stored in RX FIFO or not. To ignore any data before address byte is detected, set RXOFF (UART_FIFO [8]) to 1 and enable RS485NMM (UART_ALTCTL [8]). The receiver will ignore any data until an address byte is detected (bit 9 = 1) and store the detected address byte data in the RX FIFO. To receive any data before address byte is detected, set RXOFF (UART_FIFO [8]) to 0 and enable RS485NMM (UART_ALTCTL [8]).

If an address byte is detected (bit 9 = 1), it will generate an interrupt to CPU and RXOFF (UART_FIFO [8]) can decide whether accepting the following data bytes are stored in the RX FIFO. If software disables receiver by setting RXOFF (UART_FIFO [8]) register, when a next address byte is detected, the controller will clear the RXOFF (UART_FIFO [8]) bit and the address byte data will be stored in the RX FIFO.

RS-485 Auto Address Detection Operation Mode (AAD)

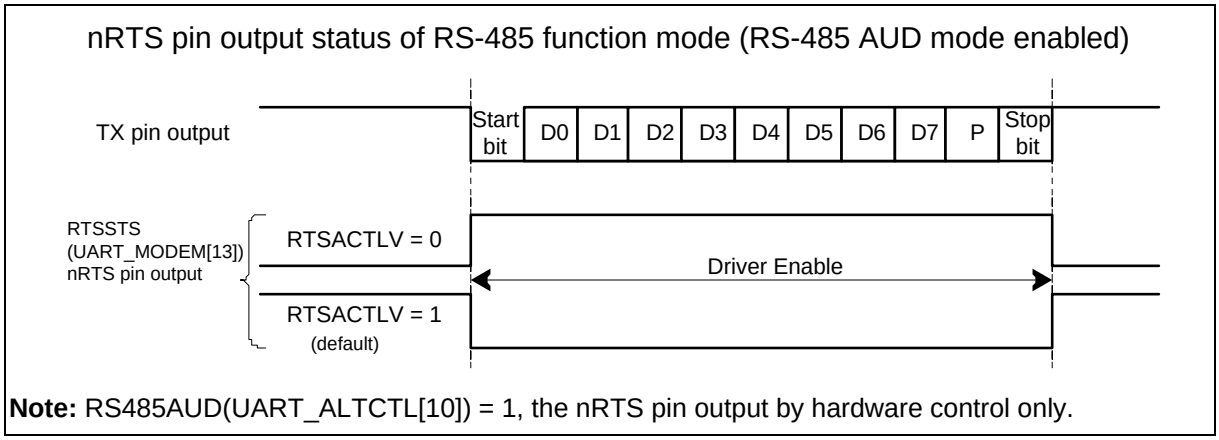
In RS-485 Auto Address Detection Operation Mode (RS485AAD (UART_ALTCTL[9]) = 1), the receiver will ignore any data until an address byte is detected (bit 9 = 1) and the address byte data matches the ADDRMV (UART_ALTCTL[31:24]) value. The address byte data will be stored in the RX FIFO. The all received byte data will be accepted and stored in the RX FIFO until an address byte data not match the ADDRMV (UART_ALTCTL[31:24]) value.

RS-485 Auto Direction Function (AUD)

RS-485 controllers support RS-485 auto direction control function (RS485AUD (UART_ALTCTL[10]) = 1). The RS-485 transceiver control is implemented by using the nRTS control signal from an asynchronous serial port. The nRTS line is connected to the RS-485 transceiver enable pin such that setting the nRTS line to high (logic 1) enables the RS-485 transceiver. Setting the nRTS line to low (logic 0) puts the transceiver into the tri-state condition. User can set RTSACTLV in UART_MODEM register to change the nRTS driving level.

Figure 6.23-24 demonstrates the RS-485 nRTS driving level in AUD mode. The nRTS pin will be automatically driven during TX data transmission.

Setting RTSACTLV(UART_MODEM[9]) can control nRTS pin output driving level. User can read the RTSSTS(UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

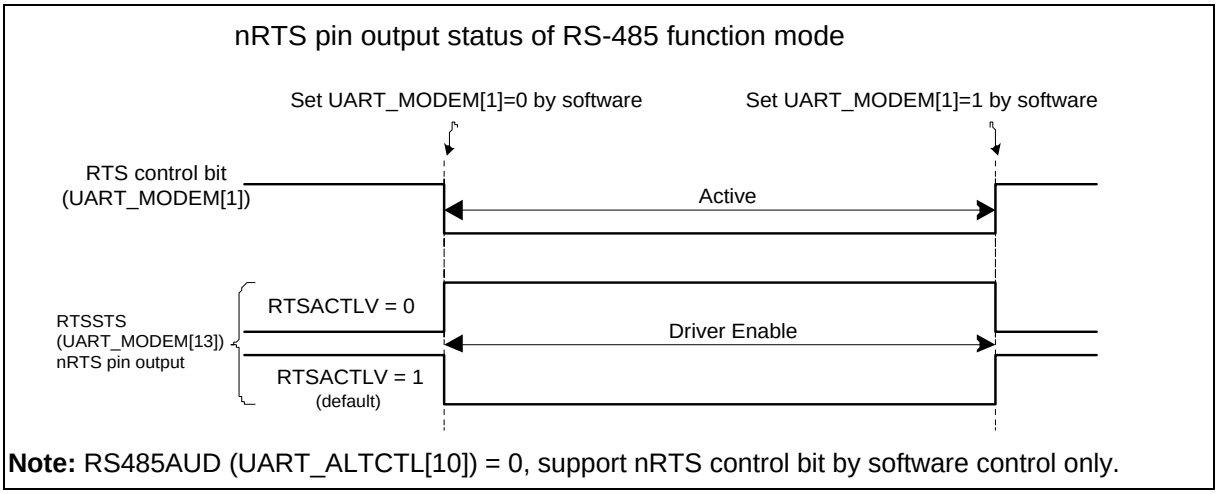


Note: RS485AUD(UART_ALTCTL[10]) = 1, the nRTS pin output by hardware control only.

Figure 6.23-24 RS-485 nRTS Driving Level in Auto Direction Mode

Figure 6.23-25 demonstrates the RS-485 nRTS driving level in software control (RS485AUD (UART_ALTCTL[10])=0). The nRTS driving level is controlled by programing the RTS(UART_MODEM[1]) control bit.

Setting RTSACTLV (UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UART_MODEM[1]) control bit. User can read the RTSSTS (UART_MODEM[13]) bit to get real nRTS pin output voltage logic status. The structure of RS-485 frame is shown in Figure 6.23-26.

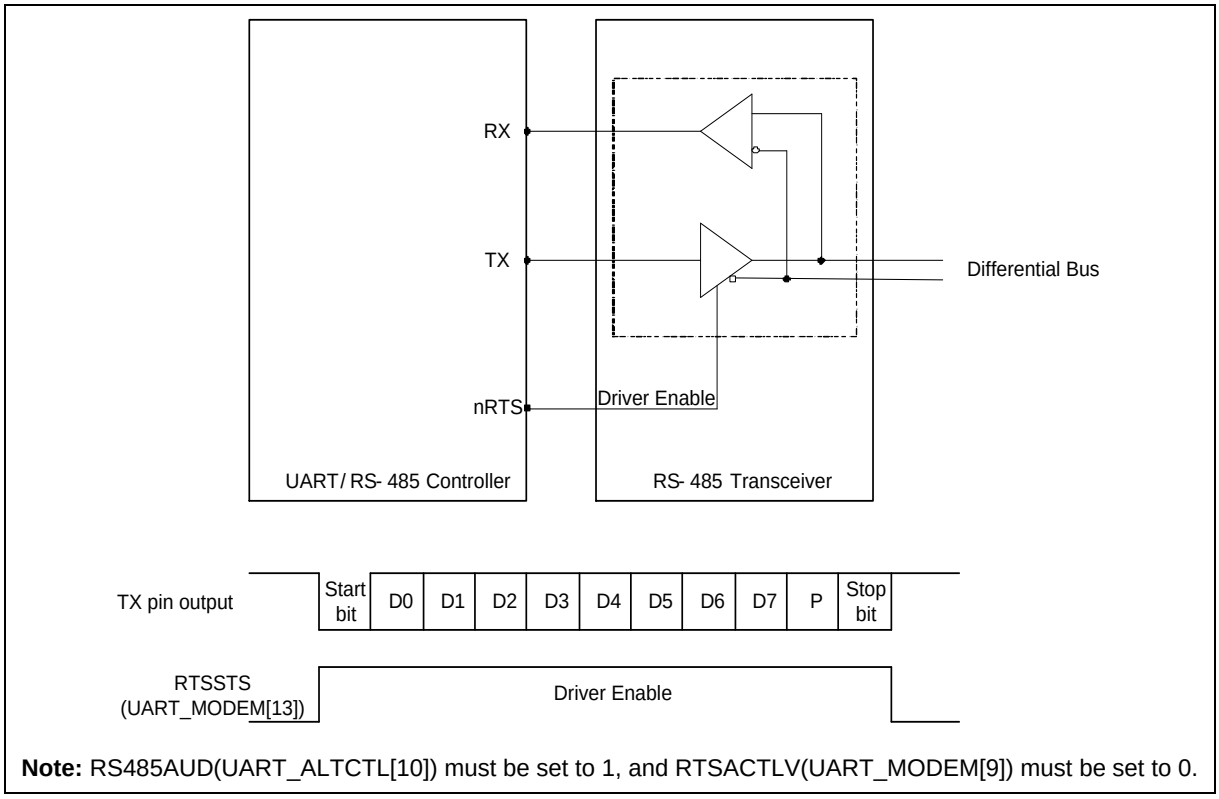


Note: RS485AUD (UART_ALTCTL[10]) = 0, support nRTS control bit by software control only.

Figure 6.23-25 RS-485 nRTS Driving Level with Software Control

Programming Sequence Example:

1. Program FUNCSEL in UART_FUNCSEL to select RS-485 function.
2. Program the RXOFF (UART_FIFO[8]) to determine enable or disable the receiver RS-485 receiver.
3. Program the RS485NMM (UART_ALTCTL[8]) or RS485AAD (UART_ALTCTL[9]) mode.
4. If the RS485AAD (UART_ALTCTL[9]) mode is selected, the ADDRMV (UART_ALTCTL[31:24]) is programmed for auto address match value.
5. Determine auto direction control by programming RS485AUD (UART_ALTCTL[10]).



Note: RS485AUD(UART_ALTCTL[10]) must be set to 1, and RTSACTLV(UART_MODEM[9]) must be set to 0.

Figure 6.23-26 Structure of RS-485 Frame

6.23.5.12 UART Single-wire Half Duplex

The UART controller provides single-wire half duplex function in UART function mode. Setting FUNCSEL (UART_FUNCSEL[2:0]) to '100' to enable the UART Single-wire function. The single-wire bus is idle (RXIDLE (UART_FIFOSTS[29]) = 1) when in RX state. Before writing data to transmit buffer (UART_DAT[7:0]), the bus state should be checked in idle (RXIDLE (UART_FIFOSTS[29]). By writing data to transmit buffer, the bus state transfers to TX state immediately. After the transmission, the bus state transfers from TX state to RX state.

The UART will not allowed to receive data when in single-wire half duplex function TX mode. If nRTS is asserted in TX mode, nRTS will make the docking UART device send out data and cause bus confliction. To reduce the bus confliction, the UART controller supports flow control function and bit error detection but does not support auto-flow control function. The nRTS is automatically inactivated in single-wire TX state. When in TX state, the UART controller will monitor bus state. If the bus state is not equal to TX state, the SWBEIF (UART_INTSTS[16]) is set.

6.23.5.13 PDMA Transfer Function

The UART controller supports PDMA transfer function.

By configuring PDMA parameter and set UART_DAT as the PDMA destination address. When TXPDMAEN (UART_INTEN[14]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

By configuring PDMA parameter and set UART_DAT as the PDMA source address. When RXPDMAEN (UART_INTEN[15]) is set to 1, the controller will start the PDMA reception process. UART controller will issue request to PDMA controller automatically when there is data in the RX FIFO buffer.

6.23.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
UART Base Address: **UARTx_BA = 0x4007_0000 + (0x1000 * x)** **x=0,1..9**				
UART_DAT **x=0,1..9**	UARTx_BA+0x00	R/W	UART Receive/Transmit Buffer Register	Undefined
UART_INTEN **x=0,1..9**	UARTx_BA+0x04	R/W	UART Interrupt Enable Register	0x0000_0000
UART_FIFO **x=0,1..9**	UARTx_BA+0x08	R/W	UART FIFO Control Register	0x0000_0101
UART_LINE **x=0,1..9**	UARTx_BA+0x0C	R/W	UART Line Control Register	0x0000_0000
UART_MODEM **x=0,1..9**	UARTx_BA+0x10	R/W	UART Modem Control Register	0x0000_0200
UART_MODEMSTS **x=0,1..9**	UARTx_BA+0x14	R/W	UART Modem Status Register	0x0000_0110
UART_FIFOSTS **x=0,1..9**	UARTx_BA+0x18	R/W	UART FIFO Status Register	0xB040_4000
UART_INTSTS **x=0,1..9**	UARTx_BA+0x1C	R/W	UART Interrupt Status Register	0x0040_0002
UART_TOUT **x=0,1..9**	UARTx_BA+0x20	R/W	UART Time-out Register	0x0000_0000
UART_BAUD **x=0,1..9**	UARTx_BA+0x24	R/W	UART Baud Rate Divider Register	0x0F00_0000
UART_IRDA **x=0,1..9**	UARTx_BA+0x28	R/W	UART IrDA Control Register	0x0000_0040
UART_ALTCTL **x=0,1..9**	UARTx_BA+0x2C	R/W	UART Alternate Control/Status Register	0x0000_000C
UART_FUNCSEL **x=0,1..9**	UARTx_BA+0x30	R/W	UART Function Select Register	0x0000_0000
UART_LINCTL **x=0,1**	UARTx_BA+0x34	R/W	UART LIN Control Register	0x000C_0000
UART_LINSTS **x=0,1**	UARTx_BA+0x38	R/W	UART LIN Status Register	0x0000_0000

UART_BRCOMP **x=0,1..9**	UARTx_BA+0x3C	R/W	UART Baud Rate Compensation Register	0x0000_0000
UART_WKCTL **x=0,1..9**	UARTx_BA+0x40	R/W	UART Wake-up Control Register	0x0000_0000
UART_WKSTS **x=0,1..9**	UARTx_BA+0x44	R/W	UART Wake-up Status Register	0x0000_0000
UART_DWKCOMP **x=0,1..9**	UARTx_BA+0x48	R/W	UART Incoming Data Wake-up Compensation Register	0x0000_0000
UART_RS485DD **x=0,1..9**	UARTx_BA+0x4C	R/W	UART RS485 Transceiver Deactivate Delay Register	0x0000_0000

6.23.7 Register Description

UART Receive/Transmit Buffer Register (UART_DAT)

Register	Offset	R/W	Description	Reset Value
UART_DAT x=0,1..9	UARTx_BA+0x00	R/W	UART Receive/Transmit Buffer Register	Undefined

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**PARITY**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DAT							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**PARITY**	**PARITY Bit Receive/Transmit Buffer** Write Operation: By writing to this bit, the PARITY bit will be stored in transmitter FIFO. If PBE (UART_LINE[3]) and PSS (UART_LINE[7]) are set, the UART controller will send out this bit follow the DAT (UART_DAT[7:0]) through the UART_TXD. Read Operation: If PBE (UART_LINE[3]) and PSS (UART_LINE[7]) are enabled, the PARITY bit can be read by this bit. **Note:** This bit has effect only when PBE (UART_LINE[3]) and PSS (UART_LINE[7]) are set.
[7:0]	**DAT**	**Data Receive/Transmit Buffer** Write Operation: By writing one byte to this register, the data byte will be stored in transmitter FIFO. The UART controller will send out the data stored in transmitter FIFO top location through the UART_TXD. Read Operation: By reading this register, the UART controller will return an 8-bit data received from receiver FIFO.

UART Interrupt Enable Register (UART_INTEN)

Register	Offset	R/W	Description	Reset Value
UART_INTEN **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x04	R/W	UART Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	TXENDIEN	Reserved			ABRIEN	Reserved	SWBEIEN
15	14	13	12	11	10	9	8
RXPDMAEN	TXPDMAEN	ATOCTSEN	ATORTSEN	TOCNTEN	Reserved		LINIEN
7	6	5	4	3	2	1	0
Reserved	WKIEN	BUFERRIEN	RXTOIEN	MODEMIEN	RLSIEN	THREIEN	RDAIEN

Bits	Description	
[31:23]	**Reserved**	Reserved.
[22]	**TXENDIEN**	**Transmitter Empty Interrupt Enable Bit** If TXENDIEN (UART_INTEN[22]) is enabled, the Transmitter Empty interrupt TXENDINT (UART_INTSTS[30]) will be generated when TXENDIF (UART_INTSTS[22]) is set (TX FIFO (UART_DAT) is empty and the STOP bit of the last byte has been transmitted). 0 = Transmitter empty interrupt Disabled. 1 = Transmitter empty interrupt Enabled.
[21:19]	**Reserved**	Reserved.
[18]	**ABRIEN**	**Auto-baud Rate Interrupt Enable Bit** 0 = Auto-baud rate interrupt Disabled. 1 = Auto-baud rate interrupt Enabled.
[17]	**Reserved**	Reserved.
[16]	**SWBEIEN**	**Single-wire Bit Error Detection Interrupt Enable Bit** Set this bit, the Single-wire Half Duplex Bit Error Detection Interrupt SWBEINT(UART_INTSTS[24]) is generated when Single-wire Bit Error Detection SWBEIF(UART_INTSTS[16]) is set. 0 = Single-wire Bit Error Detect Interrupt Disabled. 1 = Single-wire Bit Error Detect Interrupt Enabled. **Note:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is select UART Single-wire mode.
[15]	**RXPDMAEN**	**RX PDMA Enable Bit** This bit can enable or disable RX PDMA service. 0 = RX PDMA Disabled. 1 = RX PDMA Enabled. **Note:** If RLSIEN (UART_INTEN[2]) is enabled and HWRLSINT (UART_INTSTS[26]) is set to 1, the RLS (Receive Line Status) Interrupt is caused. If RLS interrupt is caused by Break Error Flag BIF(UART_FIFOSTS[6]), Frame Error Flag FEF(UART_FIFO[5]) or Parity Error Flag PEF(UART_FIFOSTS[4]), UART PDMA receive request operation is stopped. Clear Break Error Flag BIF or Frame Error Flag FEF or Parity Error Flag PEF by

		writing “1” to corresponding BIF, FEF and PEF to make UART PDMA receive request operation continue.
[14]	**TXPDMAEN**	**TX PDMA Enable Bit** 0 = TX PDMA Disabled. 1 = TX PDMA Enabled. **Note:** If RLSIEN (UART_INTEN[2]) is enabled and HWRLSINT (UART_INTSTS[26]) is set to 1, the RLS (Receive Line Status) Interrupt is caused. If RLS interrupt is caused by Break Error Flag BIF(UART_FIFOSTS[6]), Frame Error Flag FEF(UART_FIFO[5]) or Parity Error Flag PEF(UART_FIFOSTS[4]), UART PDMA transmit request operation is stopped. Clear Break Error Flag BIF or Frame Error Flag FEF or Parity Error Flag PEF by writing “1” to corresponding BIF, FEF and PEF to make UART PDMA transmit request operation continue.
[13]	**ATOCTSEN**	**nCTS Auto-flow Control Enable Bit** 0 = nCTS auto-flow control Disabled. 1 = nCTS auto-flow control Enabled. **Note:** When nCTS auto-flow is enabled, the UART will send data to external device if nCTS input assert (UART will not send data to device until nCTS is asserted).
[12]	**ATORTSEN**	**nRTS Auto-flow Control Enable Bit** 0 = nRTS auto-flow control Disabled. 1 = nRTS auto-flow control Enabled. **Note:** When nRTS auto-flow is enabled, if the number of bytes in the RX FIFO equals the RTSTRGLV (UART_FIFO[19:16]), the UART will de-assert nRTS signal.
[11]	**TOCNTEN**	**Receive Buffer Time-out Counter Enable Bit** 0 = Receive Buffer Time-out counter Disabled. 1 = Receive Buffer Time-out counter Enabled.
[10:9]	**Reserved**	Reserved.
[8]	**LINIEN**	**LIN Bus Interrupt Enable Bit** 0 = LIN bus interrupt Disabled. 1 = LIN bus interrupt Enabled. **Note:** This bit is used for LIN function mode.
[7]	**Reserved**	Reserved.
[6]	**WKIEN**	**Wake-up Interrupt Enable Bit** 0 = Wake-up Interrupt Disabled. 1 = Wake-up Interrupt Enabled.
[5]	**BUFERRIEN**	**Buffer Error Interrupt Enable Bit** 0 = Buffer error interrupt Disabled. 1 = Buffer error interrupt Enabled.
[4]	**RXTOIEN**	**RX Time-out Interrupt Enable Bit** 0 = RX time-out interrupt Disabled. 1 = RX time-out interrupt Enabled.
[3]	**MODEMIEN**	**Modem Status Interrupt Enable Bit** 0 = Modem status interrupt Disabled. 1 = Modem status interrupt Enabled.
[2]	**RLSIEN**	**Receive Line Status Interrupt Enable Bit** 0 = Receive Line Status interrupt Disabled. 1 = Receive Line Status interrupt Enabled.
[1]	**THREIEN**	**Transmit Holding Register Empty Interrupt Enable Bit**

		0 = Transmit holding register empty interrupt Disabled. 1 = Transmit holding register empty interrupt Enabled.
[0]	**RDAIEN**	**Receive Data Available Interrupt Enable Bit** 0 = Receive data available interrupt Disabled. 1 = Receive data available interrupt Enabled.

UART FIFO Control Register (UART_FIFO)

Register	Offset	R/W	Description	Reset Value
UART_FIFO **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x08	R/W	UART FIFO Control Register	0x0000_0101

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**RTSTRGLV**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**RXOFF**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RFITL				**Reserved**	**TXRST**	**RXRST**	**Reserved**

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:16]	**RTSTRGLV**	**nRTS Trigger Level for Auto-flow Control** 0000 = nRTS Trigger Level is 1 byte. 0001 = nRTS Trigger Level is 4 bytes. 0010 = nRTS Trigger Level is 8 bytes. 0011 = nRTS Trigger Level is 14 bytes. Others = Reserved. **Note:** This field is used for auto nRTS flow control.
[15:9]	**Reserved**	Reserved.
[8]	**RXOFF**	**Receiver Disable Bit** The receiver is disabled or not (set 1 to disable receiver). 0 = Receiver Enabled. 1 = Receiver Disabled. **Note:** This bit is used for RS-485 Normal Multi-drop mode. It should be programmed before RS485NMM (UART_ALTCTL [8]) is programmed.
[7:4]	**RFITL**	**RX FIFO Interrupt Trigger Level** When the number of bytes in the receive FIFO equals the RFITL, the RDAIF (UART_INTSTS[0]) will be set (if RDAIEN (UART_INTEN [0]) enabled, and an interrupt will be generated). 0000 = RX FIFO Interrupt Trigger Level is 1 byte. 0001 = RX FIFO Interrupt Trigger Level is 4 bytes. 0010 = RX FIFO Interrupt Trigger Level is 8 bytes. 0011 = RX FIFO Interrupt Trigger Level is 14 bytes. Others = Reserved.
[3]	**Reserved**	Reserved.
[2]	**TXRST**	**TX Field Software Reset** When TXRST (UART_FIFO[2]) is set, all the byte in the transmit FIFO and TX internal

		state machine are cleared. 0 = No effect. 1 = Reset the TX internal state machine and pointers. **Note 1:** This bit will automatically clear at least 3 UART peripheral clock cycles. **Note 2:** Before setting this bit, it should wait for the TXEMPTYF (UART_FIFOSTS[28]) to be set.
[1]	**RXRST**	**RX Field Software Reset** When RXRST (UART_FIFO[1]) is set, all the byte in the receiver FIFO and RX internal state machine are cleared. 0 = No effect. 1 = Reset the RX internal state machine and pointers. **Note 1:** This bit will automatically clear at least 3 UART peripheral clock cycles. **Note 2:** Before setting this bit, it should wait for the RXIDLE (UART_FIFOSTS[29]) to be set.
[0]	**Reserved**	Reserved.

UART Line Control Register (UART_LINE)

Register	Offset	R/W	Description	Reset Value
UART_LINE **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x0C	R/W	UART Line Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						RXDINV	TXDINV
7	6	5	4	3	2	1	0
PSS	BCB	SPE	EPE	PBE	NSB	WLS	

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**RXDINV**	**RX Data Inverted** 0 = Received data signal inverted Disabled. 1 = Received data signal inverted Enabled. **Note 1:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller. **Note 2:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is select UART, LIN or RS485 function.
[8]	**TXDINV**	**TX Data Inverted** 0 = Transmitted data signal inverted Disabled. 1 = Transmitted data signal inverted Enabled. **Note 1:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller. **Note 2:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is select UART, LIN or RS485 function.
[7]	**PSS**	**PARITY Bit Source Selection** The PARITY bit can be selected to be generated and checked automatically or by software. 0 = PARITY bit is generated by EPE (UART_LINE[4]) and SPE (UART_LINE[5]) setting and checked automatically. 1 = PARITY bit generated and checked by software. **Note 1:** This bit has effect only when PBE (UART_LINE[3]) is set. **Note 2:** If PSS is 0, the PARITY bit is transmitted and checked automatically. If PSS is 1, the transmitted PARITY bit value can be determined by writing PARITY (UART_DAT[8]) and the PARITY bit can be read by reading PARITY (UART_DAT[8]).
[6]	**BCB**	**Break Control Bit** 0 = Break Control Disabled.

		1 = Break Control Enabled. **Note:** When this bit is set to logic 1, the transmitted serial data output (TX) is forced to the Spacing State (logic 0). This bit acts only on TX line and has no effect on the transmitter logic.
[5]	**SPE**	**Stick Parity Enable Bit** 0 = Stick parity Disabled. 1 = Stick parity Enabled. **Note:** If PBE (UART_LINE[3]) and EPE (UART_LINE[4]) are logic 1, the PARITY bit is transmitted and checked as logic 0. If PBE (UART_LINE[3]) is 1 and EPE (UART_LINE[4]) is 0 then the PARITY bit is transmitted and checked as 1.
[4]	**EPE**	**Even Parity Enable Bit** 0 = Odd number of logic 1's is transmitted and checked in each word. 1 = Even number of logic 1's is transmitted and checked in each word. **Note:** This bit has effect only when PBE (UART_LINE[3]) is set.
[3]	**PBE**	**PARITY Bit Enable Bit** 0 = PARITY bit generated Disabled. 1 = PARITY bit generated Enabled. **Note:** PARITY bit is generated on each outgoing character and is checked on each incoming data.
[2]	**NSB**	**Number of "STOP Bit"** 0 = One "STOP bit" is generated in the transmitted data. 1 = When select 5-bit word length, 1.5 "STOP bit" is generated in the transmitted data. When select 6-, 7- and 8-bit word length, 2 "STOP bit" is generated in the transmitted data.
[1:0]	**WLS**	**Word Length Selection** This field sets UART word length. 00 = 5 bits. 01 = 6 bits. 10 = 7 bits. 11 = 8 bits.

UART Modem Control Register (UART_MODEM)

Register	Offset	R/W	Description	Reset Value
UART_MODEM **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x10	R/W	UART Modem Control Register	0x0000_0200

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**RTSSTS**	**Reserved**			**RTSACTLV**	**Reserved**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**RTS**	**Reserved**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**RTSSTS**	**nRTS Pin Status (Read Only)** This bit mirror from nRTS pin output of voltage logic status. 0 = nRTS pin output is low level voltage logic state. 1 = nRTS pin output is high level voltage logic state.
[12:10]	**Reserved**	Reserved.
[9]	**RTSACTLV**	**nRTS Pin Active Level** This bit defines the active level state of nRTS pin output. 0 = nRTS pin output is high level active. 1 = nRTS pin output is low level active. (Default) **Note 1:** Refer to Figure 6.23-13 and Figure 6.23-14 for UART function mode. **Note 2:** Refer to Figure 6.23-24 and Figure 6.23-25 for RS-485 function mode. **Note 3:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller.
[8:2]	**Reserved**	Reserved.
[1]	**RTS**	**nRTS Signal Control** This bit is direct control internal nRTS (Request-to-send) signal active or not, and then drive the nRTS pin output with RTSACTLV bit configuration. 0 = nRTS signal is active. 1 = nRTS signal is inactive. **Note 1:** The nRTS signal control bit is not effective when nRTS auto-flow control is enabled in UART function mode. **Note 2:** The nRTS signal control bit is not effective when RS-485 auto direction mode (AUD) is enabled in RS-485 function mode. **Note 3:** Single-wire mode supports this feature.
[0]	**Reserved**	Reserved.

UART Modem Status Register (UART_MODEMSTS)

Register	Offset	R/W	Description	Reset Value
UART_MODEMSTS **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x14	R/W	UART Modem Status Register	0x0000_0110

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**CTSACTLV**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**CTSSTS**	**Reserved**			**CTSDETF**

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**CTSACTLV**	**nCTS Pin Active Level** This bit defines the active level state of nCTS pin input. 0 = nCTS pin input is high level active. 1 = nCTS pin input is low level active. (Default) **Note:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, cleared TXRXDIS (UART_FUNCSEL[3]) to activate UART controller.
[7:5]	**Reserved**	Reserved.
[4]	**CTSSTS**	**nCTS Pin Status (Read Only)** This bit mirror from nCTS pin input of voltage logic status. 0 = nCTS pin input is low level voltage logic state. 1 = nCTS pin input is high level voltage logic state. **Note:** This bit echoes when UART controller peripheral clock is enabled, and nCTS multi-function port is selected.
[3:1]	**Reserved**	Reserved.
[0]	**CTSDETF**	**Detect nCTS State Change Flag** This bit is set whenever nCTS input has change state, and it will generate Modem interrupt to CPU when MODEMIEN (UART_INTEN [3]) is set to 1. 0 = nCTS input has not change state. 1 = nCTS input has change state. **Note:** This bit can be cleared by writing “1” to it.

UART FIFO Status Register (UART_FIFOSTS)

Register	Offset	R/W	Description	Reset Value
UART_FIFOSTS **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x18	R/W	UART FIFO Status Register	0xB040_4000

31	30	29	28	27	26	25	24
TXRXACT	Reserved	RXIDLE	TXEMPTYF	Reserved			TXOVIF
23	22	21	20	19	18	17	16
TXFULL	TXEMPTY	TXPTR					
15	14	13	12	11	10	9	8
RXFULL	RXEMPTY	RXPTR					
7	6	5	4	3	2	1	0
Reserved	BIF	FEF	PEF	ADDRDETF	ABRDTOIF	ABRDIF	RXOVIF

Bits	Description	
[31]	**TXRXACT**	**TX and RX Active Status (Read Only)** This bit indicates TX and RX are active or inactive. 0 = TX and RX are inactive. 1 = TX and RX are active. (Default) **Note:** When TXRXDIS (UART_FUNCSEL[3]) is set and both TX and RX are in idle state, this bit is cleared. The UART controller cannot transmit or receive data at this moment. Otherwise this bit is set.
[30]	**Reserved**	Reserved.
[29]	**RXIDLE**	**RX Idle Status (Read Only)** This bit is set by hardware when RX is idle. 0 = RX is busy. 1 = RX is idle. (Default)
[28]	**TXEMPTYF**	**Transmitter Empty Flag (Read Only)** This bit is set by hardware when TX FIFO (UART_DAT) is empty and the STOP bit of the last byte has been transmitted. 0 = TX FIFO is not empty or the STOP bit of the last byte has been not transmitted. 1 = TX FIFO is empty and the STOP bit of the last byte has been transmitted. **Note:** This bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed.
[27:25]	**Reserved**	Reserved.
[24]	**TXOVIF**	**TX Overflow Error Interrupt Flag** If TX FIFO (UART_DAT) is full, an additional write to UART_DAT will cause this bit to logic 1. 0 = TX FIFO is not overflow. 1 = TX FIFO is overflow. **Note:** This bit can be cleared by writing “1” to it.
[23]	**TXFULL**	**Transmitter FIFO Full (Read Only)**

		This bit indicates TX FIFO full or not. 0 = TX FIFO is not full. 1 = TX FIFO is full. **Note:** This bit is set when the number of usage in TX FIFO Buffer is equal to 16, otherwise it is cleared by hardware.
[22]	**TXEMPTY**	**Transmitter FIFO Empty (Read Only)** This bit indicates TX FIFO empty or not. 0 = TX FIFO is not empty. 1 = TX FIFO is empty. **Note:** When the last byte of TX FIFO has been transferred to Transmitter Shift Register, hardware sets this bit high. It will be cleared when writing data into UART_DAT (TX FIFO not empty).
[21:16]	**TXPTR**	**TX FIFO Pointer (Read Only)** This field indicates the TX FIFO Buffer Pointer. When CPU writes one byte into UART_DAT, TXPTR increases one. When one byte of TX FIFO is transferred to Transmitter Shift Register, TXPTR decreases one. The Maximum value shown in TXPTR is 15. When the using level of TX FIFO Buffer equal to 16, the TXFULL bit is set to 1 and TXPTR will show 0. As one byte of TX FIFO is transferred to Transmitter Shift Register, the TXFULL bit is cleared to 0 and TXPTR will show 15.
[15]	**RXFULL**	**Receiver FIFO Full (Read Only)** This bit initiates RX FIFO full or not. 0 = RX FIFO is not full. 1 = RX FIFO is full. **Note:** This bit is set when the number of usage in RX FIFO Buffer is equal to 16, otherwise it is cleared by hardware.
[14]	**RXEMPTY**	**Receiver FIFO Empty (Read Only)** This bit initiate RX FIFO empty or not. 0 = RX FIFO is not empty. 1 = RX FIFO is empty. **Note:** When the last byte of RX FIFO has been read by CPU, hardware sets this bit high. It will be cleared when UART receives any new data.
[13:8]	**RXPTR**	**RX FIFO Pointer (Read Only)** This field indicates the RX FIFO Buffer Pointer. When UART receives one byte from external device, RXPTR increases one. When one byte of RX FIFO is read by CPU, RXPTR decreases one. The Maximum value shown in RXPTR is 15. When the using level of RX FIFO Buffer equal to 16, the RXFULL bit is set to 1 and RXPTR will show 0. As one byte of RX FIFO is read by CPU, the RXFULL bit is cleared to 0 and RXPTR will show 15.
[7]	**Reserved**	Reserved.
[6]	**BIF**	**Break Interrupt Flag** This bit is set to logic 1 whenever the received data input (RX) is held in the “spacing state” (logic 0) for longer than a full word transmission time (that is, the total time of “START bit” + data bits + parity + STOP bits). 0 = No Break interrupt is generated. 1 = Break interrupt is generated. **Note:** This bit can be cleared by writing “1” to it.
[5]	**FEF**	**Framing Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “STOP bit” (that is, the STOP bit following the last data bit or PARITY bit is detected as logic 0). 0 = No framing error is generated.

		1 = Framing error is generated. **Note:** This bit can be cleared by writing “1” to it.
[4]	**PEF**	**Parity Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “PARITY bit”. 0 = No parity error is generated. 1 = Parity error is generated. **Note:** This bit can be cleared by writing “1” to it.
[3]	**ADDRDETF**	**RS-485 Address Byte Detect Flag** 0 = Receiver detects a data that is not an address bit (bit 9 =’0’). 1 = Receiver detects a data that is an address bit (bit 9 =’1’). **Note 1:** This field is used for RS-485 function mode and ADDRDEN (UART_ALTCTL[15]) is set to 1 to enable Address detection mode. **Note 2:** This bit can be cleared by writing “1” to it.
[2]	**ABRDTOIF**	**Auto-baud Rate Detect Time-out Interrupt Flag** This bit is set to logic “1” in Auto-baud Rate Detect mode when the baud rate counter is overflow. 0 = Auto-baud rate counter is underflow. 1 = Auto-baud rate counter is overflow. **Note:** This bit can be cleared by writing “1” to it.
[1]	**ABRDIF**	**Auto-baud Rate Detect Interrupt Flag** This bit is set to logic “1” when auto-baud rate detect function is finished. 0 = Auto-baud rate detect function is not finished. 1 = Auto-baud rate detect function is finished. **Note:** This bit can be cleared by writing “1” to it.
[0]	**RXOVIF**	**RX Overflow Error Interrupt Flag** This bit is set when RX FIFO overflow. If the number of bytes of received data is greater than RX_FIFO (UART_DAT) size 16 bytes, this bit will be set. 0 = RX FIFO is not overflow. 1 = RX FIFO is overflow. **Note:** This bit can be cleared by writing “1” to it.

UART Interrupt Status Register (UART_INTSTS)

Register	Offset	R/W	Description	Reset Value
UART_INTSTS **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x1C	R/W	UART Interrupt Status Register	0x0040_0002

31	30	29	28	27	26	25	24
ABRINT	TXENDINT	HWBUFEINT	HWTOINT	HWMODINT	HWRLSINT	Reserved	SWBEINT
23	22	21	20	19	18	17	16
Reserved	TXENDIF	HWBUFEIF	HWTOIF	HWMODIF	HWRLSIF	Reserved	SWBEIF
15	14	13	12	11	10	9	8
LININT	WKINT	BUFERRINT	RXTOINT	MODEMINT	RLSINT	THREINT	RDAINT
7	6	5	4	3	2	1	0
LINIF	WKIF	BUFERRIF	RXTOIF	MODEMIF	RLSIF	THREIF	RDAIF

Bits	Description	
[31]	**ABRINT**	**Auto-baud Rate Interrupt Indicator (Read Only)** This bit is set if ABRIEN (UART_INTEN[18]) and ABRIF (UART_ALTCTL[17]) are both set to 1. 0 = No Auto-baud Rate interrupt is generated. 1 = The Auto-baud Rate interrupt is generated.
[30]	**TXENDINT**	**Transmitter Empty Interrupt Indicator (Read Only)** This bit is set if TXENDIEN (UART_INTEN[22]) and TXENDIF(UART_INTSTS[22]) are both set to 1. 0 = No Transmitter Empty interrupt is generated. 1 = Transmitter Empty interrupt is generated.
[29]	**HWBUFEINT**	**PDMA Mode Buffer Error Interrupt Indicator (Read Only)** This bit is set if BUFERRIEN (UART_INTEN[5]) and HWBUFEIF (UART_INTSTS[21]) are both set to 1. 0 = No buffer error interrupt is generated in PDMA mode. 1 = Buffer error interrupt is generated in PDMA mode.
[28]	**HWTOINT**	**PDMA Mode RX Time-out Interrupt Indicator (Read Only)** This bit is set if RXTOIEN (UART_INTEN[4]) and HWTOIF(UART_INTSTS[20]) are both set to 1. 0 = No RX time-out interrupt is generated in PDMA mode. 1 = RX time-out interrupt is generated in PDMA mode.
[27]	**HWMODINT**	**PDMA Mode MODEM Status Interrupt Indicator (Read Only)** This bit is set if MODEMIEN (UART_INTEN[3]) and HWMODIF(UART_INTSTS[19]) are both set to 1. 0 = No Modem interrupt is generated in PDMA mode. 1 = Modem interrupt is generated in PDMA mode.
[26]	**HWRLSINT**	**PDMA Mode Receive Line Status Interrupt Indicator (Read Only)** This bit is set if RLSIEN (UART_INTEN[2]) and HWRLSIF(UART_INTSTS[18]) are both set to 1.

		0 = No RLS interrupt is generated in PDMA mode. 1 = RLS interrupt is generated in PDMA mode.
[25]	**Reserved**	Reserved.
[24]	**SWBEINT**	**Single-wire Bit Error Detect Interrupt Indicator (Read Only)** This bit is set if SWBEIEN (UART_INTEN[16]) and SWBEIF (UART_INTSTS[16]) are both set to 1. 0 = No Single-wire Bit Error Detection Interrupt generated. 1 = Single-wire Bit Error Detection Interrupt generated.
[23]	**Reserved**	Reserved.
[22]	**TXENDIF**	**Transmitter Empty Interrupt Flag** This bit is set when TX FIFO (UART_DAT) is empty and the STOP bit of the last byte has been transmitted (TXEMPTYF (UART_FIFOSTS[28]) is set). If TXENDIEN (UART_INTEN[22]) is enabled, the Transmitter Empty interrupt will be generated. 0 = No transmitter empty interrupt flag is generated. 1 = Transmitter empty interrupt flag is generated. **Note:** This bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed.
[21]	**HWBUFEIF**	**PDMA Mode Buffer Error Interrupt Flag (Read Only)** This bit is set when the TX or RX FIFO overflows (TXOVIF (UART_FIFOSTS [24]) or RXOVIF (UART_FIFOSTS[0]) is set). When BUFERRIF (UART_INTSTS[5]) is set, the transfer maybe is not correct. If BUFERRIEN (UART_INTEN [5]) is enabled, the buffer error interrupt will be generated. 0 = No buffer error interrupt flag is generated in PDMA mode. 1 = Buffer error interrupt flag is generated in PDMA mode. **Note:** This bit is cleared when both TXOVIF (UART_FIFOSTS[24]]) and RXOVIF (UART_FIFOSTS[0]) are cleared.
[20]	**HWTOIF**	**PDMA Mode RX Time-out Interrupt Flag (Read Only)** This bit is set when the RX FIFO is not empty and no activities occurred in the RX FIFO and the time-out counter equal to TOIC (UART_TOUT[7:0]). If RXTOIEN (UART_INTEN [4]) is enabled, the RX time-out interrupt will be generated. 0 = No RX time-out interrupt flag is generated in PDMA mode. 1 = RX time-out interrupt flag is generated in PDMA mode. **Note:** This bit is read only and user can read UART_DAT (RX is in active) to clear it.
[19]	**HWMODIF**	**PDMA Mode MODEM Interrupt Flag (Read Only)** This bit is set when the nCTS pin has state change (CTSDETF (UART_MODEMSTS [0] =1)). If MODEMIEN (UART_INTEN [3]) is enabled, the Modem interrupt will be generated. 0 = No Modem interrupt flag is generated in PDMA mode. 1 = Modem interrupt flag is generated in PDMA mode. **Note:** This bit is read only and reset to 0 when the bit CTSDETF (UART_MODEMSTS[0]) is cleared by writing 1 on CTSDETF (UART_MODEMSTS [0]).
[18]	**HWRLSIF**	**PDMA Mode Receive Line Status Flag (Read Only)** This bit is set when the RX receive data have parity error, frame error or break error (at least one of 3 bits, BIF (UART_FIFOSTS[6]), FEF (UART_FIFOSTS[5]) and PEF (UART_FIFOSTS[4]) is set). If RLSIEN (UART_INTEN [2]) is enabled, the RLS interrupt will be generated. 0 = No RLS interrupt flag is generated in PDMA mode. 1 = RLS interrupt flag is generated in PDMA mode. **Note 1:** In RS-485 function mode, this field includes “receiver detect any address byte received address byte character (bit9 = ‘1’) bit". **Note 2:** In UART function mode, this bit is read only and reset to 0 when all bits of BIF(UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) are

		cleared. **Note 3:** In RS-485 function mode, this bit is read only and reset to 0 when all bits of BIF(UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]), PEF(UART_FIFOSTS[4]) and ADDRDETF (UART_FIFOSTS[3]) are cleared.
[17]	**Reserved**	Reserved.
[16]	**SWBEIF**	**Single-wire Bit Error Detection Interrupt Flag** This bit is set when the single wire bus state not equals to UART controller TX state in Single-wire mode. 0 = No single-wire bit error detection interrupt flag is generated. 1 = Single-wire bit error detection interrupt flag is generated. **Note 1:** This bit is active when FUNCSEL (UART_FUNCSEL[2:0]) is select UART Single-wire mode. **Note 2:** This bit can be cleared by writing “1” to it.
[15]	**LININT**	**LIN Bus Interrupt Indicator (Read Only)** This bit is set if LINIEN (UART_INTEN[8]) and LINIF(UART_INTSTS[7]) are both set to 1. 0 = No LIN Bus interrupt is generated. 1 = The LIN Bus interrupt is generated.
[14]	**WKINT**	**UART Wake-up Interrupt Indicator (Read Only)** This bit is set if WKIEN (UART_INTEN[6]) and WKIF (UART_INTSTS[6]) are both set to 1. 0 = No UART wake-up interrupt is generated. 1 = UART wake-up interrupt is generated.
[13]	**BUFERRINT**	**Buffer Error Interrupt Indicator (Read Only)** This bit is set if BUFERRIEN(UART_INTEN[5]) and BUFERRIF(UART_ INTSTS[5]) are both set to 1. 0 = No buffer error interrupt is generated. 1 = Buffer error interrupt is generated.
[12]	**RXTOINT**	**RX Time-out Interrupt Indicator (Read Only)** This bit is set if RXTOIEN (UART_INTEN[4]) and RXTOIF(UART_INTSTS[4]) are both set to 1. 0 = No RX time-out interrupt is generated. 1 = RX time-out interrupt is generated.
[11]	**MODEMINT**	**MODEM Status Interrupt Indicator (Read Only)** This bit is set if MODEMIEN(UART_INTEN[3]) and MODEMIF(UART_INTSTS[3]) are both set to 1 0 = No Modem interrupt is generated. 1 = Modem interrupt is generated..
[10]	**RLSINT**	**Receive Line Status Interrupt Indicator (Read Only)** This bit is set if RLSIEN (UART_INTEN[2]) and RLSIF(UART_INTSTS[2]) are both set to 1. 0 = No RLS interrupt is generated. 1 = RLS interrupt is generated.
[9]	**THREINT**	**Transmit Holding Register Empty Interrupt Indicator (Read Only)** This bit is set if THREIEN (UART_INTEN[1]) and THREIF(UART_INTSTS[1]) are both set to 1. 0 = No THRE interrupt is generated. 1 = THRE interrupt is generated.

[8]	**RDAINT**	**Receive Data Available Interrupt Indicator (Read Only)** This bit is set if RDAIEN (UART_INTEN[0]) and RDAIF (UART_INTSTS[0]) are both set to 1. 0 = No RDA interrupt is generated. 1 = RDA interrupt is generated.
[7]	**LINIF**	**LIN Bus Interrupt Flag** This bit is set when LIN slave header detect (SLVHDETF (UART_LINSTS[0] =1)), LIN break detect (BRKDETF(UART_LINSTS[8]=1)), bit error detect (BITEF(UART_LINSTS[9]=1)), LIN slave ID parity error (SLVIDPEF(UART_LINSTS[2] = 1)) or LIN slave header error detect (SLVHEF (UART_LINSTS[1])). If LINIEN (UART_INTEN [8]) is enabled the LIN interrupt will be generated. 0 = None of SLVHDETF, BRKDETF, BITEF, SLVIDPEF and SLVHEF is generated. 1 = At least one of SLVHDETF, BRKDETF, BITEF, SLVIDPEF and SLVHEF is generated. **Note:** This bit is cleared when SLVHDETF(UART_LINSTS[0]), BRKDETF(UART_LINSTS[8]), BITEF(UART_LINSTS[9]), SLVIDPEF (UART_LINSTS[2]) and SLVHEF(UART_LINSTS[1]) all are cleared and software writing '1' to LINIF(UART_INTSTS[7]).
[6]	**WKIF**	**UART Wake-up Interrupt Flag (Read Only)** This bit is set when TOUTWKF (UART_WKSTS[4]), RS485WKF (UART_WKSTS[3]), RFRTWKF (UART_WKSTS[2]), DATWKF (UART_WKSTS[1]) or CTSWKF(UART_WKSTS[0]) is set to 1. 0 = No UART wake-up interrupt flag is generated. 1 = UART wake-up interrupt flag is generated. **Note:** This bit is cleared if all of TOUTWKF, RS485WKF, RFRTWKF, DATWKF and CTSWKF are cleared to 0 by writing 1 to the corresponding interrupt flag.
[5]	**BUFERRIF**	**Buffer Error Interrupt Flag (Read Only)** This bit is set when the TX FIFO or RX FIFO overflows (TXOVIF (UART_FIFOSTS[24]) or RXOVIF (UART_FIFOSTS[0]) is set). When BUFERRIF (UART_INTSTS[5]) is set, the transfer is not correct. If BUFERRIEN (UART_INTEN [5]) is enabled, the buffer error interrupt will be generated. 0 = No buffer error interrupt flag is generated. 1 = Buffer error interrupt flag is generated. **Note:** This bit is cleared if both of RXOVIF(UART_FIFOSTS[0]) and TXOVIF(UART_FIFOSTS[24]) are cleared to 0 by writing 1 to RXOVIF(UART_FIFOSTS[0]) and TXOVIF(UART_FIFOSTS[24]).
[4]	**RXTOIF**	**RX Time-out Interrupt Flag (Read Only)** This bit is set when the RX FIFO is not empty and no activities occurred in the RX FIFO and the time-out counter equal to TOIC (UART_TOUT[7:0]). If RXTOIEN (UART_INTEN [4]) is enabled, the RX time-out interrupt will be generated. 0 = No RX time-out interrupt flag is generated. 1 = RX time-out interrupt flag is generated. **Note:** This bit is read only and user can read UART_DAT (RX is in active) to clear it.
[3]	**MODEMIF**	**MODEM Interrupt Flag (Read Only)** This bit is set when the nCTS pin has state change (CTSDETF (UART_MODEMSTS[0]) = 1). If MODEMIEN (UART_INTEN [3]) is enabled, the Modem interrupt will be generated. 0 = No Modem interrupt flag is generated. 1 = Modem interrupt flag is generated. **Note:** This bit is read only and reset to 0 when bit CTSDETF is cleared by a write 1 on CTSDETF(UART_MODEMSTS[0]).
[2]	**RLSIF**	**Receive Line Interrupt Flag (Read Only)** This bit is set when the RX receive data have parity error, frame error or break error (at least one of 3 bits, BIF(UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and

		PEF(UART_FIFOSTS[4]), is set). If RLSIEN (UART_INTEN [2]) is enabled, the RLS interrupt will be generated. 0 = No RLS interrupt flag is generated. 1 = RLS interrupt flag is generated. **Note 1:** In RS-485 function mode, this field is set include “receiver detect and received address byte character (bit9 = ‘1’) bit". At the same time, the bit of ADDRDETF (UART_FIFOSTS[3]) is also set. **Note 2:** This bit is read only and reset to 0 when all bits of BIF (UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) are cleared. **Note 3:** In RS-485 function mode, this bit is read only and reset to 0 when all bits of BIF (UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]), PEF(UART_FIFOSTS[4]) and ADDRDETF (UART_FIFOSTS[3]) are cleared.
[1]	**THREIF**	**Transmit Holding Register Empty Interrupt Flag (Read Only)** This bit is set when the last data of TX FIFO is transferred to Transmitter Shift Register. If THREIEN (UART_INTEN[1]) is enabled, the THRE interrupt will be generated. 0 = No THRE interrupt flag is generated. 1 = THRE interrupt flag is generated. **Note:** This bit is read only and will be cleared when writing data into UART_DAT (TX FIFO is not empty).
[0]	**RDAIF**	**Receive Data Available Interrupt Flag (Read Only)** When the number of bytes in the RX FIFO equals the RFITL then the RDAIF(UART_INTSTS[0]) will be set. If RDAIEN (UART_INTEN [0]) is enabled, the RDA interrupt will be generated. 0 = No RDA interrupt flag is generated. 1 = RDA interrupt flag is generated. **Note:** This bit is read only and will be cleared when the number of unread bytes of RX FIFO drops below the threshold level (RFITL(UART_FIFO[7:4]).

UART Time-out Register (UART_TOUT)

Register	Offset	R/W	Description	Reset Value
UART_TOUT **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x20	R/W	UART Time-out Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DLY							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOIC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**DLY**	**TX Delay Time Value** This field is used to program the transfer delay time between the last STOP bit and next START bit. The unit is bit time.
[7:0]	**TOIC**	**Time-out Interrupt Comparator** The time-out counter resets and starts counting (the counting clock = baud rate) whenever the RX FIFO receives a new data word if time out counter is enabled by setting TOCNTEN (UART_INTEN[11]). Once the content of time-out counter is equal to that of time-out interrupt comparator (TOIC (UART_TOUT[7:0])), a receiver time-out interrupt (RXTOINT(UART_INTSTS[12])) is generated if RXTOIEN (UART_INTEN [4]) enabled. A new incoming data word or RX FIFO empty will clear RXTOIF (UART_INTSTS[4]). In order to avoid receiver time-out interrupt generation immediately during one character is being received, TOIC value should be set between 40 and 255. So, for example, if TOIC is set with 40, the time-out interrupt is generated after four characters are not received when 1 STOP bit and no parity check is set for UART transfer.

UART Baud Rate Divider Register (UART_BAUD)

Register	Offset	R/W	Description	Reset Value
UART_BAUD **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x24	R/W	UART Baud Rate Divider Register	0x0F00_0000

31	30	29	28	27	26	25	24
Reserved		BAUDM1	BAUDM0	EDIVM1			
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
BRD							
7	6	5	4	3	2	1	0
BRD							

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**BAUDM1**	**BAUD Rate Mode Selection Bit 1** This bit is baud rate mode selection bit 1. UART provides three baud rate calculation modes. This bit combines with BAUDM0 (UART_BAUD[28]) to select baud rate calculation mode. The detailed description is shown in Table 6.23-4. **Note:** In IrDA mode must be operated in mode 0.
[28]	**BAUDM0**	**BAUD Rate Mode Selection Bit 0** This bit is baud rate mode selection bit 0. UART provides three baud rate calculation modes. This bit combines with BAUDM1 (UART_BAUD[29]) to select baud rate calculation mode. The detailed description is shown in Table 6.23-4.
[27:24]	**EDIVM1**	**Extra Divider for BAUD Rate Mode 1** This field is used for baud rate calculation in mode 1 and has no effect for baud rate calculation in mode 0 and mode 2. The detailed description is shown in Table 6.23-4.
[23:16]	**Reserved**	Reserved.
[15:0]	**BRD**	**Baud Rate Divider** The field indicates the baud rate divider. This filed is used in baud rate calculation. The detailed description is shown in Table 6.23-4.

UART IrDA Control Register (UART_IRDA)

Register	Offset	R/W	Description	Reset Value
UART_IRDA **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x28	R/W	UART IrDA Control Register	0x0000_0040

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**RXINV**	**TXINV**	**Reserved**			**TXEN**	**Reserved**

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**RXINV**	**IrDA Inverse Receive Input Signal** 0 = None inverse receiving input signal. 1 = Inverse receiving input signal. (Default) **Note 1:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, clear TXRXDIS (UART_FUNCSEL[3]) to activate UART controller. **Note 2:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) is selects IrDA function.
[5]	**TXINV**	**IrDA Inverse Transmitting Output Signal** 0 = None inverse transmitting signal. (Default). 1 = Inverse transmitting output signal. **Note 1:** Before setting this bit, TXRXDIS (UART_FUNCSEL[3]) should be set then waited for TXRXACT (UART_FIFOSTS[31]) is cleared. When the configuration is done, clear TXRXDIS (UART_FUNCSEL[3]) to activate UART controller. **Note 2:** This bit is valid when FUNCSEL (UART_FUNCSEL[2:0]) selects IrDA function.
[4:2]	**Reserved**	Reserved.
[1]	**TXEN**	**IrDA Receiver/Transmitter Selection Enable Bit** 0 = IrDA Transmitter Disabled and Receiver Enabled. (Default) 1 = IrDA Transmitter Enabled and Receiver Disabled.
[0]	**Reserved**	Reserved.

UART Alternate Control/Status Register (UART_ALTCTL)

Register	Offset	R/W	Description	Reset Value
UART_ALTCTL **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x2C	R/W	UART Alternate Control/Status Register	0x0000_000C

31	30	29	28	27	26	25	24
ADDRMV							
23	22	21	20	19	18	17	16
Reserved			ABRDBITS		ABRDEN	ABRIF	Reserved
15	14	13	12	11	10	9	8
ADDRDEN	Reserved				RS485AUD	RS485AAD	RS485NMM
7	6	5	4	3	2	1	0
LINTXEN	LINRXEN	Reserved		BRKFL			

Bits	Description	
[31:24]	**ADDRMV**	**Address Match Value** This field contains the RS-485 address match values. **Note:** This field is used for RS-485 auto address detection mode.
[23:21]	**Reserved**	Reserved.
[20:19]	**ABRDBITS**	**Auto-baud Rate Detect Bit Length** 00 = 1-bit time from START bit to the 1st rising edge. The input pattern shall be 0x01. 01 = 2-bit time from START bit to the 1st rising edge. The input pattern shall be 0x02. 10 = 4-bit time from START bit to the 1st rising edge. The input pattern shall be 0x08. 11 = 8-bit time from START bit to the 1st rising edge. The input pattern shall be 0x80. **Note :** The calculation of bit number includes the START bit.
[18]	**ABRDEN**	**Auto-baud Rate Detect Enable Bit** 0 = Auto-baud rate detect function Disabled. 1 = Auto-baud rate detect function Enabled. **Note :** This bit is cleared automatically after auto-baud detection is finished.
[17]	**ABRIF**	**Auto-baud Rate Interrupt Flag (Read Only)** This bit is set when auto-baud rate detection function finished or the auto-baud rate counter was overflow and if ABRIEN(UART_INTEN [18]) is set then the auto-baud rate interrupt will be generated. 0 = No auto-baud rate interrupt flag is generated. 1 = Auto-baud rate interrupt flag is generated. **Note:** This bit is read only, but it can be cleared by writing “1” to ABRDTOIF (UART_FIFOSTS[2]) and ABRDIF(UART_FIFOSTS[1]).
[16]	**Reserved**	Reserved.
[15]	**ADDRDEN**	**RS-485 Address Detection Enable Bit** This bit is used to enable RS-485 Address Detection mode. 0 = Address detection mode Disabled. 1 = Address detection mode Enabled.

		Note: This bit is used for RS-485 any operation mode.
[14:11]	**Reserved**	Reserved.
[10]	**RS485AUD**	**RS-485 Auto Direction Function** 0 = RS-485 Auto Direction Operation function (AUD) Disabled. 1 = RS-485 Auto Direction Operation function (AUD) Enabled. **Note:** It can be active with RS-485_AAD or RS-485_NMM operation mode.
[9]	**RS485AAD**	**RS-485 Auto Address Detection Operation Mode** 0 = RS-485 Auto Address Detection Operation mode (AAD) Disabled. 1 = RS-485 Auto Address Detection Operation mode (AAD) Enabled. **Note:** It cannot be active with RS-485_NMM operation mode.
[8]	**RS485NMM**	**RS-485 Normal Multi-drop Operation Mode** 0 = RS-485 Normal Multi-drop Operation mode (NMM) Disabled. 1 = RS-485 Normal Multi-drop Operation mode (NMM) Enabled. **Note:** It cannot be active with RS-485_AAD operation mode.
[7]	**LINTXEN**	**LIN TX Break Mode Enable Bit** 0 = LIN TX Break mode Disabled. 1 = LIN TX Break mode Enabled. **Note:** When TX break field transfer operation is finished, this bit will be cleared automatically.
[6]	**LINRXEN**	**LIN RX Enable Bit** 0 = LIN RX mode Disabled. 1 = LIN RX mode Enabled.
[5:4]	**Reserved**	Reserved.
[3:0]	**BRKFL**	**UART LIN Break Field Length** This field indicates a 4-bit LIN TX break field count. **Note 1:** This break field length is BRKFL + 1. **Note 2:** According to LIN spec, the reset value is 0xC (break field length = 13).

UART Function Select Register (UART_FUNCSEL)

Register	Offset	R/W	Description	Reset Value
UART_FUNCSEL **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x30	R/W	UART Function Select Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXRXSWP	**DGE**	**Reserved**		**TXRXDIS**	**FUNCSEL**		

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**TXRXSWP**	**TX and RX Swap Enable Bit** Setting this bit Swaps TX pin and RX pin. 0 = TX and RX Swap Disabled. 1 = TX and RX Swap Enabled.
[6]	**DGE**	**Deglitch Enable Bit** 0 = Deglitch Disabled. 1 = Deglitch Enabled. **Note 1:** When this bit is set to logic 1, any pulse width less than about 150 ns will be considered a glitch and will be removed in the serial data input (RX). This bit acts only on RX line and has no effect on the transmitter logic. **Note 2:** It is recommended to set this bit only when operating at baud rate under 2.5 Mbps.
[5:4]	**Reserved**	Reserved.
[3]	**TXRXDIS**	**TX and RX Disable Bit** Setting this bit can disable TX and RX. 0 = TX and RX Enabled. 1 = TX and RX Disabled. **Note:** The TX and RX will not be disabled immediately when this bit is set. The TX and RX complete current task before TX and RX are disabled. When TX and RX are disabled, the TXRXACT (UART_FIFOSTS[31]) is cleared.
[2:0]	**FUNCSEL**	**Function Select** 000 = UART function. 001 = LIN function. 010 = IrDA function. 011 = RS-485 function. 100 = UART Single-wire function. Others = Reserved.

UART LIN Control Register (UART_LINCTL)

Register	Offset	R/W	Description	Reset Value
UART_LINCTL **x=0,1**	UARTx_BA+0x34	R/W	UART LIN Control Register	0x000C_0000

31	30	29	28	27	26	25	24
PID							
23	22	21	20	19	18	17	16
HSEL		BSL		BRKFL			
15	14	13	12	11	10	9	8
Reserved			BITERREN	LINRXOFF	BRKDETEN	IDPEN	SENDH
7	6	5	4	3	2	1	0
Reserved			MUTE	SLVDUEN	SLVAREN	SLVHDEN	SLVEN

Bits	Description	
[31:24]	**PID**	**LIN PID Bits** This field contains the LIN frame ID value in LIN function mode, and the frame ID parity can be generated by software or hardware depends on IDPEN (UART_LINCTL[9]) = 1. If the parity generated by hardware, user fill ID0~ID5 (PID [29:24]), hardware will calculate P0 (PID[30]) and P1 (PID[31]), otherwise user must filled frame ID and parity in this field. **Note 1:** User can fill any 8-bit value to this field and the bit 24 indicates ID0 (LSB first). **Note 2:** This field can be used for LIN master mode or slave mode.
[23:22]	**HSEL**	**LIN Header Select** 00 = The LIN header includes “break field”. 01 = The LIN header includes “break field” and “sync field”. 10 = The LIN header includes “break field”, “sync field” and “frame ID field”. 11 = Reserved. **Note:** This bit is used to master mode for LIN to send header field (SENDH (UART_LINCTL [8]) = 1) or used to slave to indicate exit from mute mode condition (MUTE (UART_LINCTL[4] = 1).
[21:20]	**BSL**	**LIN Break/Sync Delimiter Length** 00 = The LIN break/sync delimiter length is 1-bit time. 01 = The LIN break/sync delimiter length is 2-bit time. 10 = The LIN break/sync delimiter length is 3-bit time. 11 = The LIN break/sync delimiter length is 4-bit time. **Note:** This bit used for LIN master to sending header field.
[19:16]	**BRKFL**	**LIN Break Field Length** This field indicates a 4-bit LIN TX break field count. **Note 1:** These registers are shadow registers of BRKFL (UART_ALTCTL[3:0]). User can read/write it by setting BRKFL (UART_ALTCTL[3:0]) or BRKFL (UART_LINCTL[19:16]). **Note 2:** This break field length is BRKFL + 1. **Note 3:** According to LIN spec, the reset value is 12 (break field length = 13).
[15:13]	**Reserved**	Reserved.

[12]	**BITERREN**	**Bit Error Detect Enable Bit** 0 = Bit error detection function Disabled. 1 = Bit error detection function Enabled. **Note:** In LIN function mode, when occur bit error, the BITEF (UART_LINSTS[9]) flag will be asserted. If the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated.
[11]	**LINRXOFF**	**LIN Receiver Disable Bit** If the receiver is enabled (LINRXOFF (UART_LINCTL[11]) = 0), all received byte data will be accepted and stored in the RX FIFO, and if the receiver is disabled (LINRXOFF (UART_LINCTL[11] = 1), all received byte data will be ignore. 0 = LIN receiver Enabled. 1 = LIN receiver Disabled. **Note:** This bit is only valid when operating in LIN function mode (FUNCSEL (UART_FUNCSEL[2:0]) = 001).
[10]	**BRKDETEN**	**LIN Break Detection Enable Bit** When detect consecutive dominant greater than 11 bits, and are followed by a delimiter character, the BRKDETF (UART_LINSTS[8]) flag is set at the end of break field. If the LINIEN (UART_INTEN [8])=1, an interrupt will be generated. 0 = LIN break detection Disabled. 1 = LIN break detection Enabled.
[9]	**IDPEN**	**LIN ID Parity Enable Bit** 0 = LIN frame ID parity Disabled. 1 = LIN frame ID parity Enabled. **Note 1:** This bit can be used for LIN master to sending header field (SENDH (UART_LINCTL[8])) = 1 and HSEL (UART_LINCTL[23:22]) = 10 or be used for enable LIN slave received frame ID parity checked. **Note 2:** This bit is only used when the operation header transmitter is in HSEL (UART_LINCTL[23:22]) = 10.
[8]	**SENDH**	**LIN TX Send Header Enable Bit** The LIN TX header can be “break field” or “break and sync field” or “break, sync and frame ID field”, it is depend on setting HSEL (UART_LINCTL[23:22]). 0 = Send LIN TX header Disabled. 1 = Send LIN TX header Enabled. **Note 1:** This bit is shadow bit of LINTXEN (UART_ALTCTL [7]); user can read/write it by setting LINTXEN (UART_ALTCTL [7]) or SENDH (UART_LINCTL [8]). **Note 2:** When transmitter header field (it may be “break” or “break + sync” or “break + sync + frame ID” selected by HSEL (UART_LINCTL[23:22]) field) transfer operation finished, this bit will be cleared automatically.
[7:5]	**Reserved**	Reserved.
[4]	**MUTE**	**LIN Mute Mode Enable Bit** 0 = LIN mute mode Disabled. 1 = LIN mute mode Enabled. **Note:** The exit from mute mode condition and each control and interactions of this field are explained in Mute Mode and LIN Exit from Mute Mode Condition section.
[3]	**SLVDUEN**	**LIN Slave Divider Update Method Enable Bit** 0 = UART_BAUD updated is written by software (if no automatic resynchronization update occurs at the same time). 1 = UART_BAUD is updated at the next received character. User must set the bit before checksum reception. **Note 1:** This bit only is valid in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1). **Note 2:** This bit used for LIN Slave Automatic Resynchronization mode. (for Non-Automatic Resynchronization mode, this bit should be kept cleared)

		Note 3: The control and interactions of this field are explained in Slave Mode with Automatic Resynchronization section.
[2]	**SLVAREN**	**LIN Slave Automatic Resynchronization Mode Enable Bit** 0 = LIN automatic resynchronization Disabled. 1 = LIN automatic resynchronization Enabled. **Note 1:** This bit only is valid in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1). **Note 2:** When operation in Automatic Resynchronization mode, the baud rate setting must be mode2 (BAUDM1 (UART_BAUD [29]) and BAUDM0 (UART_BAUD [28]) must be 1). **Note 3:** The control and interactions of this field are explained in Slave Mode with Automatic Resynchronization section.
[1]	**SLVHDEN**	**LIN Slave Header Detection Enable Bit** 0 = LIN slave header detection Disabled. 1 = LIN slave header detection Enabled. **Note 1:** This bit is only valid in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1). **Note 2:** In LIN function mode, when detect header field (break + sync + frame ID), SLVHDETF (UART_LINSTS [0]) flag will be asserted. If the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated.
[0]	**SLVEN**	**LIN Slave Mode Enable Bit** 0 = LIN slave mode Disabled. 1 = LIN slave mode Enabled.

UART LIN Status Register (UART_LINSTS)

Register	Offset	R/W	Description	Reset Value
UART_LINSTS **x=0,1**	UARTx_BA+0x38	R/W	UART LIN Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						BITEF	BRKDETF
7	6	5	4	3	2	1	0
Reserved				SLVSYNCF	SLVIDPEF	SLVHEF	SLVHDETF

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**BITEF**	**Bit Error Detect Status Flag** At TX transfer state, hardware will monitor the bus state, if the input pin (UART_RXD) state not equals to the output pin (UART_TXD) state, BITEF (UART_LINSTS[9]) will be set. When occur bit error, if the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated. 0 = Bit error not detected. 1 = Bit error detected. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid when enable bit error detection function (BITERREN (UART_LINCTL [12]) = 1).
[8]	**BRKDETF**	**LIN Break Detection Flag** This bit is set by hardware when a break is detected and be cleared by writing 1 to it through software. 0 = LIN break not detected. 1 = LIN break detected. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid when LIN break detection function is enabled (BRKDETEN (UART_LINCTL[10]) =1).
[7:4]	**Reserved**	Reserved.
[3]	**SLVSYNCF**	**LIN Slave Sync Field** This bit indicates that the LIN sync field is being analyzed in Automatic Resynchronization mode. When the receiver header have some error been detect, user must reset the internal circuit to re-search new frame header by writing 1 to this bit. 0 = The current character is not at LIN sync state. 1 = The current character is at LIN sync state. **Note 1:** This bit is only valid in LIN Slave mode (SLVEN(UART_LINCTL[0]) = 1). **Note 2:** This bit can be cleared by writing 1 to it. **Note 3:** When writing 1 to it, hardware will reload the initial baud rate and re-search a new

		frame header.
[2]	**SLVIDPEF**	**LIN Slave ID Parity Error Flag** This bit is set by hardware when receipted frame ID parity is not correct. 0 = No active. 1 = Receipted frame ID parity is not correct. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid in LIN slave mode (SLVEN (UART_LINCTL [0])= 1) and enable LIN frame ID parity check function IDPEN (UART_LINCTL [9]).
[1]	**SLVHEF**	**LIN Slave Header Error Flag** This bit is set by hardware when a LIN header error is detected in LIN slave mode and be cleared by writing 1 to it. The header errors include “break delimiter is too short (less than 0.5 bit time)”, “frame error in sync field or Identifier field”, “sync field data is not 0x55 in Non-Automatic Resynchronization mode”, “sync field deviation error with Automatic Resynchronization mode”, “sync field measure time-out with Automatic Resynchronization mode” and “LIN header reception time-out”. 0 = LIN header error not detected. 1 = LIN header error detected. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid when UART is operated in LIN slave mode (SLVEN (UART_LINCTL [0]) = 1) and enables LIN slave header detection function (SLVHDEN (UART_LINCTL [1])).
[0]	**SLVHDETF**	**LIN Slave Header Detection Flag** This bit is set by hardware when a LIN header is detected in LIN slave mode and be cleared by writing 1 to it. 0 = LIN header not detected. 1 = LIN header detected (break + sync + frame ID). **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is only valid in LIN slave mode (SLVEN (UART_LINCTL [0]) = 1) and enable LIN slave header detection function (SLVHDEN (UART_LINCTL [1])). **Note 3:** When enable ID parity check IDPEN (UART_LINCTL [9]), if hardware detect complete header (“break + sync + frame ID”), the SLVHDETF will be set whether the frame ID correct or not.

UART Baud Rate Compensation Register (UART_BRCOMP)

Register	Offset	R/W	Description	Reset Value
UART_BRCOMP **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x3C	R/W	UART Baud Rate Compensation Register	0x0000_0000

31	30	29	28	27	26	25	24
BRCOMPDEC	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**BRCOMP**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BRCOMP							

Bits	Description	
[31]	**BRCOMPDEC**	**Baud Rate Compensation Decrease** 0 = Positive (increase one module clock) compensation for each compensated bit. 1 = Negative (decrease one module clock) compensation for each compensated bit.
[30:9]	**Reserved**	Reserved.
[8:0]	**BRCOMP**	**Baud Rate Compensation Patten** These 9-bits are used to define the relative bit is compensated or not. BRCOMP[7:0] is used to define the compensation of DAT (UART_DAT[7:0]) and BRCOMP[8] is used to define PARITY (UART_DAT[8]).

UART Wake-up Control Register (UART_WKCTL)

Register	Offset	R/W	Description	Reset Value
UART_WKCTL **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x40	R/W	UART Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**WKTOUTEN**	**WKRS485EN**	**WKRFRTEN**	**WKDATEN**	**WKCTSEN**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**WKTOUTEN**	**Received Data FIFO Reached Threshold Time-out Wake-up Enable Bit** 0 = Received Data FIFO reached threshold time-out wake-up system function Disabled. 1 = Received Data FIFO reached threshold time-out wake-up system function Enabled. **Note 1:** When the system is in Power-down mode, Received Data FIFO reached threshold time-out will wake up system from Power-down mode. **Note 2:** It is suggested the function is enabled when the WKRFRTEN (UART_WKCTL[2]) is set to 1.
[3]	**WKRS485EN**	**RS-485 Address Match Wake-up Enable Bit** 0 = RS-485 Address Match (AAD mode) wake-up system function Disabled. 1 = RS-485 Address Match (AAD mode) wake-up system function Enabled. **Note 1:** When the system is in Power-down mode, RS-485 Address Match will wake-up system from Power-down mode. **Note 2:** This bit is used for RS-485 Auto Address Detection (AAD) mode in RS-485 function mode and ADDRDEN (UART_ALTCTL[15]) is set to 1.
[2]	**WKRFRTEN**	**Received Data FIFO Reached Threshold Wake-up Enable Bit** 0 = Received Data FIFO reached threshold wake-up system function Disabled. 1 = Received Data FIFO reached threshold wake-up system function Enabled. **Note:** When the system is in Power-down mode, Received Data FIFO reached threshold will wake-up system from Power-down mode.
[1]	**WKDATEN**	**Incoming Data Wake-up Enable Bit** 0 = Incoming data wake-up system function Disabled. 1 = Incoming data wake-up system function Enabled. **Note:** When the system is in Power-down mode, incoming data will wake-up system from Power-down mode.
[0]	**WKCTSEN**	**nCTS Wake-up Enable Bit** 0 = nCTS Wake-up system function Disabled. 1 = nCTS Wake-up system function Enabled.

		Note: When the system is in Power-down mode, an external nCTS change will wake up system from Power-down mode.

UART Wake-up Status Register (UART_WKSTS)

Register	Offset	R/W	Description	Reset Value
UART_WKSTS **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x44	R/W	UART Wake-up Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**TOUTWKF**	**RS485WKF**	**RFRTWKF**	**DATWKF**	**CTSWKF**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**TOUTWKF**	**Received Data FIFO Threshold Time-out Wake-up Flag** This bit is set if chip wake-up from power-down state by Received Data FIFO Threshold Time-out wake-up. 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by Received Data FIFO reached threshold time-out. **Note 1:** If WKTOUTEN (UART_WKCTL[4]) is enabled, the Received Data FIFO reached threshold time-out wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.
[3]	**RS485WKF**	**RS-485 Address Match Wake-up Flag** This bit is set if chip wake-up from power-down state by RS-485 Address Match (AAD mode). 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by RS-485 Address Match (AAD mode) wake-up. **Note 1:** If WKRS485EN (UART_WKCTL[3]) is enabled, the RS-485 Address Match (AAD mode) wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.
[2]	**RFRTWKF**	**Received Data FIFO Reached Threshold Wake-up Flag** This bit is set if chip wake-up from power-down state by Received Data FIFO reached threshold wake-up. 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by Received Data FIFO Reached Threshold wake-up. **Note 1:** If WKRFRTEN (UART_WKCTL[2]) is enabled, the Received Data FIFO Reached Threshold wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.

[1]	**DATWKF**	**Incoming Data Wake-up Flag** This bit is set if chip wake-up from power-down state by data wake-up. 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by Incoming Data wake-up. **Note 1:** If WKDATEN (UART_WKCTL[1]) is enabled, the Incoming Data wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.
[0]	**CTSWKF**	**nCTS Wake-up Flag** This bit is set if chip wake-up from power-down state by nCTS wake-up. 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by nCTS wake-up. **Note 1:** If WKCTSEN (UART_WKCTL[0]) is enabled, the nCTS wake-up cause this bit is set to '1'. **Note 2:** This bit can be cleared by writing '1' to it.

UART Incoming Data Wake-up Compensation Register (UART_DWKCOMP)

Register	Offset	R/W	Description	Reset Value
UART_DWKCOMP x=0,1,2,3,4,5,6,7,8,9	UARTx_BA+0x48	R/W	UART Incoming Data Wake-up Compensation Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
STCOMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STCOMP							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**STCOMP**	**START Bit Compensation Value** These bits field indicate how many clock cycle selected by UART_CLK do the UART controller can get the 1st bit (START bit) when the device is wake-up from Power-down mode. **Note:** It is valid only when WKDATEN (UART_WKCTL[1]) is set.

UART RS485 Transceiver Deactivate Delay Register (UART_RS485DD)

Register	Offset	R/W	Description	Reset Value
UART_RS485DD **x=0,1,2,3,4,5,6,7,8,9**	UARTx_BA+0x4C	R/W	UART RS485 Transceiver Deactivate Delay Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RTSDDLY							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RTSDDLY							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**RTSDDLY**	**RS485 Transceiver Deactivate Delay Value** These bits field indicate how many clock cycles selected by UART_CLK do the UART controller delay the RS485 transceiver state trancing when the state trancing of RS485 transceiver is from TX to RX state. These bits field have no effect when the state trancing of RS485 transceiver is from RX to TX state. **Note:** It is valid only when RS485AUD (UART_ALTCTL[10]) is set.

6.24 Ethernet MAC Controller (EMAC)

6.24.1 Overview

Portions Copyright © Synopsys, Inc. Used with permission. All rights reserved.

The EMAC enables a host to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard.

In addition to the default interfaces defined in the IEEE 802.3 specifications, the EMAC supports several industry standard interfaces to the PHY. The EMAC is compliant with the following standards:

- IEEE 802.3-2008 for Ethernet MAC
- IEEE 1588-2008 standard for precision networked clock synchronization
- AMBA 2.0 for AHB master/slave port
- RMII specification version 1.2 from RMII consortium

6.24.2 Features

- 10 and 100 Mbps data transfer rates with the following PHY interfaces:
  - RMII interface to communicate with an external Fast Ethernet PHY
- Full-duplex operation:
  - IEEE 802.3x flow control automatic transmission of zero-quanta Pause frame on flow control input de-assertion
  - Forwarding of received Pause frames to the user application
- Half-duplex operation:
  - CSMA/CD Protocol support
  - Flow control using backpressure support
- Preamble and start of frame data (SFD) insertion in Transmit path
- Preamble and SFD deletion in the Receive path
- Automatic CRC and pad generation controllable on a per-frame basis
- Automatic Pad and CRC Stripping options for receive frames
- Programmable frame length to support Standard or Jumbo Ethernet frames with up to 16 Kbytes of size
- Programmable Interframe Gap (IFG) (40-96 bit times in steps of 8)
- Option to transmit frames with reduced preamble size
- Separate 32-bit status for transmit and receive packets
- IEEE 802.1Q VLAN tag detection for reception frames
- Separate transmission, reception, and control interfaces to the application
- Receive module for checksum off-load for received IPv4 and TCP packets encapsulated by the Ethernet frame (Type 1)
- Enhanced Receive module for checking IPv4 header checksum and TCP, UDP, or ICMP checksum encapsulated in IPv4 or IPv6 datagrams (Type 2)
- Support Ethernet frame timestamping as described in IEEE 1588-2002 and IEEE 1588-2008.

  The 64-bit timestamps are given in the transmit or receive status of each frame

- MDIO master interface for PHY device configuration and management
- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet
- Flexibility to control the Pulse-Per-Second (PPS) output signal (ptp_pps_o)
- CRC replacement, Source Address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted frames with per-frame control
- Programmable watchdog timeout limit in the receive path

6.24.3 Architecture

6.24.3.1 Block Diagram

The EMAC is composed of 4 main blocks:

- The **Direct Memory Access (DMA)** interface
- The **MAC Transaction Layer (MTL)** transfers data to system memory through the AHB master interface.
- The host CPU uses the AHB slave interface to access the EMAC subsystem's **Control and Status registers (CSRs) and Operating Mode Registers (OMRs)**
- The **Media Access Control (MAC)** module

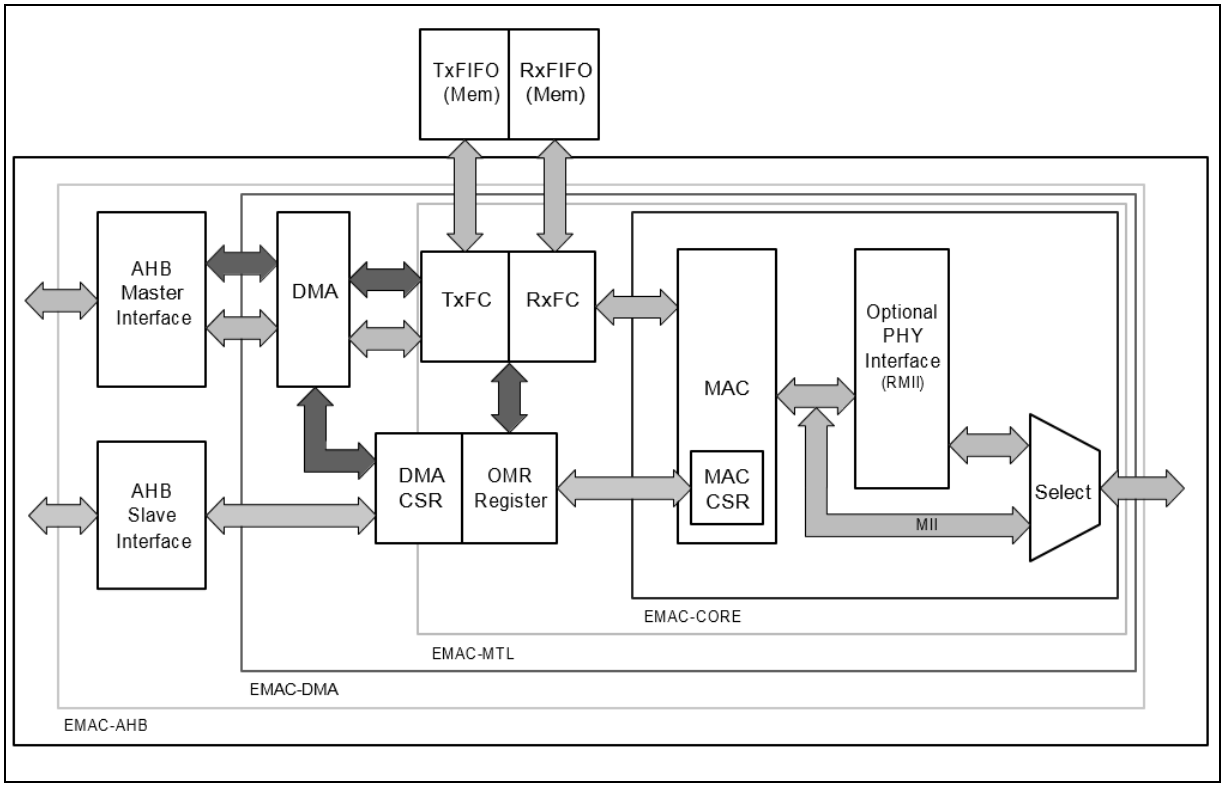


Figure 6.24-1 EMAC Block Diagram

6.24.3.2 DMA Controller

The DMA has independent Transmit and Receive engines, and a CSR space. The Transmit engine transfers data from system memory to the device port (MTL), while the Receive engine transfers data from the device port to the system memory. The controller use descriptors to efficiently move data from source to destination with minimal Host CPU intervention. The DMA is designed for packet-oriented data transfers such as frames in Ethernet. The controller can be programmed to interrupt the Host CPU

for situations such as Frame Transmit and Receive transfer completion, and other normal/error conditions.

The DMA and the Host driver communicate through two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

Control and Status registers are described in detail in “Register Description”. Descriptors are described in detail in “Descriptors”.

The DMA transfers data frames received by the MAC to the Receive Buffer in the Host memory, and Transmit data frames from the Transmit Buffer in the Host memory. Descriptors that reside in the Host memory act as pointers to these buffers.

There are two descriptor lists; one for reception, and one for transmission. The base address of each list is written into Register 1003 (Receive Descriptor List Address Register) and Register 1004 (Transmit Descriptor List Address Register), respectively. A descriptor list is forward linked (either implicitly or explicitly). The last descriptor may point back to the first entry to create a ring structure. Explicit chaining of descriptors is accomplished by setting the second address chained in both Receive and Transmit descriptors (RDES1[24] and TDES1[24]). The descriptor lists reside in the Host physical memory address space. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used, physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the Host physical memory space, and consists of an entire frame or part of a frame, but cannot exceed a single frame. Buffers contain only data, buffer status is maintained in the descriptor. Data chaining refers to frames that span multiple data buffers. However, a single descriptor cannot span multiple frames. The DMA skips to the next frame buffer when the end of frame is detected. Data chaining can be enabled or disabled.

The descriptor ring and chain structure is shown in Figure 6.24-2.

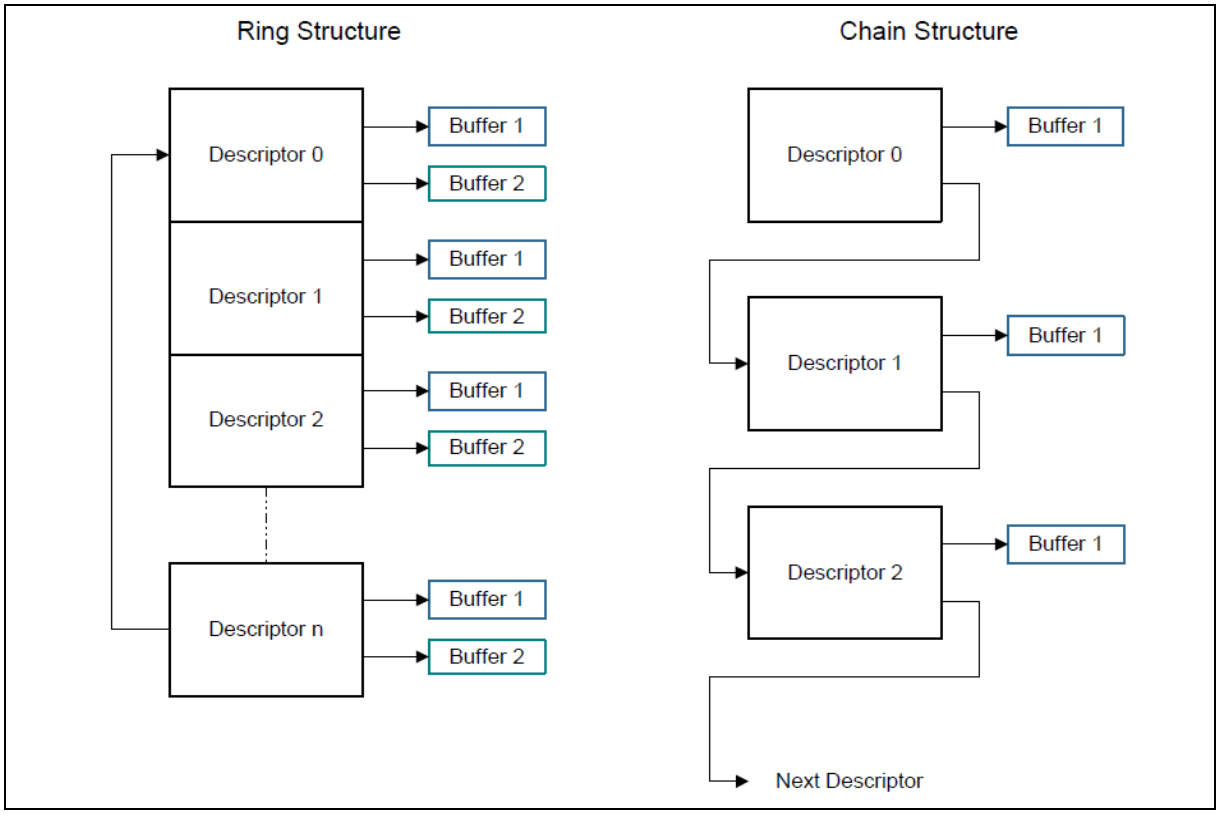


Figure 6.24-2 Descriptor Ring and Chain Structure

Initialization

Initialization for the EMAC is as follows:

1. Write to Register 1000 (Bus Mode Register) to set Host bus access parameters.
2. Write to Register 1007 (Interrupt Enable Register) to mask unnecessary interrupt causes.
3. Create the Transmit and Receive descriptor lists, and then write to DMA Register 1003 (Receive Descriptor List Address Register) and Register 1004 (Transmit Descriptor List Address Register), providing the DMA with the starting address of each list.
4. Write to Register 1 (MAC Frame Filter) for desired filtering options.
5. Write to Register 0 (MAC Configuration Register) to configure the operating mode and enable the transmit operation (Bit 3).
6. Program Bit 11 (DM) of Register 0 (MAC Configuration Register) based on the line status received or read from the PHY status registers after auto-negotiation.
7. Write to Register 1006 (Operation Mode Register) to set Bits 13 and 1 to start transmission and reception.
8. Write to Register 0 (MAC Configuration Register) to enable the Receive operation (Bit 2).

   The Transmit and Receive engines enter the Running state and attempt to acquire descriptors from the respective descriptor lists. The Receive and Transmit engines then begin processing Receive and Transmit operations. The Transmit and Receive processes are independent of each other and can be started or stopped separately.

Host Bus Burst Access

The DMA attempts to execute fixed-length Burst transfers on the AHB master interface if configured to do so through FB bit of Register 1000 (Bus Mode Register). The maximum Burst length is indicated and limited by the PBL field (Bits [13:8]) Register 1000 (Bus Mode Register). The Receive and Transmit descriptors are always accessed in the maximum possible (limited by PBL or 16 * 8/bus width) burst-size for the 16- bytes to be read.

The Transmit DMA initiates a data transfer only when sufficient space to accommodate the configured burst is available in MTL Transmit FIFO or the number of bytes till the end of frame (when it is less than the configured burst length). The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB Interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4, 8, or 16 and SINGLE transactions. Otherwise (no fixed-length burst), it transfers data using INCR (undefined length) and SINGLE transactions.

The Receive DMA initiates a data transfer only when sufficient data to accommodate the configured burst is available in MTL Receive FIFO or when the end of frame (when it is less than the configured burst length) is detected in the Receive FIFO. The DMA indicates the start address and the number of transfers required to the AHB master interface. When the AHB Interface is configured for fixed-length burst, then it transfers data using the best combination of INCR4, 8, or 16 and SINGLE transactions. If the end-of frame is reached before the fixed-burst ends on the AHB interface, then dummy transfers are performed in- order to complete the fixed-burst. Otherwise (FB bit of Register 1000 (Bus Mode Register) is reset), it transfers data using INCR (undefined length) and SINGLE transactions.

When the AHB interface is configured for address-aligned beats, both DMA engines ensure that the first burst transfer the AHB initiates is less than or equal to the size of the configured PBL. Thus, all subsequent beats start at an address that is aligned to the configured PBL. The DMA can only align the address for beats up to size 16 (for PBL > 16), because the AHB interface does not support more than INCR16.

Host Data Buffer Alignment

The Transmit and Receive data buffers do not have any restrictions on start address alignment. For example, in systems with 32-bit memory, the start address for the buffers can be aligned to any of the four bytes. However, the DMA always initiates write transfers, with address aligned to the bus width, with dummy data (old data) in the byte lanes that are not valid. This typically happens during the transfer

of the beginning or end of an Ethernet frame. The software driver should discard the dummy bytes based on the start address of the buffer and size of the frame.

Buffer Size Calculations

The DMA does not update the size fields in the Transmit and Receive descriptors. The DMA updates only the status fields (RDES and TDES) of the descriptors. The driver has to perform the size calculations.

The Transmit DMA transfers the exact number of bytes (indicated by buffer size field of TDES1) towards the MAC. If a descriptor is marked as first (FS bit of TDES1 is set), then the DMA marks the first transfer from the buffer as the start of frame. If a descriptor is marked as last (LS bit of TDES1), then the DMA marks the last transfer from that data buffer as the end-of frame to the MTL.

The Receive DMA transfers data to a buffer until the buffer is full or the end-of frame is received from the MTL. If a descriptor is not marked as last (LS bit of RDES0), then the descriptor's corresponding buffer(s) are full and the amount of valid data in a buffer is accurately indicated by its buffer size field minus the data buffer pointer offset when the FS bit of that descriptor is set. The offset is zero when the data buffer pointer is aligned to the data bus width. If a descriptor is marked as last, then the buffer may not be full (as indicated by the buffer size in RDES1). To compute the amount of valid data in this final buffer, the driver must read the frame length (FL bits of RDES0[29:16]) and subtract the sum of the buffer sizes of the preceding buffers in this frame. The Receive DMA always transfers the start of next frame with a new descriptor.

Even when the start address of a receive buffer is not aligned to the data width of system bus, the system should allocate a receive buffer of a size aligned to the system bus width. For example, if the system allocates a 1,024-byte (1 Kbytes) receive buffer starting from address 0x1000, the software can program the buffer start address in the Receive descriptor to have a 0x1002 offset. The Receive DMA writes the frame to this buffer with dummy data in the first two locations (0x1000 and 0x1001). The actual frame is written from location 0x1002. Thus, the actual useful space in this buffer is 1,022 bytes, even though the buffer size is programmed as 1,024 bytes, because of the start address offset.

DMA Arbiter

The arbiter inside the DMA module performs the round-robin arbitration between the Transmit and Receive channel accesses to the AHB master interface.

Transmission

Tx DMA Operation: Default (Non-OSF) Mode

The Transmit DMA engine by default mode proceeds as follows:

1. The host sets up the transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Frame data.
2. When Bit 13 (ST) of Register 1006 (Operation Mode Register) is set, the DMA enters the Run state.
3. While in the Run state, the DMA polls the Transmit Descriptor list for frames requiring transmission. After polling starts, it continues in either sequential descriptor ring order or chained order. If the DMA detects a descriptor flagged as owned by the Host (TDES0[31] = 1'b0), or if an error condition occurs, transmission is suspended and both the Bit 2 (Transmit Buffer Unavailable) and Bit 16 (Normal Interrupt Summary) of the Register 1005 (Status Register) are set. The Transmit Engine proceeds to step 9.
4. If the acquired descriptor is flagged as owned by DMA (TDES0[31] = 1'b1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
5. The DMA fetches the Transmit data from the Host memory and transfers the data to the MTL for transmission.

6. If an Ethernet frame is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3, 4, and 5 are repeated until the end-of- Ethernet-frame data is transferred to the MTL.
7. When frame transmission is complete, if IEEE 1588 timestamping was enabled for the frame (as indicated in the transmit status) the timestamp value obtained from MTL is written to the transmit descriptor (TDES2 and TDES3) that contains the end-of-frame buffer. The status information is then written to this transmit descriptor (TDES0). Because the Own bit is cleared during this step, the Host now owns this descriptor. If timestamping was not enabled for this frame, the DMA does not alter the contents of TDES2 and TDES3.
8. Bit 0 (Transmit Interrupt) of Register 1005 (Status Register) is set after completing transmission of a frame that has Interrupt on Completion (TDES1[31]) set in its Last Descriptor. The DMA engine then returns to step 3.
9. In the Suspend state, the DMA tries to reacquire the descriptor (and thereby return to step 3) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the host stopped the DMA by clearing Bit 13 (ST) of Register 1006 (Operation Mode Register), the DMA enters the STOP state.

The Tx DMA transmission flow by default mode is shown in Figure 6.24-3.

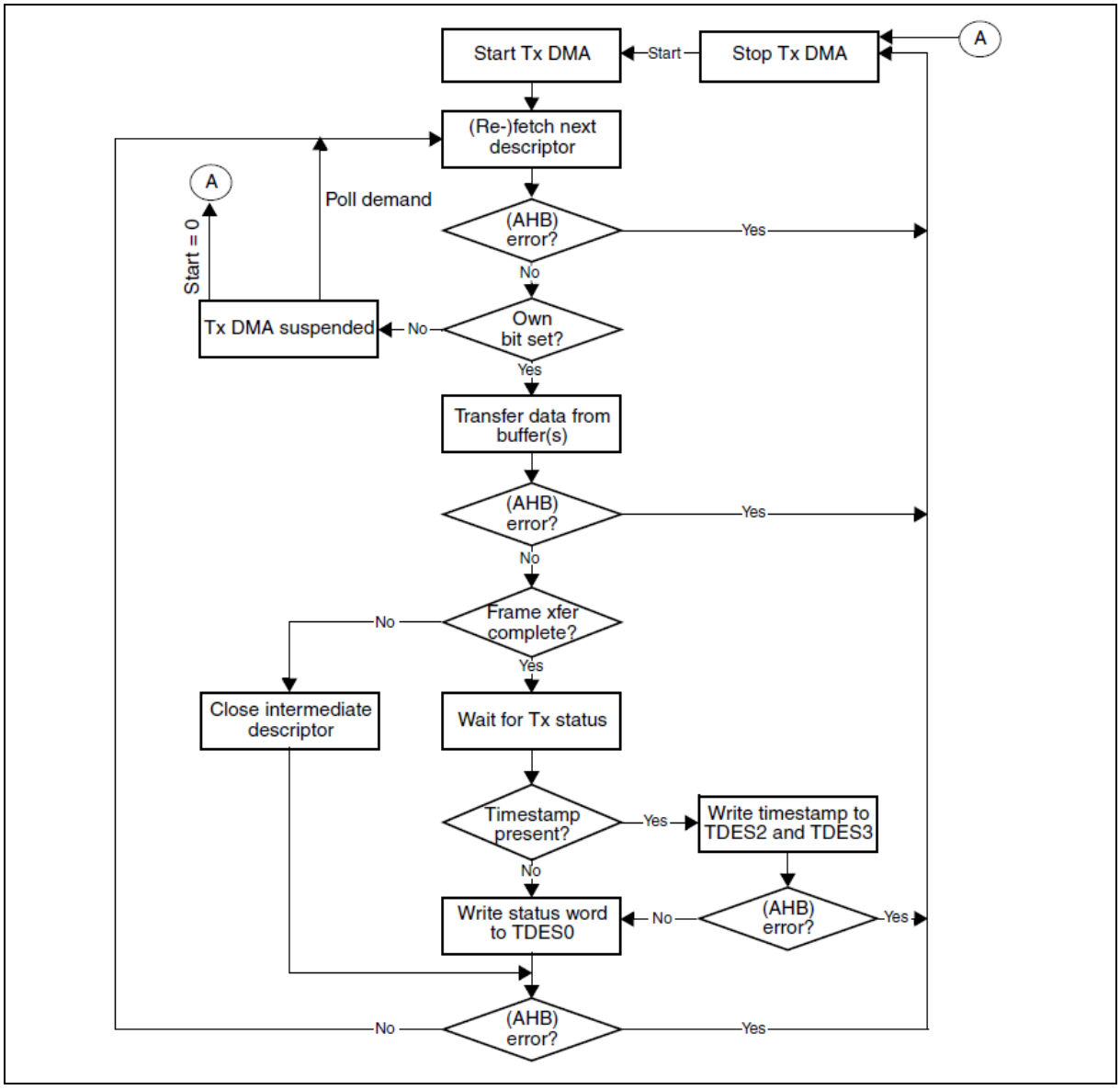


Figure 6.24-3 Tx DMA Operation by default Mode

Tx DMA Operation: OSF Mode

While in the Run state, the transmit process can simultaneously acquire two frames without closing the Status descriptor of the first [if Bit 2 (OSF) in Register 1006 (Operation Mode Register) is set]. As the transmit process finishes transferring the first frame, it immediately polls the Transmit Descriptor list for the second frame. If the second frame is valid, the transmit process transfers this frame before writing the first frame's status information.

In OSF mode, the Run State Transmit DMA operates in the following sequence:

1. The DMA operates as described in steps 1–6 of the Tx DMA (default mode).
2. Without closing the previous frame's last descriptor, the DMA fetches the next descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and skips to step 7.
4. The DMA fetches the Transmit frame from the Host memory and transfers the frame to the MTL

until the end-of-frame data is transferred, closing the intermediate descriptors if this frame is split across multiple descriptors.

5. The DMA waits for the previous frame's frame transmission status and timestamp. When the status is available, the DMA writes the timestamp to TDES2 and TDES3, if such timestamp was captured (as indicated by a status bit). The DMA then writes the status, with a cleared Own bit, to the corresponding TDES0, thus closing the descriptor. If timestamping was not enabled for the previous frame, the DMA does not alter the contents of TDES2 and TDES3.
6. If enabled, the Transmit interrupt is set, the DMA fetches the next descriptor, then proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA writes the timestamp (if enabled for the current frame) to TDES2 and TDES3, then writes the status to the corresponding TDES0. It then sets relevant interrupts and returns to Suspend mode.

   If no status is pending and the host stopped the DMA by clearing Bit 13 (ST) of Register 1006 (Operation Mode Register), the DMA enters the STOP state.
8. The DMA can exit Suspend mode and enter the Run state (go to step 1 or step 2 depending on pending status) only after receiving a Transmit Poll demand (Register 1001 (Transmit Poll Demand Register)).

The DMA fetches the next descriptor in advance before closing the current descriptor. Therefore, the descriptor chain should have more than two different descriptors for correct and proper operation.

The basic flow is described in Figure 6.24-4.

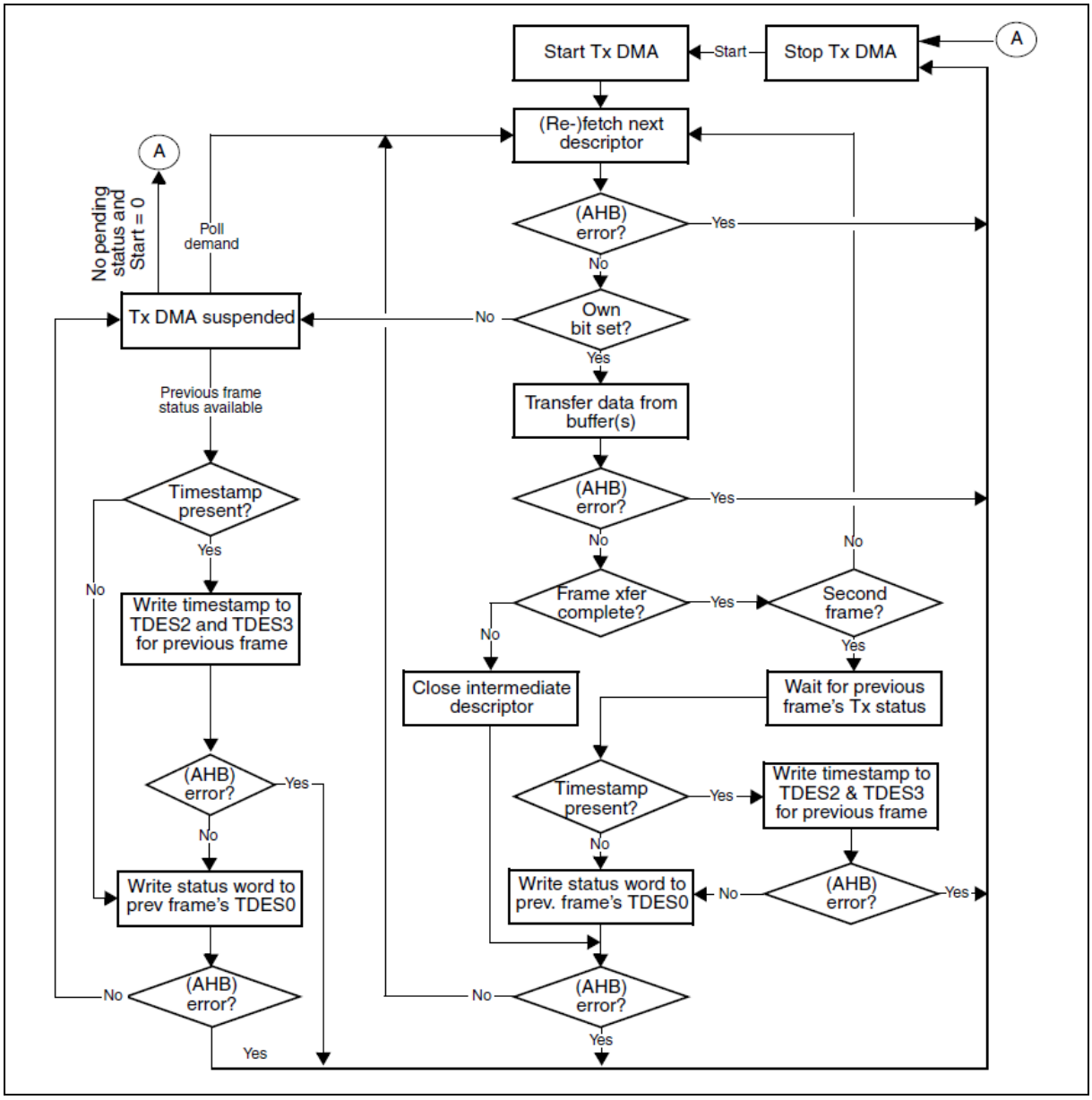


Figure 6.24-4 Tx DMA Operation in OSF Mode

Transmit Frame Processing

The Transmit DMA expects that the data buffers contain complete Ethernet frames, excluding preamble, pad bytes, and FCS fields. The DA, SA, and Type/Length fields contain valid data. If the Transmit Descriptor indicates that the MAC must disable CRC or PAD insertion, the buffer must have complete Ethernet frames (excluding preamble), including the CRC bytes.

Frames can be data-chained and can span several buffers. Frames must be delimited by the First Descriptor (TDES1[29]) and the Last Descriptor (TDES1[30]), respectively.

As transmission starts, the First Descriptor must have (TDES1[29]) set. When this occurs, frame data transfers from the Host buffer to the MTL Transmit FIFO. Concurrently, if the current frame has the Last Descriptor (TDES1[30]) clear, the Transmit Process attempts to acquire the Next Descriptor. The Transmit Process expects this descriptor to have TDES1[29] clear. If TDES1[30] is clear, it indicates an intermediary buffer. If TDES1[30] is set, it indicates the last buffer of the frame.

After the last buffer of the frame has been transmitted, the DMA writes back the final status information to the Transmit Descriptor 0 (TDES0) word of the descriptor that has the last segment set in Transmit Descriptor 1 (TDES1[30]). At this time, if Interrupt on Completion (TDES1[31]) is set, the Bit 0 (Transmit Interrupt) of Register 1005 (Status Register) is set, the Next Descriptor is fetched, and the process repeats.

The actual frame transmission begins after the MTL Transmit FIFO has reached either a programmable transmit threshold (Bits [16:14] of Register 1006 (Operation Mode Register)), or a full frame is contained in the FIFO. There is also an option for Store and Forward Mode (Bit 21 of Register 1006 (Operation Mode Register)). Descriptors are released (Own bit TDES0[31] clears) when the DMA finishes transferring the frame.

To ensure proper transmission of a frame and the next frame, you must specify a non-zero buffer size for the transmit descriptor that has the Last Descriptor (TDES1[30]) set.

Transmit Polling Suspended

Transmit polling can be suspended by either of the following conditions:

- The DMA detects a descriptor owned by the Host (TDES0[31]=0). To resume, the driver must give descriptor ownership to the DMA and then issue a Poll Demand command.
- A frame transmission is aborted when a transmit error because of underflow is detected. The appropriate Transmit Descriptor 0 (TDES0) bit is set.

If the DMA goes into SUSPEND state because of the first condition, then both Bit 16 (Normal Interrupt Summary) and Bit 2 (Transmit Buffer Unavailable) of Register 1005 (Status Register) are set. If the second condition occurs, both Bit 15 (Abnormal Interrupt Summary) and Bit 5 (Transmit Underflow) of Register 1005 (Status Register) are set, and the information is written to Transmit Descriptor 0, causing the suspension.

In both cases, the position in the Transmit List is retained. The retained position is that of the descriptor following the Last Descriptor closed by the DMA.

The driver must explicitly issue a Transmit Poll Demand command after rectifying the suspension cause.

Reception

The Receive DMA engine's reception sequence is depicted in Figure 6.24-5 and proceeds as follows:

1. The host sets up receive descriptors (RDES0-RDES3) and sets the Own bit (RDES0[31]).
2. When Bit 1 (SR) of Register 1006 (Operation Mode Register) is set, the DMA enters the Run state. While in the Run state, the DMA polls the Receive Descriptor list, attempting to acquire free descriptors. If the fetched descriptor is not free (is owned by the host), the DMA enters the Suspend state and jumps to step 9.
3. The DMA decodes the receive data buffer address from the acquired descriptors.
4. Incoming frames are processed and placed in the acquired descriptor's data buffers.
5. When the buffer is full or the frame transfer is complete, the Receive engine fetches the next descriptor.
6. If the current frame transfer is complete, the DMA proceeds to step 7. If the DMA does not own the next fetched descriptor and the frame transfer is not complete (EOF is not yet transferred), the DMA sets the Descriptor Error bit in the RDES0 (unless flushing is disabled in Bit 24 of Register 1006 (Operation Mode Register)). The DMA closes the current descriptor (clears the Own bit) and marks it as intermediate by clearing the Last Segment (LS) bit in the RDES0 value (marks it as Last Descriptor if flushing is not disabled), then proceeds to step 8. If the DMA does own the next descriptor but the current frame transfer is not complete, the DMA closes the current descriptor as intermediate and reverts to step 4.
7. If IEEE 1588 timestamping is enabled, the DMA writes the timestamp (if available) to the current descriptor's RDES2 and RDES3. It then takes the receive frame's status from the MTL and

writes the status word to the current descriptor's RDES0, with the Own bit cleared and the Last Segment bit set.

If the host stopped the Rx DMA by clearing Bit 1 (SR) of Register 1006 (Operation Mode Register), the DMA goes to the Stop state. Otherwise, the Rx DMA proceeds to the step 8.

8. The Receive engine checks the latest descriptor's Own bit. If the host owns the descriptor (Own bit is 1'b0), the Bit 7 (Receive Buffer Unavailable) of Register 1005 (Status Register) is set and the DMA Receive engine enters the Suspended state (step 9). If the DMA owns the descriptor, the engine returns to step 4 and awaits the next frame.
9. Before the Receive engine enters the Suspend state, partial frames are flushed from the Receive FIFO. You can control flushing using Bit 24 of Register 1006 (Operation Mode Register).
10. The Receive DMA goes to the STOP state if the host had cleared Bit 1 (SR) of Register 1006 (Operation Mode Register). Otherwise, it exits the Suspend state when a Receive Poll demand is given or the start of next frame is available from the Receive FIFO of the MTL. The engine proceeds to step 2 and refetches the next descriptor.

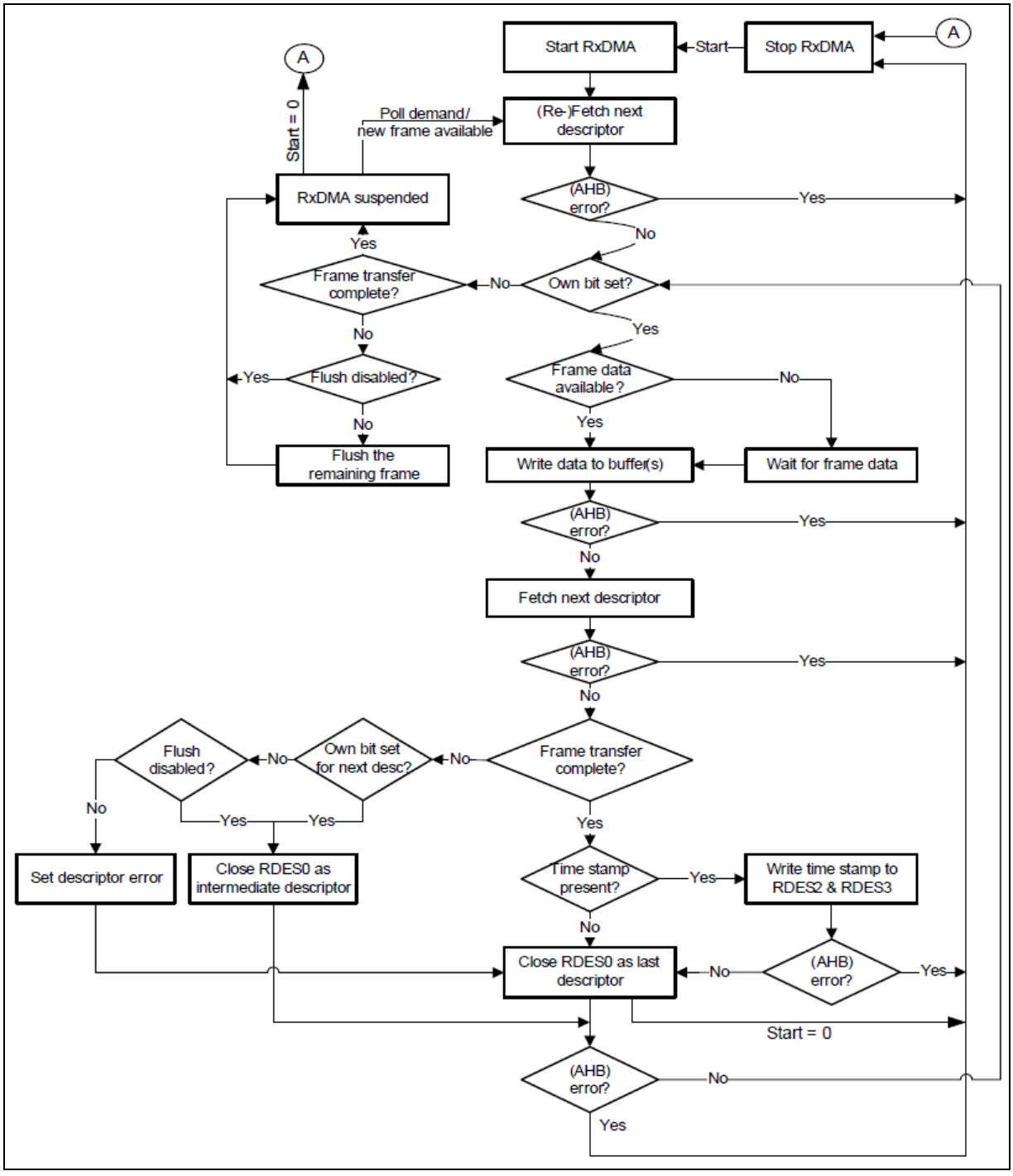


Figure 6.24-5 Receive DMA Operation

The DMA does not acknowledge accepting the status from the MTL until it has completed the timestamp write-back and is ready to perform status write-back to the descriptor.

If software has enabled timestamping through CSR, when a valid timestamp value is not available for the frame (for example, because the receive FIFO was full before the timestamp could be written to it), the DMA writes all-ones to RDES2 and RDES3. Otherwise (that is, if timestamping is not enabled), the RDES2 and RDES3 remain unchanged.

Receive Descriptor Acquisition

The Receive Engine always attempts to acquire an extra descriptor in anticipation of an incoming frame. Descriptor acquisition is attempted if any of the following conditions is satisfied:

- Bit 1 (Start or Stop Receive) of Register 1006 (Operation Mode Register) has been set immediately after being placed in the Run state.
- The data buffer of current descriptor is full before the frame ends for the current transfer.
- The controller has completed frame reception, but the current Receive Descriptor is not yet closed.
- The receive process has been suspended because of a host-owned buffer (RDES0[31] = 0) and a new frame is received.
- A Receive poll demand has been issued.

Receive Frame Processing

The MAC transfers the received frames to the Host memory only when the frame passes the address filter and frame size is greater than or equal to configurable threshold bytes set for the Receive FIFO of MTL, or when the complete frame is written to the FIFO in the store-and-forward mode.

If the frame fails the address filtering, it is dropped in the MAC block itself (unless Bit 31 (Receive All) of Register 1 (MAC Frame Filter) is set). Frames that are shorter than 64 bytes, because of collision or premature termination, can be removed from the MTL Receive FIFO.

After 64 (configurable threshold) bytes have been received, the MTL block requests the DMA block to begin transferring the frame data to the Receive Buffer pointed by the current descriptor. The DMA sets First Descriptor (RDES0[9]) after the DMA Host Interface (AHB or MDC) becomes ready to receive a data transfer (if DMA is not fetching transmit data from the host), to delimit the frame. The descriptors are released when the Own (RDES[31]) bit is reset to 1'b0, either as the Data buffer fills up or as the last segment of the frame is transferred to the Receive buffer. If the frame is contained in a single descriptor, both Last Descriptor (RDES[8]) and First Descriptor (RDES[9]) are set.

The DMA fetches the next descriptor, sets the Last Descriptor (RDES[8]) bit, and releases the RDES0 status bits in the previous frame descriptor. Then the DMA sets the Bit 6 (Receive Interrupt) of Register 1005 (Status Register). The same process repeats unless the DMA encounters a descriptor flagged as being owned by the host. If this occurs, the Receive Process sets the Bit 7 (Receive Buffer Unavailable) of Register 1005 (Status Register) and then enters the Suspend state. The position in the receive list is retained.

Receive Process Suspended

If a new Receive frame arrives while the Receive Process is in Suspend state, the DMA refetches the current descriptor in the Host memory. If the descriptor is now owned by the DMA, the Receive Process re-enters the Run state and starts frame reception. If the descriptor is still owned by the host, by default, the DMA discards the current frame at the top of the MTL Rx FIFO and increments the missed frame counter. If more than one frame is stored in the MTL Rx FIFO, the process repeats.

The discarding or flushing of the frame at the top of the MTL Rx FIFO can be avoided by disabling Flushing (Bit 24 of Register 1006 (Operation Mode Register)). In such conditions, the receive process sets the Receive Buffer Unavailable status and returns to the Suspend state.

Interrupts

Interrupts can be generated as a result of various events. The DMA Register 1005 (Status Register) contains all the bits that might cause an interrupt. Register 1007 (Interrupt Enable Register) contains an enable bit for each of the events that can cause an interrupt.

There are two groups of interrupts, Normal and Abnormal, as described in Register 1005 (Status Register). Interrupts are cleared by writing a 1'b1 to the corresponding bit position. When all the enabled interrupts within a group are cleared, the corresponding summary bit is cleared. When both the summary

bits are cleared, the interrupt signal is de-asserted. If the MAC is the cause for assertion of the interrupt, then any of the GPI or TTI bits of Register 1005 (Status Register) are set high.

The Register 1005 (Status Register) is the (interrupt) status register. The interrupt pin is asserted because of any event in this status register only if the corresponding interrupt enable bit is set in Register 1007 (Interrupt Enable Register).

Interrupts are not queued and if the interrupt event occurs before the driver has responded to it, no additional interrupts are generated. For example, Receive Interrupt [Bit 6 of Register 1005 (Status Register)] indicates that one or more frames were transferred to the Host buffer. The driver must scan all descriptors, from the last recorded position to the first one owned by the DMA.

An interrupt is generated only once for simultaneous, multiple events. The driver must scan the Register 1005 (Status Register) for the cause of the interrupt. The interrupt is not generated again unless a new interrupting event occurs, after the driver has cleared the appropriate bit in Register 1005 (Status Register). For example, the controller generates a Receive interrupt [Bit 6 of Register 1005 (Status Register)] and the driver begins reading Register 1005 (Status Register). Next, Receive Buffer Unavailable [Bit 7 of Register 1005 (Status Register)] occurs. The driver clears the Receive interrupt. Even then, the interrupt pin signal is not de-asserted, because of the active or pending Receive Buffer Unavailable interrupt.

An interrupt timer RIWT (Bits 7:0 in Register 1009 (Receive Interrupt Watchdog Timer Register)) is given for flexible control of Receive Interrupt. When this Interrupt timer is programmed with a non-zero value, it gets activated as soon as the Rx DMA completes a transfer of a received frame to system memory without asserting the Receive Interrupt because it is not enabled in the corresponding Receive Descriptor (RDES1[31] in Table 6.24-19). When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RI is enabled in Register 1007 (Interrupt Enable Register). This timer gets disabled before it runs out, when a frame is transferred to memory and the RI is set because it is enabled for that descriptor.

Error Response to DMA

For any data transfer initiated by a DMA channel, if the slave replies with an error response, that DMA stops all operations and updates the error bits and the Fatal Bus Error bit in the Register 1005 (Status Register). The DMA controller can resume operation only after soft resetting or hard resetting the EMAC and re-initializing the DMA.

6.24.4 Basic Configuration

- Clock Source Configuration
  - Enable EMAC0 controller clock in EMAC0CKEN (CLK_AHBCLK0[5])
- Reset Configuration
  - Reset EMAC0 controller in EMAC0RST (SYS_IPRST0[5])

6.24.5 Functional Description

6.24.5.1 IEEE 1588-2002 Timestamps

The IEEE 1588-2002 standard defines a protocol, Precision Time Protocol (PTP), which enables precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the sub-microsecond range with minimal network and local clock computing resources.

The PTP is transported over UDP/IP. The system or network is classified into master and slave nodes for distributing the timing and clock information. Figure 6.24-6 shows the process that PTP uses for synchronizing a slave node to a master node by exchanging PTP messages.

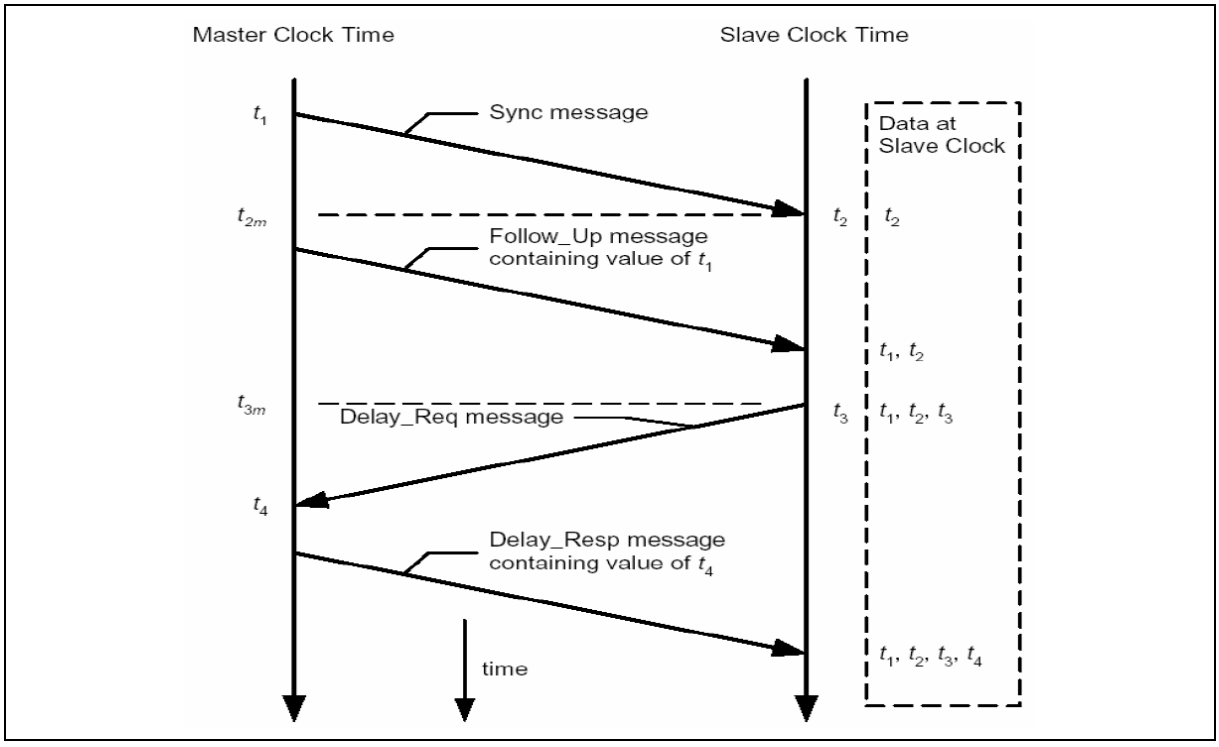


Figure 6.24-6 Networked Time Synchronization

As shown in Figure 6.24-6, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the master's reference time information. The time at which this message leaves the master's system is t1. This time must be captured, for Ethernet ports, at MII.
2. The slave receives the Sync message and also captures the exact time, t2, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t1 information for later use.
4. The slave sends a Delay_Req message to the master, noting the exact time, t3, at which this frame leaves the MII.
5. The master receives the message, capturing the exact time, t4, at which it enters its system.
6. The master sends the t4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t1, t2, t3, and t4 to synchronize its local timing reference to the master's timing reference.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII. This timing information must be captured and returned to the software for the proper implementation of PTP with high accuracy.

Reference Timing Source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588 specification. The EMAC provides the following two options for using the reference timing source in a node:

- External Timestamp Input Option

This option takes an external 64-bit timing reference and its clock as input. The clock input is used to synchronize the timing reference to the MAC clock domain. The 64-bit timing reference is split into the following two 32-bit signals:

- Upper 32-bits providing the time in seconds
- Lower 32-bits providing the time in nanoseconds

This timing reference is used to timestamp the frames.

- Internal Reference Time Option

  This option takes only the reference clock input and uses it to generate the Reference time (also called the System Time) internally and capture timestamps. The generation, update, and modification of the System Time are described in “System Time Register Module”.

System Time Register Module

The System Time Generator module is optional and is not available if external time updating is enabled. The 64-bit time is maintained in this module and updated using the input reference clock (clk_ptp_ref_i).

This time is the source for taking snapshots (timestamps) of Ethernet frames being transmitted or received at the MII.

The System Time counter can be initialized or corrected using the coarse correction method. In this method, the initial value or the offset value is written to the Timestamp Update register (“IEEE 1588 Timestamp Registers”). For initialization, the System Time counter is written with the value in the Timestamp Update registers, while for system time correction, the offset value is added to or subtracted from the system time.

In the fine correction method, a slave clock’s (clk_ptp_ref_i) frequency drift with respect to the master clock (as defined in IEEE 1588) is corrected over a period of time instead of in one clock, as in coarse correction. This helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register, as shown in Figure 6.24-7. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. Here, the accumulator acts as a high-precision frequency multiplier or divider.

This algorithm is depicted in Figure 6.24-7.

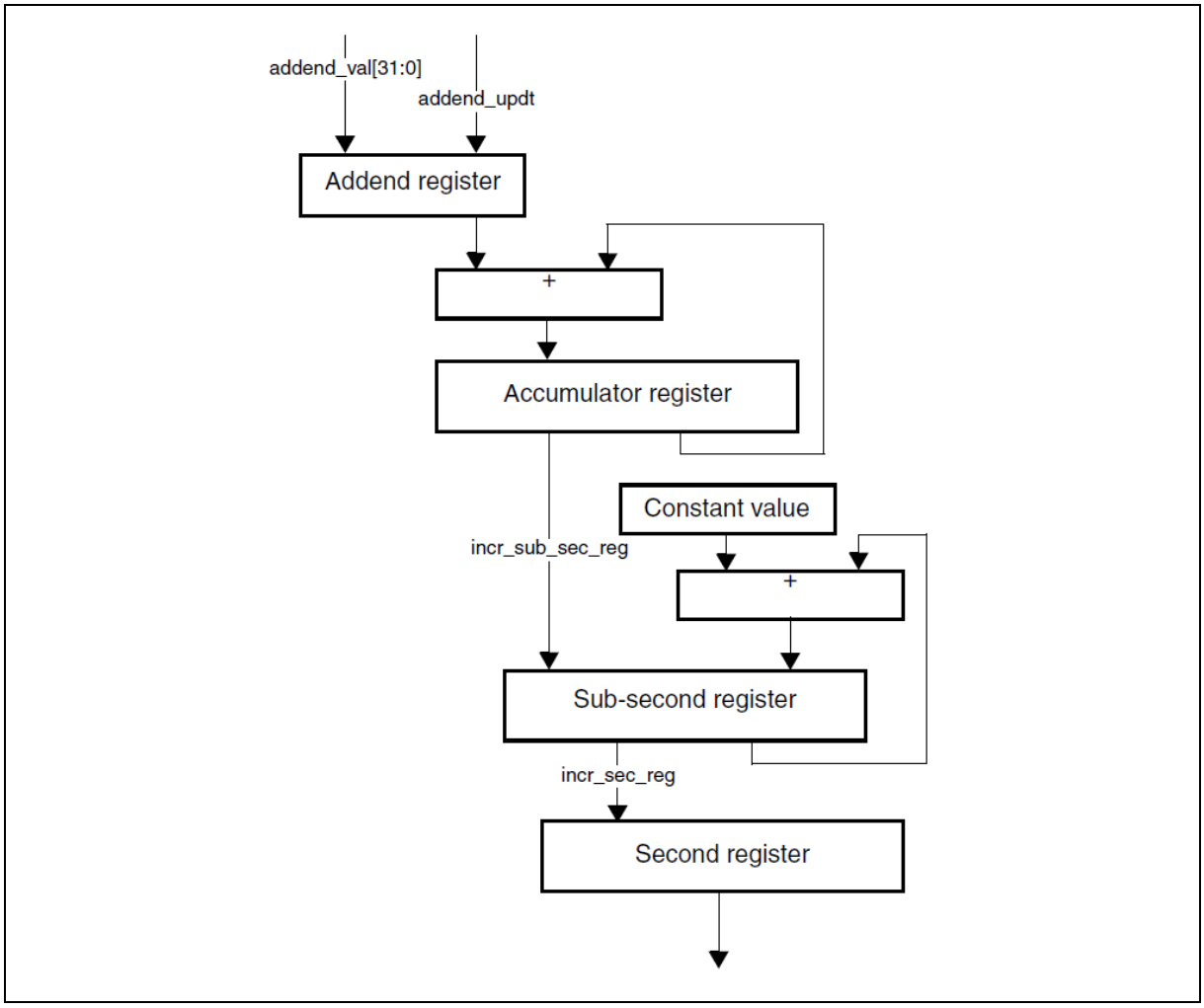


Figure 6.24-7 System Time Update Using Fine Method

The System Time Update logic requires a 50- MHz clock frequency to achieve 20-ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. Hence, if the reference clock (clk_ptp_ref_i) is, for example, 66 MHz, this ratio is calculated as 66 MHz / 50 MHz = 1.32. Hence, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, to 65 MHz for example, the ratio is 65 / 50, or 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC4EC4. If the clock drifts higher, to 67 MHz for example, the addend register must be set to 0xBF0B7672. When the clock drift is nil, the default addend value of 0xC1F07C1F (2^{32} / 1.32) must be programmed.

In Figure 6.24-7, the constant value used to accumulate the sub-second register is decimal 43, which achieves an accuracy of 20 ns in the system time (in other words, it is incremented in 20-ns steps).

The software must calculate the drift in frequency based on the Sync messages and update the Addend register accordingly.

Initially, the slave clock is set with $FreqCompensationValue_0$ in the Addend register. This value is as follows:

$$FreqCompensationValue_0 = 2^{32} / FreqDivisionRatio$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm described below must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new

value.

The algorithm is as follows:

- At time $MasterSyncTime_n$ the master sends the slave clock a Sync message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as:

  $MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$

- The master clock count for current Sync cycle, $MasterClockCount_n$ is given by:

  $MasterClockCount_n = MasterClockTime_n – MasterClockTime_{n-1}$ (assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)

- The slave clock count for current Sync cycle, $SlaveClockCount_n$ is given by:

  $SlaveClockCount_n = SlaveClockTime_n – SlaveClockTime_{n-1}$

- The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is given by:

  $ClockDiffCount_n = MasterClockCount_n – SlaveClockCount_n$

- The frequency-scaling factor for slave clock, FreqScaleFactorn is given by:

  $FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$

- The frequency compensation value for Addend register, $FreqCompensationValue_n$ is given by:

  $FreqCompensationValue_n = FreqScaleFactor_n * FreqCompensationValue_{n-1}$

In theory, this algorithm achieves lock in one Sync cycle; however, it may take several cycles, because of changing network propagation delays and operating conditions.

This algorithm is self-correcting: if for any reason the slave clock is initially set to a value from the master that is incorrect, the algorithm corrects it at the cost of more Sync cycles.

Transmit Path Functions

The MAC captures a timestamp when the Start Frame Delimiter (SFD) of a frame is sent on MII. The frames for which you want to capture timestamps are controllable on a per-frame basis. In other words, each transmit frame can be marked to indicate whether a timestamp should be captured for that frame.

The MAC does not process the transmitted frames to identify the PTP frames. You need to specify the frames for which you want to capture timestamps. To return the captured timestamps to the application, the MAC uses a separate 64-bit bus. The MAC gives the timestamp, along with the Transmit status of the frame, on this bus.

You can use the control bits in the transmit descriptor (see “Descriptor Format With IEEE 1588-2005 Timestamps Enabled”). The MAC returns the timestamp to the software inside the corresponding transmit descriptor, thus connecting the timestamp automatically to the specific PTP frame. The 64-bit timestamp information is written to the TDES2 and TDES3 fields. The TDES2 field holds the 32 least significant bits of the timestamp, except as described in “Transmit Timestamp Field”.

When the alternate (enhanced) descriptor is selected, the MAC writes the 64-bit timestamp in TDES6 and TDES7, respectively.

Receive Path Functions

The MAC captures the timestamp of all frames received on the MII interface.

The DMA returns the timestamp to the software in the corresponding receive descriptor. The 64-bit timestamp information is written back to the RDES2 and RDES3 fields. The RDES2 holds the 32 least significant bits of the timestamp, except as mentioned in “Receive Timestamp”. The timestamp is written only to that receive descriptor for which the Last Descriptor status field has been set to 1 (the EOF

marker). When the timestamp is not available (for example, because of an Rx FIFO overflow), an all-ones pattern is written to the descriptors (RDES2 and RDES3), indicating that timestamp is not correct. If the software uses a control register bit to disable timestamping, the DMA does not alter RDES2 or RDES3.

When the alternate (enhanced) descriptor is selected, the 64-bit timestamp is written in RDES6 and RDES7, respectively. RDES0[7] indicates whether the timestamp is updated in RDES6 and RDES77 or not.

Timestamp Error Margin

According to the IEEE 1588 specifications, a timestamp must be captured at the SFD of the transmitted and received frames at the MII interface. Because the reference timing source (the PTP clock, clk_ptp_ref_i) is taken as different from the MII clocks, a small error margin is introduced, because of the transfer of information across asynchronous clock domains.

In the transmit path, the captured and reported timestamp has a maximum error margin of 2 PTP clocks. It means that the captured timestamp has the reference timing source value that is given within 2 clocks after the SFD has been transmitted on the GMII.

Similarly, in the receive path, the error margin is 3 MII clocks, plus up to 2 PTP clocks. You can ignore the error margin because of the three MII clocks by assuming that this constant delay is present in the system (or link) before the SFD data reaches the MII interface of MAC.

Frequency Range of Reference Timing Clock

The timestamp information is transferred across asynchronous clock domains, that is, from MAC clock domain to application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is 4 clock cycles of MII and 3 clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second frame.

The maximum PTP clock frequency is limited by the maximum resolution of the reference time (1 ns resulting in 1 GHz) and the timing constraints achievable for logic operating on the PTP clock. In addition, the resolution, or granularity, of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time to the MII clock domain. This relationship is given by the following equation:

```
3*PTP clock period + 4*MII clock period <= Minimum gap between two SFDs
```

Because the MII clock frequency is fixed by IEEE specification, the minimum PTP clock frequency required for proper operation depends upon the operating mode and operating speed of the MAC as shown in Table 6.24-1.

Mode	Minimum Gap Between Two SFDs	Minimum PTP Frequency With External Timestamp Input	Minimum PTP Frequency With Internal Timestamp
10-Mbps full duplex	168 MII clocks (128 clocks for a 64-byte frame + 24 clocks of min IFG + 16 clocks of preamble)	~45 kHz	5 MHz
10-Mbps half duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	170 kHz	5 MHz
100-Mbps full duplex	168 MII clocks (128 clocks for a 64-byte frame + 24 clocks of min IFG + 16 clocks of preamble)	~0.5 MHz	5 MHz

100-Mbps half duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	4.55 MHz	5 MHz

Table 6.24-1 Minimum PTP Clock Frequency Example

When IEEE1588 timestamping is enabled with internal timestamp, use PTP clock frequency which is greater than 5 MHz. This is because the 8-bit Register 449 (Sub- Second Increment Register) limits the minimum PTP frequency that can be used to ~4 MHz.

6.24.5.2 IEEE 1588-2008 Advanced Timestamps

In addition to the basic timestamp features mentioned in IEEE 1588-2002 Timestamps, the EMAC supports the following advanced timestamp features defined in the IEEE 1588-2008 standard.

- Supports the IEEE 1588-2008 (version 2) timestamp format.
- Provides an option to take snapshot of all frames or only PTP type frames.
- Provides an option to take snapshot of only event messages.
- Provides an option to take the snapshot based on the clock type: ordinary, boundary, end-to-end, and peer-to-peer.
- Provides an option to select the node to be a master or slave for ordinary and boundary clock.
- Identifies the PTP message type, version, and PTP payload in frames sent directly over Ethernet and sends the status.
- Provides an option to measure sub-second time in digital or binary format.

Peer-to-Peer PTP Transparent Clock (P2P TC) Message Support

The IEEE 1588-2008 version supports Peer-to-Peer PTP (Pdelay) message in addition to SYNC, Delay Request, Follow-up, and Delay Response messages. Figure 6.24-8 shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

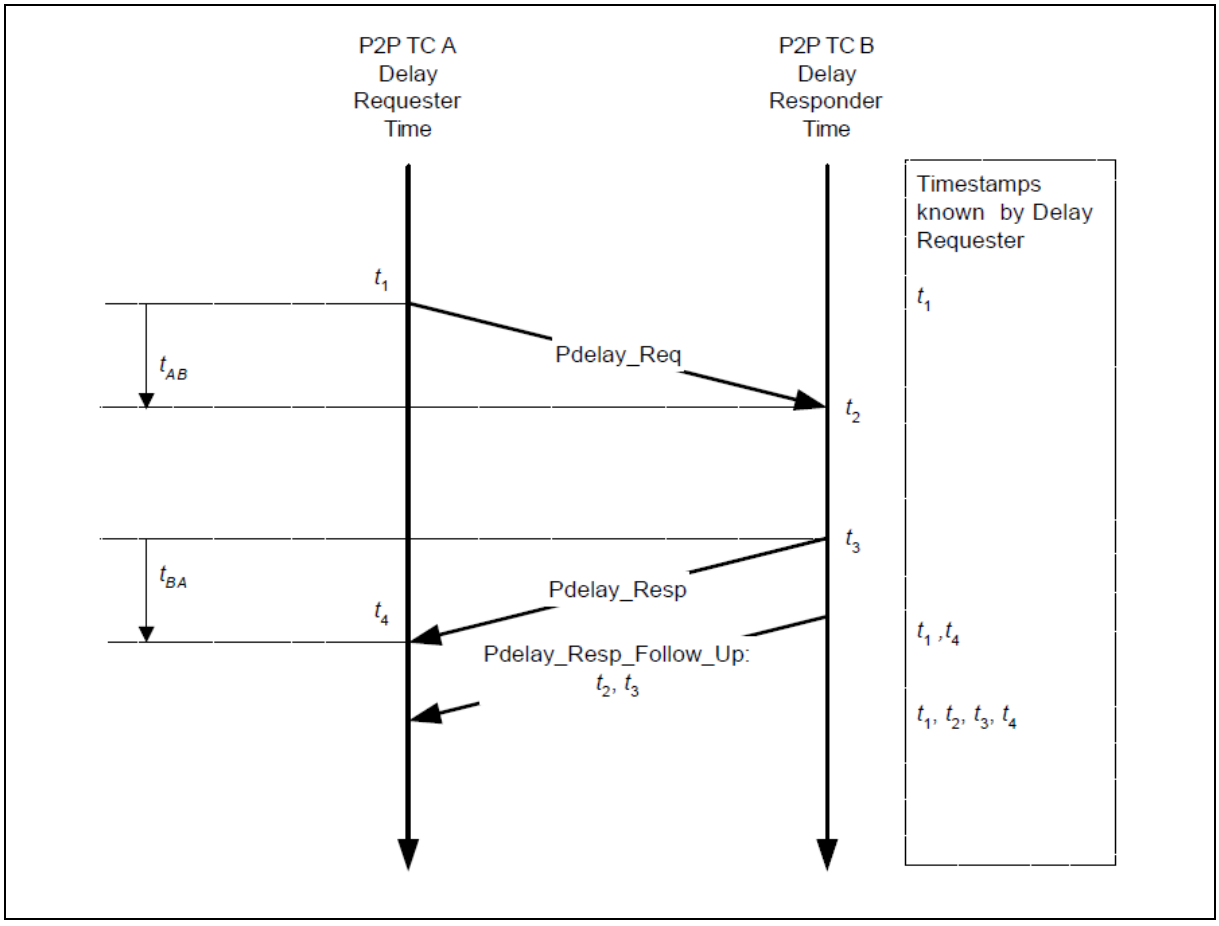


Figure 6.24-8 Propagation Delay Calculation in Clocks Supporting Peer-to-Peer Path Correction

As shown in Figure 6.24-8, the propagation delay is calculated in the following way:

1. Port-1 issues a Pdelay_Req message and generates a timestamp, t1, for the Pdelay_Req message.
2. Port-2 receives the Pdelay_Req message and generates a timestamp, t2, for this message.
3. Port-2 returns a Pdelay_Resp message and generates a timestamp, t3, for this message.

4. To minimize errors because of any frequency offset between the two ports, Port-2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. The Port-2 returns any one of the following:

   - The difference between the timestamps t2 and t3 in the Pdelay_Resp message.
   - The difference between the timestamps t2 and t3 in the Pdelay_Resp_Follow_Up message.
   - The timestamps t2 and t3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages respectively.

5. Port-1 generates a timestamp, t4, on receiving the Pdelay_Resp message.
6. Port-1 uses all four timestamps to compute the mean link delay.

Clock Types

The EMAC supports the following clock types defined in the IEEE 1588-2008 standard:

- Ordinary Clock

- Boundary Clock
- End-to-End Transparent Clock
- Peer-to-Peer Transparent Clock

Ordinary Clock

The ordinary clock in a domain supports a single copy of the protocol. The ordinary clock has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

- Sends and receives PTP messages. The timestamp snapshot can be controlled as described in "Register 448 (Timestamp Control Register)".
- Maintains the data sets such as timestamp values.

Table 6.24-2 shows the messages for which you can take the timestamp snapshot on the receive side for Master and slave nodes.

Master	Slave
Delay_Req	SYNC

Table 6.24-2 Ordinary Clock: PTP Messages for Snapshot

For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the control bit (TSVER2ENA) and selecting the snapshot mode in Register 448 (Timestamp Control Register).

Boundary Clock

The boundary clock typically has several physical ports communicating with the network. The messages related to synchronization, master-slave hierarchy, and signaling terminate in the protocol engine of the boundary clock and are not forwarded. The PTP message type status given by the MAC (see "Receive Path Functions") helps you to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

- The clock data sets are common to all ports of the boundary clock
- The local clock is common to all ports of the boundary clock.

Therefore, the features of the ordinary clock are also applicable to the boundary clock.

End-to-End Transparent Clock

The end-to-end transparent clock supports the end-to-end delay measurement mechanism between slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the Ingress port to the Egress port.

The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet inside the end-to-end transparent clock is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both Ingress and Egress ports only for the messages mentioned in Table 6.24-3. You can take the snapshot by setting the snapshot select bits (SNAPTYPSEL) to 10 in Register 448 (Timestamp Control Register).

PTP Messages

SYNC
Delay_Req

Table 6.24-3 End to End Transparent Clock: PTP Messages for Snapshot

Peer-to-Peer Transparent Clock

The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer. The residence time of the Pdelay_Req and the associated Pdelay_resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in Table 6.24-4.

PTP Messages
SYNC
Pdelay_Req
Pdelay_Resp

Table 6.24-4 Peer-to-Peer Transparent Clock: PTP Messages for Snapshot

You can take the snapshot by setting the snapshot select bits (SNAPTYPESEL) to 11 in Register 448 (Timestamp Control Register).

PTP Processing and Control

Table 6.24-5 shows the common message header for the PTP messages. This format is taken from IEEE standard 1588-2008 (Revision of IEEE Std. 1588-2002).

BITS **7 6 5 4 3 2 1 0**		**OCTETS**	**OFFSET**
transportSpecific	messageType	1	0
Reserved	versionPTP	1	1
messageLength		2	2
domainNumber		1	4
Reserved		1	5
flagField		2	6
correctionField		8	8
Reserved		4	16
sourcePortIdentity		10	20
sequenceId		2	30
controlField (*)		1	32

logMessageInterval	1	33

(*) – controlField is used in version 1. In version 2, messageType field is used for detecting different message types.

Table 6.24-5 Message Format Defined in IEEE 1588-2008

There are some fields in the Ethernet payload that you can use to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP frames:

- PTP Frames Over IPv4
- PTP Frames Over IPv6
- PTP Frames Over Ethernet

PTP Frames Over IPv4

Table 6.24-6 provides information about the fields that are matched to control snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of IEEE 1588-2008 standard and the message format defined in Table 6.24-5.

Field Matched	**Octet Position**	**Matched Value**	**Description**
MAC Frame Type	12, 13	0x0800	IPv4 datagram
IP version and Header Length	14	0x45	IP version is IPv4
Layer 4 Protocol	23	0x11	UDP
IP Multicast Address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed. 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP Multicast Address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP-Primary multicast address: 224.0.1.129 PTP-Pdelay multicast address: 224.0.0.107
UDP Destination Port	36, 37	0x013F, 0x0140	0x013F – PTP event message (*) 0x0140 – PTP general messages
PTP Control Field (IEEE version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP Message Type Field (IEEE version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce

			0xC: Signaling 0xD: Management
PTP Version	43 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

(*) PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only).

Table 6.24-6 IPv4-UDP PTP Frame Fields Required for Control and Status

PTP Frames Over IPv6

Table 6.24-7 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of IEEE 1588-2008 standard and the message format defined in Table 6.24-5.

Field Matched	**Octet Position**	**Matched Value**	**Description**
MAC Frame Type	12, 13	0x86DD	IP datagram
IP Version	14 (Bits [7:4])	0x6	IP version is IPv6
Layer 4 Protocol	20 (*)	0x11	UDP
PTP Multicast Address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP-Primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP-Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP Destination Port	56, 57 (*)	0x013F, 0x140	0x013F: PTP event message 0x0140: PTP general messages
PTP Control Field (IEEE 1588 version 1)	94 (*)	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management (version1)
PTP Message Type Field (IEEE version 2)	62 (*) (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

(*) The Extension Header is not defined for PTP packets.

Table 6.24-7 IPv6-UDP PTP Frame Fields Required for Control and Status

PTP Frames Over Ethernet

Table 6.24-8 provides information about the fields that are matched to control the snapshots for the PTP

packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged frames are offset by 4. This is based on Annex D of the IEEE 1588-2008 standard and the message format defined in Table 6.24-5.

Field Matched	Octet Position	Matched Value	Description
MAC Destination Multicast Address[a]	0-5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses[b]: ■ 01-1B-19-00-00-00 ■ 01-80-C2-00-00-0E[c]
MAC Frame Type	12, 13	0x88F7	PTP Ethernet frame
PTP Control Field (IEEE version 1)	46	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP Message Type Field (IEEE version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	15 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

a. The address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if Bit 18 (TSENMACADDR: Enable MAC address for PTP frame filtering) of the Timestamp Control register is set.

b. IEEE standard 1588-2008, Annex F

c. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

Table 6.24-8 Ethernet PTP Frame Fields Required for Control And Status

Reference Timing Source

The MAC supports the following reference timing source features defined in the IEEE 1588-2008 standard:

- 48-Bit Seconds Field
- Fixed Pulse-Per-Second Output
- Flexible Pulse-Per-Second Output

48-Bit Seconds Field

The MAC supports 80-bit timestamp. The timestamp has the following fields:

- UInteger48 secondsField

The seconds field is the integer portion of the timestamp in units of seconds and is 48-bits wide. For example, 2.000000001 seconds are represented as secondsField = 0x0000_0000_0002.

- UInteger32 nanosecondsField

The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 nanoseconds are represented as nanoSeconds = 0x0000_0001.

The nanoseconds field supports the following two modes:

- Digital rollover mode: In digital rollover mode, the maximum value in the nanoseconds field is 0x3B9A_C9FF, that is, (10e9-1) nanoseconds.
- Binary rollover mode: In binary rollover mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF_FFFF. Accuracy is ~0.466 ns per bit.

You can set these modes by using Bit 9 (TSCTRLSSR) of Register 448 (Timestamp Control Register).

When you select the advanced timestamp feature, the timestamp maintained in the MAC is still 64-bit wide. The overflow to the upper 16-bits of seconds register happens once in 130 years. You can read the values of the upper 16-bits of the seconds field from the CSR register.

Fixed Pulse-Per-Second Output

The EMAC supports the pulse-per-second (PPS) output that is given to indicate 1 second interval (default). You can change the frequency of the PPS output by setting Bits[3:0], PPSCTRL, in Register 459 (PPS Control Register).

Flexible Pulse-Per-Second Output

The EMAC also provides the flexibility to program the start or stop time, width, and interval of the pulse generated on the ptp_pps_o output.

The EMAC provides the following features with the Flexible PPS output option:

- Supports programming the start or stop time in terms of system time.
- Supports programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The existing Target Time registers (register 455 and 456) are used to program the start and stop time.
- Supports programming the stop time in advance, that is, you can program the stop time before the actual start time has elapsed.
- Supports programming the width, between the rising edge and corresponding falling edge of PPS signal output, in terms of number of units of sub-second increment value programmed in Register 449 (Sub-Second Increment Register). You can program the width of pulse from 1 to 232-1 units of sub-second increment value.
- Supports programming the interval, between the rising edges of PPS signal, in terms of number of units of sub-second increment value. You can program the interval between pulses from 1 to 232-1 units of sub-second increment value.
- Provides the option to cancel the programmed PPS start or stop request.
- Indicates error if the start or stop time being programmed has already elapsed.

The PTP Reference clock mentioned in the following sections is the clock at which the system time gets updated. When Bit 1 (Timestamp Fine or Coarse Update) of Register 448 (Timestamp Control Register) is set to 0, this clock is similar to the clk_ptp_ref_i clock. In the Fine Correction mode, this is the clock tick at which the system time gets updated (using incr_sub_sec_reg shown in Figure 6.24-7).

Flexible Pulse-Per-Second Output: PPS Start or Stop Time

You can initially program the start time in the Target Time registers (455 and 456).

If required, you can again program the start or stop time but you can do it only after the earlier programmed value is synchronized to the PTP clock domain. Bit 31 of EMAC registers 456 indicates that the synchronization is complete. This enables you to program the start or stop time in advance, even before the earlier stop or start time has elapsed.

The start or stop time should be programmed with advanced system time to ensure proper PPS signal output. If the application programs a start or stop time that has already elapsed, then the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

Flexible Pulse-Per-Second Output: PPS Width and Interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of sub-second increment value. For example, to have a PPS pulse width of 40ns and interval of 100ns, with the PTP reference clock of 50 MHz, you should program the width and interval to values 2 and 5 respectively. You can achieve smaller granularity by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, you should program or update the interval and width of the PPS signal output.

Transmit Path Functions

The structure of the descriptor changes when you enable the advanced timestamp feature. The advanced timestamp feature is supported only through Alternate (Enhanced) descriptors format. The descriptor is 32-bytes long (8 DWORDS) and the snapshot of the timestamp is written in descriptor TDES6 and TDES7. For detailed information about descriptors, see “Alternate or Enhanced Descriptors”.

Receive Path Functions

When you select the advanced timestamp feature, the MAC processes the received frames to identify valid PTP frames. You can control the snapshot of the time, to be sent to the application, by using the following options of Register 448 (Timestamp Control Register).

- Enable snapshot for all frames.
- Enable snapshot for IEEE 1588 version 2 or version 1 timestamp.
- Enable snapshot for PTP frames transmitted directly over Ethernet or UDP-IP-Ethernet.
- Enable timestamp snapshot for the received frame for IPv4 or IPv6.
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP).
- Enable the node to be a master or slave and select the snapshot type. This controls the type of messages for which snapshots are taken.

The DMA returns the timestamp to the software inside the corresponding Transmit and Receive Descriptor. The advanced timestamp feature is supported only with the 32-bytes long Alternate (Enhanced) descriptor. The extended status, containing the timestamp message status and the IPC status, is written in descriptor RDES4 and the snapshot of the timestamp is written in descriptors RDES6 and RDES7. For detailed information about descriptors, see “Alternate or Enhanced Descriptors”.

6.24.5.3 Source Address, VLAN, and CRC Insertion, Replacement, or Deletion

The EMAC supports the following functions for transmit frames:

- Source Address Insertion or Replacement
- VLAN Insertion, Replacement, or Deletion
- CRC Replacement

Source Address Insertion or Replacement

The software can use the SA insertion or replacement feature to instruct the MAC to do the following for transmit frames:

- Insert the content of the MAC Address Registers in the SA field.

- Replace the content of the SA field with the content of the MAC Address Registers.

The software can enable the SA insertion or replacement feature for all transmit frames or selective frames.

- To enable SA insertion or replacement feature for all frames, program Bits[30:28] of Register 0 (MAC Configuration Register).
- Program the SA Insertion Control field (TDES1 Bits [31:29]) in the first transmit descriptor of the frame. When Bit 31 of TDES1 is set, the SA Insertion Control field indicates insertion or replacement by MAC Address1 registers. When Bit 31 of TDES1 is reset, it indicates insertion or replacement by MAC Address 0 registers. If MAC Address1 Registers are not enabled, then MAC Address0 Registers are used for insertion or replacement irrespective of the value of the most significant bit of the SA Insertion Control field.

When SA insertion is enabled, the application should ensure that the frames that are sent to the MAC do not have the SA field. This is because the MAC does not check the presence of SA field in the transmit frame and just inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application should ensure that the frames that are sent to the MAC have the SA field. The MAC just replaces the six bytes, following the Destination Address field in the transmit frame, with the content of the MAC Address Registers.

VLAN Insertion, Replacement, or Deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for transmit frames:

- Insert or replace the VLAN Type field (C-VLAN or S-VLAN indicated by Bit 19 (CSVL) of Register 353 (VLAN Tag Inclusion or Replacement Register) and VLAN Tag field in the transmit frame with Bit [15:0], VLT, of Register 353 (VLAN Tag Inclusion or Replacement Register).
- Delete the VLAN Type and VLAN Tag fields in transmit frame.

The software can enable the VLAN insertion, replacement, or deletion feature for all transmit frames or selective frames. To enable this function for all transmit frames, program Bits[17:16] of Register 353 (VLAN Tag Inclusion or Replacement Register). To enable this function for selective frames, program the VLAN Insertion Control field (TDES0 Bits [19:18]) in the first transmit descriptor of the frame.

When VLAN replacement or deletion is enabled, the MAC checks the presence of the VLAN Type field (0x8100 or 0x88a8), after the Destination address (DA) and SA fields, in the transmit frame. The replace or delete operation does not occur if the VLAN Type field is not detected in the two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the transmit frame and just inserts the VLAN Type and VLAN Tag fields.

CRC Replacement

The software can use the CRC replacement feature to instruct the MAC to replace the FCS field in the transmit frame with the CRC computed by the MAC. This feature works on per-frame basis. To enable the CRC replacement feature, program the CRC Replacement Control field (Bit 24 of Transmit Descriptor Word 0 (TDES0)) in the first transmit descriptor of the frame.

This feature is valid only when Disable CRC control (Bit 27 in TDES0) is enabled, that is, the FCS field is provided by the software in the transmit frame. If SA or VLAN insertion control is enabled, the MAC appends or replaces the FCS field with the computed CRC when Disable CRC Control is enabled or disabled, respectively.

Table 6.24-9 explains how CRC replacement is done based on the values of Bit 27 (DC) and Bit 24 (CRCR) of Transmit Descriptor Word 0 (TDES0).

Bit 27 (DC)	Bit 24 (CRCR)	Description
0	x	Append CRC

		When DC = 0, the MAC appends the computed CRC irrespective of the CRCR setting.
1	1	Replace CRC
1	0	No operation (User has appended the CRC)

Table 6.24-9 CRC Replacement Based on TDES0 Bit 27 and Bit 24

6.24.5.4 Checksum Offload Engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. Because the most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams, the MAC has an optional Checksum Offload Engine (COE) to support checksum calculation and insertion in the transmit path, and error detection in the receive path.

Transmit Checksum Offload Engine

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each frame by setting the CIC bits (Bits 28:27 of TDES1, described in "Transmit Descriptor 1 (TDES1)").

The checksum for TCP, UDP, or ICMP is calculated over a complete frame, and then inserted into its corresponding header field. Because of this requirement, this function is enabled only when the Transmit FIFO is configured for the store-and-forward mode (that is, when the TSF bit is set in DMA Register 6). If the EMAC is configured for Threshold (cut-through) mode, the Transmit COE is bypassed.

You must make sure that the Transmit FIFO is deep enough to store a complete frame before that frame is transferred to the MAC transmitter. The reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid dead-lock. When reading starts, the COE fails and consequently all succeeding frames may get corrupted because of improper recovery. Therefore, you must enable the checksum insertion only in the frames that are less than the number of bytes in size (even in the store-and-forward mode).

See IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

IP Header Checksum Engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Ethernet frame's Type field has the value 0x0800 and the IP datagram's Version field has the value 0x4. The input frame's checksum field is ignored during calculation and replaced with the calculated value.

IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields. The result of this IP header checksum calculation is indicated by the IP Header Error status bit in the

Transmit status (Bit 16 in Table 6.24-13). This status bit is set whenever the values of the Ethernet Type field and the IP header's Version field are not consistent, or when the Ethernet frame does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the IP header's Version field is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total frame length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86dd but the IP header Version field is not equal to 0x6.

If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field

indicates an IPv4 payload.

TCP/UDP/ICMP Checksum Engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP.

The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. This engine can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input frame. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 16'h0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 16'h0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (Bit 12 in Table 6.24-13). This engine sets the Payload Checksum Error status bit when it detects that the frame has been forwarded to the MAC Transmitter engine in the store-and-forward mode without the end of frame (EOF) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 6.24-10 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as "No" in the table.

You should not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Transmit Checksum Offload Engine because it may result in incorrect checksum insertion or unexpected behavior.

Packet Type	Hardware IP Header Checksum Insertion	Hardware TCP/UDP Checksum Insertion
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in MAC) but less than or equal to the frame size constraint specified in "Transmit Checksum Offload Engine".	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in MAC) but less than or equal to the frame size constraint specified in "Transmit Checksum Offload Engine".	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: ■ Hop-by-hop options (in IPv6 main header) ■ Hop-by-hop options (in IPv6 extension header)	 ■ Not applicable ■ Not applicable	 ■ Yes ■ No

■ Destinations options ■ Routing (with segment left 0) ■ Routing (with segment left > 0) ■ TCP, UDP, or ICMP ■ Any other next header field in main or extension headers	■ Not applicable ■ Not applicable ■ Not applicable ■ Not applicable ■ Not applicable	■ Yes ■ No ■ No ■ Yes ■ No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: ■ IPv4 packet in an IPv4 tunnel ■ IPv6 packet in an IPv4 tunnel	■ Yes (IPv4 tunnel header) ■ Yes (IPv4 tunnel header)	■ No ■ No
IPv6 Tunnels: ■ IPv4 packet in an IPv6 tunnel ■ IPv6 packet in an IPv6 tunnel	■ Not applicable ■ Not applicable	■ No ■ No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Table 6.24-10 Transmit Checksum Offload Engine Functions for Different Packet Types

Receive Checksum Offload Engine

There are two types of Receive Checksum Offload engine available for configuration.

Type 1

The application can enable IP header checksum checking and TCP/UDP checksum offload by setting Bit 10 (IPC) of Register 0 (MAC Configuration Register). This module calculates the checksum by taking 16-bit one's complement of sum of the Ethernet frame's payload data's (DATA field) one's complement. The payload data is assumed to start from byte 15 (19 for a VLAN-tagged frame) of the received Ethernet frame. This module only processes IPv4 datagrams, bypassing and not processing all other types (such as IPv6). The receiver identifies an IPv4 frame by checking if the first nibble of the Ethernet payload (IP header) contains value 0x4.

This module also compares the calculated IP checksum with the received frame's IPv4 header checksum. Bytes 25 and 26 of the received Ethernet frame (29 and 30 for a VLAN-tagged frame) are taken as the IP header checksum. The header checksum is calculated against the header length field (20 bytes minimum). The result of the comparison (pass or fail) is given to the RFC, which sets the appropriate bit in the receive status word. If the Header Length field value is less than 5 or if the IP Version field does not equal 4, an error is indicated for the IP header checksum.

The one's complement sum of the IP datagram's 16-bit payload is also calculated. The start of the payload is considered to be the data after the IP header. If the data payload ends with a non-aligned half-word, then a pad byte is added for the sum calculation. The 16-bit one's complement of the resultant sum is forwarded to the RFC module, which inserts it into the data stream (towards the application) right after the FCS bytes (MS byte first) of the Ethernet payload. This 16-bit sum helps the software check the TCP/UDP header checksums faster. This 16-bit sum (which is always appended to the Ethernet frame in this mode) is invalid when the IP header checksum bit shows an Error status.

Type 2

In this mode, both IPv4 and IPv6 frames in the received Ethernet frames are detected and processed

for data integrity. Like Type 1, you can enable this module by setting Bit 10 (IPC) of Register 0 (MAC Configuration Register). The MAC receiver identifies IPv4 or IPv6 frames by checking for value 0x0800 or 0x86DD, respectively, in the received Ethernet frames' Type field. This identification also applies to single VLAN- tagged frames.

The Receive Checksum Offload engine calculates IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received frame does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload is less than the expected payload length given in the IP header.

Table 6.24-11 describes the functions supported by Type 2 Receive Checksum Offload engine based on the packet type.

In Type 2 mode, the MAC does not append any payload checksum bytes to the received Ethernet frames (as the Type 1 engine does).

Packet Type	**Hardware IP Header Checksum Checking**	**Hardware TCP/UDP/ICMP Checksum Checking**
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1522 bytes (2000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1522 bytes (2000 bytes when IEEE 802.3ad Support for 2K Packets is enabled in the MAC)	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: ■ Hop-by-hop options (in IPv6 main header) ■ Hop-by-hop options (in IPv6 extension header) ■ Destinations options ■ Routing (with segment left 0) ■ Routing (with segment left > 0) ■ Authentication ■ TCP, UDP, or ICMP ■ Any other next header field in main or extension headers	 ■ Not applicable ■ Not applicable ■ Not applicable ■ Not applicable ■ Not applicable ■ Not applicable ■ Not applicable ■ Not applicable	 ■ Yes ■ No ■ Yes ■ Yes ■ No ■ Yes ■ Yes ■ No
IPv4 packet has TCP header with Options fields	Yes	Yes

IPv4 Tunnels: ■ IPv4 packet in an IPv4 tunnel ■ IPv6 packet in an IPv4 tunnel	 ■ Yes (IPv4 tunnel header) ■ Yes (IPv4 tunnel header)	 ■ No ■ No
IPv6 Tunnels: ■ IPv4 packet in an IPv6 tunnel ■ IPv6 packet in an IPv6 tunnel	 ■ Not applicable ■ Not applicable	 ■ No ■ No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Table 6.24-11 Receive Checksum Offload Engine (Type 2) Functions for Different Packet Types

6.24.5.5 Power Management Block

The power management (PMT) block supports the reception of magic packet frames. The PMT block does not perform the clock gate function, but generates interrupts for magic packets that the MAC receives.

When you enable the power-down mode in the PMT block, the MAC drops all received frames and does not forward any frame to the MTL RxFIFO or the application. The MAC comes out of the power-down mode only when a magic packet frame is received and the corresponding detection is enabled.

The PMT block is available in the receive path of MAC. You can use Bit 1 of Register 11 (PMT Control and Status Register) to generate power management events. The application should program these bits. You can access the PMT registers in the similar manner as you access the MAC CSR registers. For mapping information, see Register 11 (PMT Control and Status Register).

Magic Packet Detection

The magic packet frame is based on a method that uses Advanced Micro Device's magic packet technology to power up the sleeping device on the network. The MAC receives a specific packet of information, called a magic packet, addressed to the node on the network.

The MAC checks only those magic packets that are addressed to the MAC or a broadcast address to determine whether these packets meet the wake-up requirements. The magic packets that pass the address filtering (unicast or broadcast) are checked to determine whether they meet the remote wake-up frame data format of 6 bytes of all ones followed by a MAC Address appearing 16 times.

The application enables the magic packet wake-up by writing 1 to Bit 1 (MGKPKTEN) of the PMT Control and Status Register. The PMT block constantly monitors each frame addressed to the node for a specific magic packet pattern. Each frame received is checked for a 48'hFF_FF_FF_FF_FF_FF pattern following the destination and source address field. The PMT block then checks the frame for 16 repetitions of the MAC address without any breaks or interruptions. In case of a break in the 16 repetitions of the address, the PMT block again scans the 48'hFF_FF_FF_FF_FF_FF pattern in the incoming frame. The 16 repetitions can be anywhere in the frame, but must be preceded by the synchronization stream (48'hFF_FF_FF_FF_FF_FF). The device can also accept a multicast frame, as long as the 16 duplications of the MAC address are detected.

If the MAC address of a node is 48'h00_11_22_33_44_55, then the MAC scans for the following data sequence:

Destination Address Source Address …….. FF FF FF FF FF FF

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55

…CRC

The magic packet detection is updated in the PMT Control and Status Register for the received magic packet. A PMT interrupt to the Application triggers a read to the PMT Control and Status Register to determine whether a magic packet frame has been received.

PMT Interrupts

The PMT interrupt signal is asserted when a valid magic packet frame is received. When software resets the Bit 0 (PWRDWN) in PMT Control and Status Register, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

6.24.5.6 Station Management Agent

The Station Management Agent (SMA) module allows the Application to access any PHY registers through a 2-wire Station Management interface (MIM).

The application can select one of the PHYs and one of the 32 registers within any PHY and send control data or receive status information. Only one register in one PHY can be addressed at any given time. For more details about the communication from the Application to the PHYs, refer to the Reconciliation Sublayer and Media Independent Interface Specifications section of the IEEE 802.3z specification, 1000BASE Ethernet. The application sends the control data to the PHY and receives status information from the PHY through the SMA module.

Functions

The MAC initiates the Management Write or Read operation. The MDC clock is a divided clock from the CSR clock. The divide factor depends on the clock range setting in the GMII Address register. Clock range is set as follows:

Selection	CSR Clock	MDC Clock
0000	60–100 MHz	CSR clock/42
0001	100–150 MHz	CSR clock/62
0010	20–35 MHz	CSR clock/16
0011	35–60 MHz	CSR clock/26
0100	150–250 MHz	CSR clock/102
0101	250–300 MHz	CSR clock/124
0110, 0111	Reserved	

The following figure displays the structure of a frame on the MDIO line:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE

Table 6.24-12 provides information about the frame fields.

Field	Description
IDLE	The mdio line is three-state; there is no clock on gmii_mdc_o.
PREAMBLE	32 continuous bits of value 1
START	Start of frame is 2'01
OPCODE	2'b10 for Read and 2'b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround is 2'bZ0 for Read and 2'b10 for Write
DATA	Any 16-bit value. In a Write operation, the MAC drives mdio. In a Read operation, PHY drives it.

Table 6.24-12 MDIO Frame Structure

GMII/MII Management Write Operation

When you set Bit 1 (GMII Write) and Bit 0 (GMII Busy) in Register 4 (GMII Address Register), the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (GMII Data Register) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the GMII Management Interface using the Management Frame Format specified in the GMII specifications (as per IEEE Standard 802.3-2002, Section 22.2.4.5). The application should not change the content of the GMII Address register or GMII Data register while the transaction is going on. The MAC ignores the Write operations performed to the GMII Address register or the GMII Data Register during this period (the Busy bit is high), and the transaction is completed without any error on the MCI interface.

When the Write operation is complete, the SMA indicates this to the CSR. The CSR then resets the Busy bit. The SMA module divides the CSR (Application) clock with the clock divider programmed (CR bits of GMII Address Register) to generate the MDC clock for this interface. The MAC drives the MDIO line for the complete duration of the frame. The frame format for the Write operation is as follows:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111...11	01	01	AAAAA	RRRRR	10	DDD . . . .DDD	Z

GMII/MII Management Read Operation

When you set Bit 1 and Bit 0 to 0 in Register 4 (GMII Address Register), the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the GMII Management Interface using the Management Frame Format specified in the GMII specifications (as per IEEE Standard 802.3-2002, Section 22.2.4.5). The application should not change the contents of the GMII Address register or GMII Data register while the transaction is going on. The MAC ignores the Write operations to the GMII Address register or GMII Data Register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface.

When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the GMII Data register with the data read from the PHY. The SMA module divides the CSR (Application) clock with the clock divider programmed (CR bits of GMII Address Register) to generate the MDC clock for this interface. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line.

The frame format for the Read operation is as follows:

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111...11	01	10	AAAAA	RRRRR	Z0	DDD . . . .DDD	Z

The MAC drives the MDO on negedge of MDC so that it can be sampled on posedge with sufficient setup or hold time (half cycle of MDC). Similarly, the PHY can drive MDI on negedge so that the MAC samples it on posedge, thus providing half cycle setup or hold time.

6.24.5.7 Reduced Media Independent Interface

The Reduced Media Independent Interface (RMII) specification reduces the pin count between Ethernet PHYs and Switch ASICs (only in 10/100 mode). According to the IEEE 802.3u standard, an MII contains 16 pins for data and control. In devices incorporating multiple MAC or PHY interfaces (such as switches), the number of pins adds significant cost with increase in port count. The RMII specification addresses this problem by reducing the pin count to 7 for each port — a 62.5% decrease in pin count.

The RMII module is instantiated between the MAC and the PHY. This helps in translating MAC's MII into the RMII. The RMII block has the following characteristics:

- Supports 10-Mbps and 100-Mbps operating rates. It does not support the 1000-Mbps operation.
- Provides independent, 2-bit wide transmit and receive paths by sourcing two clock references externally.

Block Diagram

Figure 6.24-9 shows the position of the RMII block relative to the EMAC and RMII PHY. The RMII block is placed in front of the EMAC to translate the MII signals to RMII signals.

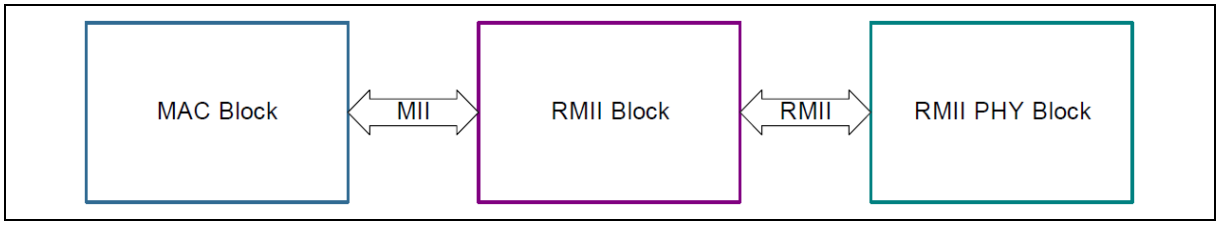


Figure 6.24-9 RMII Block Diagram

Transmit Bit Ordering

Each nibble from the MII must be transmitted on the RMII a di-bit at a time with the order of di-bit transmission shown in Figure 6.24-10. The lower order bits (D0 and D1) are transmitted first followed by higher order bits (D2 and D3).

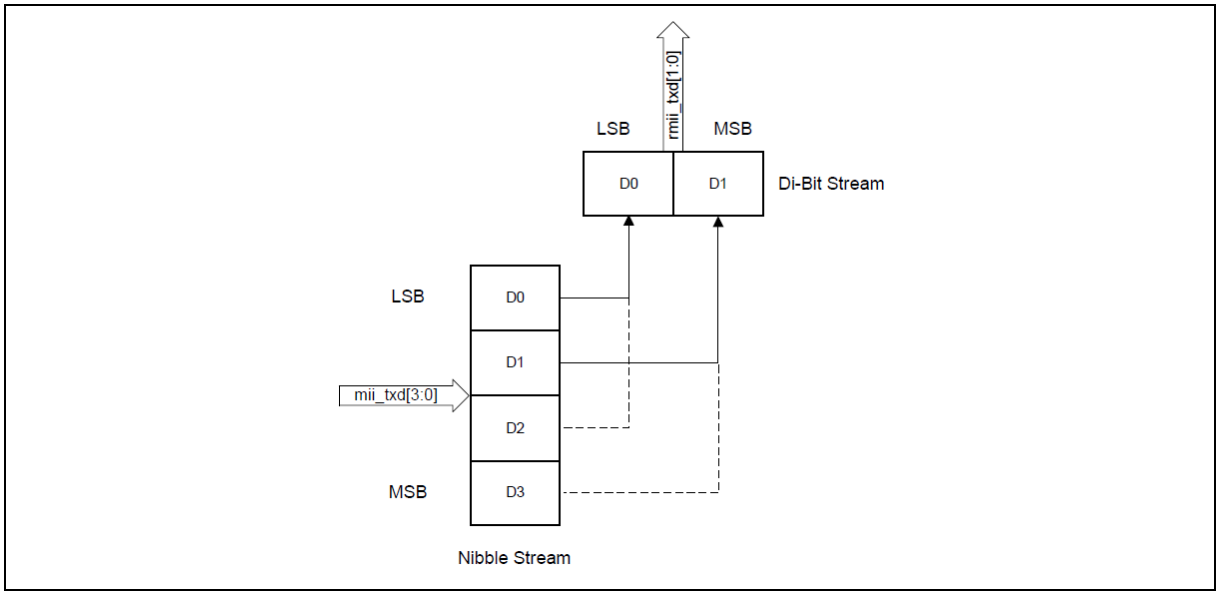


Figure 6.24-10 Transmission Bit Ordering

Receive Bit Ordering

Each nibble is transmitted to the MII from the di-bit received from the RMII in the nibble transmission order shown in Figure 6.24-11. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

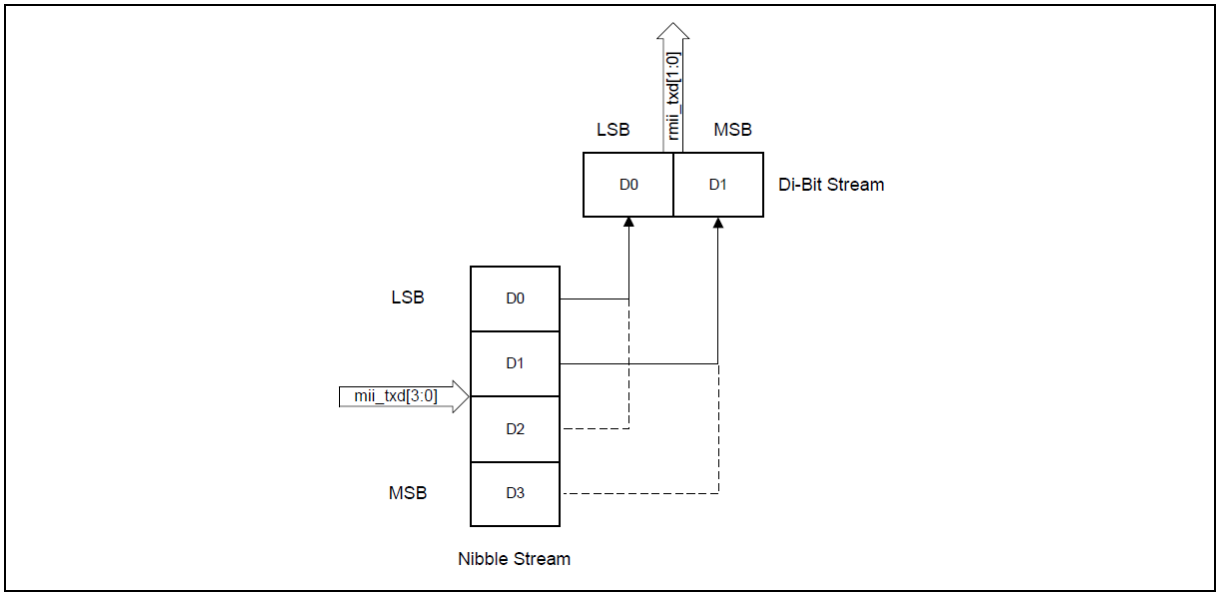


Figure 6.24-11 Receive Bit Ordering

6.24.5.8 Interrupts from the MAC

Interrupts can be generated from the MAC as a result of various events in the optional modules. These interrupt events are combined with the events in the DMA. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the host or software.

The Interrupt Status register in the Register 1005 (Status Register) describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented

from asserting the interrupt by setting the corresponding mask bits in the Register 15 (Interrupt Mask Register).

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

The TTI and GPI signals in Figure 6.24-12 refer to the bits that can be read in the Register 1005 (Status Register).

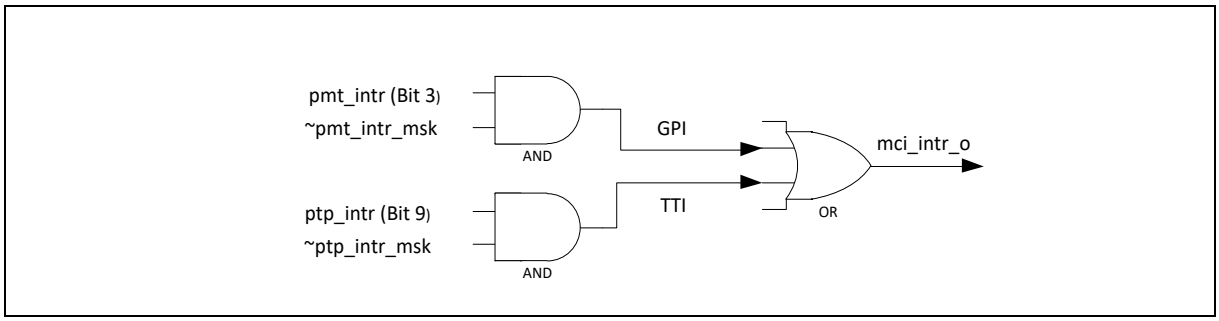


Figure 6.24-12 MAC Interrupt Masking Scheme

6.24.6 Descriptors

This chapter describes the EMAC descriptors. It contains the following sections:

- "Overview"
- "Descriptor Endianness"
- "Normal Descriptor"
- "Alternate or Enhanced Descriptors"

6.24.6.1 Overview

The DMA in the Ethernet subsystem transfers data based on a linked list of descriptors. The descriptors are created in the system memory. The EMAC supports the following two types of descriptors:

- Normal Descriptor

  This is the default descriptor format which can have 4 DWORDS. For information about this descriptor type, see 6.24.6.3 "Normal Descriptor".

- Enhanced Descriptor

  This is the alternate descriptor format which can have 8 DWORDS (32-bytes).

Each descriptor contains two buffers, two byte-count buffers, and two address pointers, which enable the adapter port to be compatible with various types of memory management schemes. Once configured, you cannot change the descriptor type.

There is no limit for number of descriptors that can be used for a single frame.

6.24.6.2 Descriptor Endianness

The descriptor addresses must be aligned to the used bus width (Word, DWord, or LWord for 32-bit, 64-bit, or 128-bit bus respectively).

Figure 6.24-13 shows the normal Receive and Transmit descriptors for 64-bit data bus when the endian mode of data bus and descriptors is little-endian.

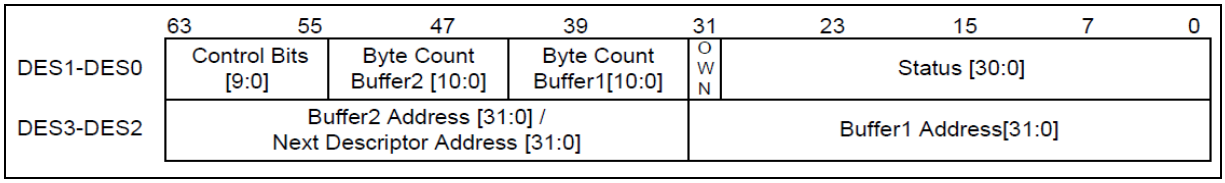


Figure 6.24-13 Rx/Tx Descriptors in Same-Endian Mode for 64-Bit, Little-Endian Format Data Bus

6.24.6.3 Normal Descriptor

This is the default descriptor format which can have 4 DWORDS. You can use this descriptor format when advanced features such as IEEE 1588-2008 Advanced Timestamping, and SA, VLAN, and CRC Insertion are not enabled.

Transmit Descriptor

The DMA in EMAC requires at least one descriptor for a transmit frame. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor has control fields which can be used to control the MAC operation on per-transmit frame basis.

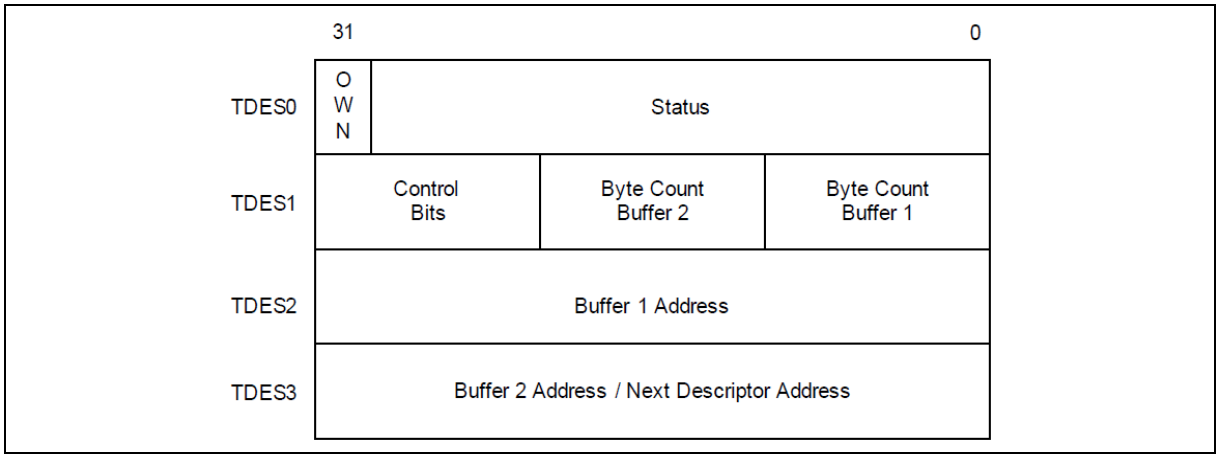


Figure 6.24-14 Transmit Descriptor Format in Little-Endian Mode With a 32-Bit Data Bus

Transmit Descriptor 0 (TDES0)

TDES0 contains the transmitted frame status and the descriptor ownership information. The descriptions in the Tables Table 6.24-13 through Table 6.24-16 are for the default mode of a little-endian 32-bit data bus with same-endian descriptors, or big-endian data bus with reverse-endian descriptors. If the DUT is configured for big-endian mode, the 32-bit data bus with same-endian descriptors or little-endian data bus with reverse-endian descriptors, then byte lanes 3 and 0 are swapped whereas byte lanes 1 and 2 are swapped, that is, Bits [31:24] are available on data bus [7:0], and vice-versa.

The status bits are valid when the LS bit is set. Therefore, when multiple descriptors are used for transmitting a single frame, the status bits of only last descriptor (that is, descriptor with LS bit set) are valid.

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are empty. The ownership bit of the First Descriptor of the frame should be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.
30:18	Reserved

17	TTSS: Tx Timestamp Status This status bit indicates that a timestamp has been captured for the corresponding transmit frame. When this bit is set, TDES2 and TDES3 have timestamp values that were captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES1[30]) in a descriptor is set. This bit is valid only when IEEE1588 timestamping feature is enabled; otherwise, it is reserved.
16	IHE: IP Header Error When set, this bit indicates that the Checksum Offload engine detected an IP header error. This bit is valid only when Tx Checksum Offload is enabled. Otherwise, it is reserved. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.
15	ES: Error Summary Indicates the logical OR of the following bits: ■ TDES0[16]: IP Header Error ■ TDES0[14]: Jabber Timeout ■ TDES0[13]: Frame Flush ■ TDES0[12]: Payload Checksum Error ■ TDES0[11]: Loss of Carrier ■ TDES0[10]: No Carrier ■ TDES0[9]: Late Collision ■ TDES0[8]: Excessive Collision ■ TDES0[2]: Excessive Deferral ■ TDES0[1]: Underflow Error
14	JT: Jabber Timeout When set, this bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is only set when the JD bit of Register 0 (MAC Configuration Register) is not set.
13	FF: Frame Flushed When set, this bit indicates that the DMA or MTL flushed the frame because of a software flush command given by the CPU.
12	PCE: Payload Checksum Error This bit, when set, indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either because of insufficient bytes, as indicated by the IP Header's Payload Length field, or the MTL starting to forward the frame to the MAC transmitter in the store-and-forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet frame being transmitted to avoid deadlock, the MTL starts forwarding the frame when the FIFO is full, even in the store- and-forward mode.
11	LOC: Loss of Carrier When set, this bit indicates that Loss of Carrier occurred during frame transmission (that is, the phy_crs_i signal was inactive for one or more transmit clock periods during frame transmission). This is valid only for the frames transmitted without collision and when the MAC operates in the half-duplex mode.
10	NC: No Carrier When set, this bit indicates that the carrier sense signal from the PHY was not asserted during transmission. **Note:** When any optional PHY interface is enabled, the carrier errors (NC or LOC) cannot be generated. This is because the CRS signal is internally-generated based on the internal MII TX_EN and RX_DV derived from the optional PHY interface signals.
9	LC: Late Collision When set, this bit indicates that frame transmission was aborted due to a collision occurring after the collision window (64 byte times including Preamble in MII Mode). Not valid if Underflow Error is set.
8	EC: Excessive Collision When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If Bit 9 (Disable Retry) in Register 0 (MAC Configuration Register) is set, this bit is set after the first collision and the transmission of the frame is aborted.

7	VF: VLAN Frame When set, this bit indicates that the transmitted frame was a VLAN-type frame.
6:3	CC: Collision Count This 4-bit counter value indicates the number of collisions occurring before the frame was transmitted. The count is not valid when the Excessive Collisions bit (TDES0[8]) is set.
2	ED: Excessive Deferral When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000-Mbps mode, or in Jumbo Frame enabled mode) if Bit 4 (Deferral Check) is set high in Register 0 (MAC Configuration Register).
1	UF: Underflow Error When set, this bit indicates that the MAC aborted the frame because data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty Transmit Buffer while transmitting the frame. The transmission process enters the suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]).
0	DB: Deferred Bit When set, this bit indicates that the MAC defers before transmission because of the presence of carrier. This bit is valid only in the half-duplex mode..

Table 6.24-13 Transmit Descriptor 0

Transmit Descriptor 1 (TDES1)

TDES1 contains the buffer sizes and other bits which control the descriptor chain or ring and the frame being transferred.

Bit	Description
31	IC: Interrupt on Completion When set, this bit sets Transmit Interrupt (Register 5[0]) after the present frame has been transmitted. **Note:** This bit is valid only when the last segment bit (TDES1[30]) is set.
30	LS: Last Segment When set, this bit indicates that the buffer contains the last segment of the frame. When this bit is set, the TBS1 or TBS2 field should have a non-zero value.
29	FS: First Segment When set, this bit indicates that the buffer contains the first segment of a frame.
28:27	CIC: Checksum Insertion Control These bits control the insertion of checksums in Ethernet frames that encapsulate TCP, UDP, or ICMP over IPv4 or IPv6: ■ 2'b00: Do nothing. Checksum Engine is bypassed ■ 2'b01: Insert IPv4 header checksum. Use this value to insert IPv4 header checksum when the frame encapsulates an IPv4 datagram. ■ 2'b10: Insert TCP/UDP/ICMP checksum. The checksum is calculated over the TCP, UDP, or ICMP segment only and the TCP, UDP, or ICMP pseudo-header checksum is assumed to be present in the corresponding input frame's Checksum field. An IPv4 header checksum is also inserted if the encapsulated datagram conforms to IPv4. ■ 2'b11: Insert a TCP/UDP/ICMP checksum that is fully calculated in this engine. In other words, the TCP, UDP, or ICMP pseudo-header is included in the checksum calculation, and the input frame's corresponding Checksum field has an all-zero value. An IPv4 Header checksum is also inserted if the encapsulated datagram conforms to IPv4. The Checksum engine detects whether the TCP, UDP, or ICMP segment is encapsulated in IPv4 or IPv6 and processes its data accordingly. These bits are valid only when the first segment bit (TDES1[29]) is set.
26	DC: Disable CRC When set, the MAC does not append the Cyclic Redundancy Check (CRC) to the end of the transmitted frame. This

	bit is valid only when the first segment (TDES1[29]) is set.
25	TER: Transmit End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The returns to the base address of the list, creating a descriptor ring.
24	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES1[24] is set, TBS2 (TDES1[21–11]) are “don’t care” values. TDES1[25] takes precedence over TDES1[24].
23	DP: Disable Padding When set, the MAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes and the CRC field is added despite the state of the DC (TDES1[26]) bit. This is valid only when the first segment (TDES1[29]) is set.
22	TTSE: Transmit Timestamp Enable When set, this bit enables IEEE1588 hardware timestamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES1[29]) is set.
21:11	TBS2: Transmit Buffer 2 Size These bits indicate the Second Data Buffer in bytes. This field is not valid if TDES1[24] is set.
10:0	TBS1: Transmit Buffer 1 Size These bits indicate the First Data Buffer byte size. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of TCH (Bit 24).

Table 6.24-14 Transmit Descriptor 1

Transmit Descriptor 2 (TDES2)

TDES2 contains the address pointer to the first buffer of the descriptor.

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. See “Host Data Buffer Alignment” for further detail on buffer address alignment.

Table 6.24-15 Transmit Descriptor 2

Transmit Descriptor 3 (TDES3)

TDES3 contains the address pointer either to the second buffer of the descriptor or the next descriptor.

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)

Table 6.24-16 Transmit Descriptor 3

Receive Descriptor

The DMA in EMAC requires at least two descriptors when receiving a frame. The Receive state machine of the DMA always attempts to acquire an extra descriptor in anticipation of an incoming frame. (The size of the incoming frame is unknown). Before the Rx DMA closes a descriptor, it attempts to acquire the next descriptor even if no frames are received.

In a single descriptor (receive) system, the subsystem generates a descriptor error if receive buffer is unable to accommodate the incoming frame and the next descriptor is not owned by the DMA. Thus, the Host is forced to increase either its descriptor pool or the buffer size. Otherwise, the subsystem starts dropping all incoming frames.

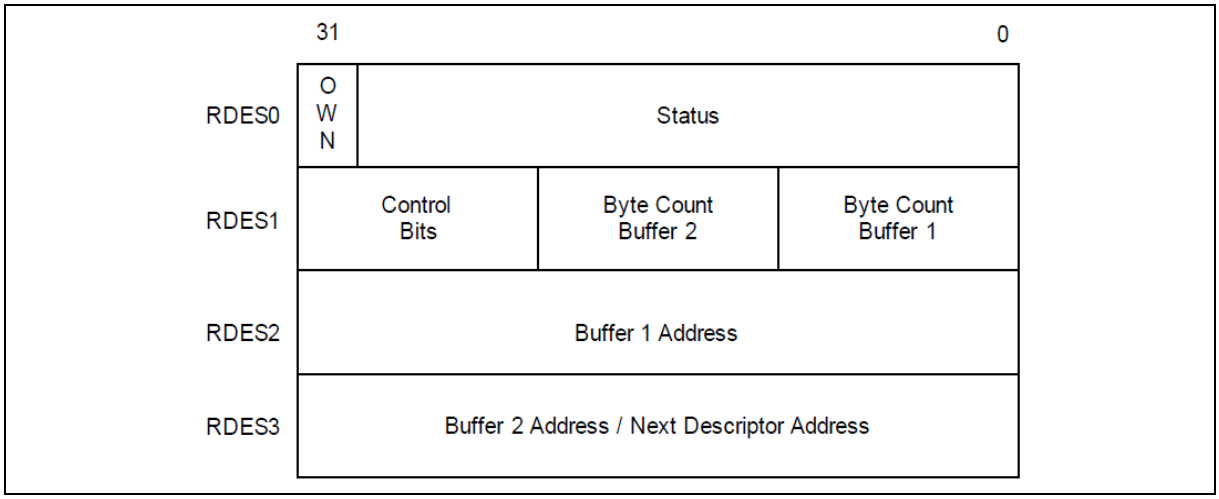


Figure 6.24-15 Receive Descriptor Format in Little-Endian Mode with a 32-Bit Data Bus

Receive Descriptor 0 (RDES0)

RDES0 contains the received frame status, the frame length, and the descriptor ownership information. The descriptions in Table 6.24-17 and Tables Table 6.24-19 through Table 6.24-21 are for the default mode of a little-endian 32-bit data bus with same-endian descriptors.

The status bits are valid when the LS bit is set. Therefore, when multiple descriptors are used for a single frame, status bits of only last descriptor (that is, descriptor with LS bit set) are valid.

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA of the EMAC. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.
30	AFM: Destination Address Filter Fail When set, this bit indicates a frame that failed in the DA Filter in the MAC.
29:16	FL: Frame Length These bits indicate the byte length of the received frame that was transferred to host memory. This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bit is reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame. When the Last Descriptor bit is not set, this field indicates the accumulated number of bytes that have been transferred for the current frame. The inclusion of CRC length in the frame length depends on the settings of Bit 7 and Bit 25 in Register 0 (MAC Configuration Register).
15	ES: Error Summary Indicates the logical OR of the following bits: ■ RDES0[0]: Payload Checksum Error ■ RDES0[1]: CRC Error ■ RDES0[3]: Receive Error ■ RDES0[4]: Watchdog Timeout ■ RDES0[6]: Late Collision

	■ RDES0[7]: IPC Checksum (Type 2) or Giant Frame ■ RDES0[11]: Overflow Error ■ RDES0[14]: Descriptor Error This field is valid only when the Last Descriptor (RDES0[8]) is set.
14	DE: Descriptor Error When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.
13	SAF: Source Address Filter Fail When set, this bit indicates that the SA field of frame failed the SA Filter in the MAC.
12	LE: Length Error When set, this bit indicates that the actual length of the frame received and the Length/ Type field do not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset. Length error status is not valid when CRC error is present.
11	OE: Overflow Error When set, this bit indicates that the received frame is damaged because of buffer overflow in Rx FIFO. **Note:** This bit is set only when the DMA transfers a partial frame to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial frames are dropped completely in Rx FIFO.
10	VLAN: VLAN Tag When set, this bit indicates that the frame pointed to by this descriptor is a VLAN frame tagged by the MAC. The VLAN tagging depends on checking VLAN fields of the received frame based on the Register 7 (VLAN Tag Register) settings.
9	FS: First Descriptor When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.
8	LS: Last Descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame
7	IPC Checksum Error or Giant Frame When IP Checksum Engine (Type 1) is enabled, this bit, when set, indicates one of the following: ■ The 16-bit IPv4 Header checksum calculated by the MAC did not match the received checksum bytes. ■ The header checksum checking is bypassed for non-IPv4 frames. If this bit is set because of any of these conditions, then Bit 15 (ES) is NOT set. If this bit is set when Full Checksum Offload Engine (Type 2) is enabled, it indicates an error in the IPv4 or IPv6 header. This error can be because of inconsistent Ethernet Type field and IP header Version field values, a header checksum mismatch in IPv4, or an Ethernet frame lacking the expected number of IP header bytes. See Table 6.24-18 for more details.
6	LC: Late Collision When set, this bit indicates that a late collision has occurred while receiving the frame in half-duplex mode.
5	FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 1,536). When this bit is reset, it indicates that the received frame is an IEEE802.3 frame. This bit is not valid for Runt frames which are less than 14 bytes. In addition, when Receive COE (Type 2) is selected and Bit 10 (IPC) of Register 0 (MAC Configuration Register) is set, this bit has value as indicated in Table 6.24-18.
4	RWT: Receive Watchdog Timeout When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.
3	RE: Receive Error When set, this bit indicates that the gmii_rxer_i signal is asserted while gmii_rxdv_i is asserted during frame reception. This error also includes carrier extension error in half-duplex mode. Error can be of less or no extension, or error (rxd ≠ 0f) during extension.

2	DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in MII Mode.
1	CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.
0	Rx MAC Address or Payload Checksum Error When set, this bit indicates that the Rx MAC Address registers value (1 to 15) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field. This bit is invalid when Bit 30 is set. If Full Checksum Offload Engine is enabled, this bit, when set, indicates the TCP, UDP, or ICMP checksum the core calculated does not match the received encapsulated TCP, UDP, or ICMP segment's Checksum field. This bit is also set when the received number of payload bytes does not match the value indicated in the Length field of the encapsulated IPv4 or IPv6 datagram in the received Ethernet frame. See Table 6.24-18 for more details.

Table 6.24-17 Receive Descriptor 0

When the Full Checksum Offload Engine (Type 2) is selected and it is enabled by setting Bit 10 (IPC) of Register 0 (MAC Configuration Register), the permutations of Bits 5, 7, and 0 reflect the conditions described in Table 6.24-18.

When the Full Checksum Offload Engine (Type 2) is selected and Bit 10 (IPC) of Register 0 (MAC Configuration Register) is reset, only the conditions shown in first two rows of Table 6.24-18 are valid. However, the frame status changes for the second condition as described in Table 6.24-18.

Bit 5: Frame Type	**Bit 7: IPC Checksum Error**	**Bit 0: Payload Checksum Error**	**Frame Status**
0	0	0	IEEE 802.3 Type frame (Length field value is less than 1,536) **Note:** This frame status is also valid when the Full Checksum Offload Engine (Type 2) is selected and Bit 10 (IPC) of Register 0 (MAC Configuration Register) is reset.
1	0	0	■ Frame status when the Full Checksum Offload Engine (Type 2) is selected and Bit 10 (IPC) is set: IPv4/IPv6 Type frame in which no checksum error is detected. ■ Frame status when the Full Checksum Offload Engine (Type 2) is selected and Bit 10 (IPC) is reset: The frame is an IEEE 802.3 Type frame (Length field value is greater than or equal to 1,536).
1	0	1	IPv4/IPv6 Type frame with a payload checksum error (as described for PCE) detected
1	1	0	IPv4/IPv6 Type frame with an IP header checksum error (as described for IPC CE) detected
1	1	1	IPv4/IPv6 Type frame with both IP header and payload checksum errors detected
0	0	1	IPv4/IPv6 Type frame with no IP header checksum error and the payload check bypassed, due to an unsupported payload
0	1	1	A Type frame that is neither IPv4 or IPv6 (the Checksum Offload engine bypasses checksum completely.)
0	1	0	Reserved

Table 6.24-18 Receive Descriptor 0 when COE (Type 2) is Enabled

Receive Descriptor 1 (RDES1)

RDES1 contains the buffer sizes and other bits that control the descriptor chain or ring.

Bit	Description
31	DIC: Disable Interrupt on Completion When set, this bit prevents the setting of the bit 6 (RI) of the Register 1005 (Status Register) for the received frame that ends in the buffer pointed to by this descriptor. This, in turn, disables the assertion of the interrupt to Host due to RI for that frame. **Note:** This bit is valid only when the last descriptor bit (RDES0[8]) is set.
30:26	Reserved
25	RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a Descriptor Ring.
24	RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When RDES1[24] is set, RBS2 (RDES1[21-11]) is a “don’t care” value. RDES1[25] takes precedence over RDES1[24].
23:22	Reserved
21:11	RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size in bytes. The buffer size must be a multiple of 4, 8, or 16 depending upon the bus widths (32, 64, or 128), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. In the case where the buffer size is not a multiple of 4, 8, or 16, the resulting behavior is undefined. This field is not valid if RDES1[24] is set.
10:0	RBS1: Receive Buffer 1 Size Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4, 8, or 16 depending upon the bus widths (32, 64, or 128), even if the value of RDES2 (buffer1 address pointer) is not aligned. In the case where the buffer size is not a multiple of 4, 8, or 16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 24).

Table 6.24-19 Receive Descriptor 1

Receive Descriptor 2 (RDES2)

RDES2 contains the address pointer to the first data buffer in the descriptor.

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. The DMA performs a write operation with the RDES2[3:0, 2:0, or 1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3:0, 2:0, or 1:0], corresponding to bus width of 128, 64, or 32, if the address pointer is to a buffer where the middle or last part of the frame is stored.

Table 6.24-20 Receive Descriptor 2

Receive Descriptor 3 (RDES3)

RDES3 contains the address pointer either to the second data buffer in the descriptor or to the next descriptor.

Bit	Description

31:0	Buffer 2 Address Pointer (Next Descriptor Address) These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[3, 2, or 1:0] = 0, corresponding to a bus width of 128, 64, or 32. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3[3:0, 2:0, or 1:0], corresponding to a bus width of 128, 64, or 32, if the address pointer is to a buffer where the middle or last part of the frame is stored.

Table 6.24-21 Receive Descriptor 3

Descriptor Format With IEEE 1588-2005 Timestamps Enabled

The default descriptor format (as described in “Transmit Descriptor” and “Receive Descriptor”), and field descriptions remain unchanged when created by software (Own bit is set in DES0). However, if the software has enabled IEEE 1588-2005 functionality, the DES2 and DES3 descriptor fields (see Figure 6.24-16) take on a different meaning when the DMA closes the descriptor (Own bit in DES0 is cleared).

The DMA updates the DES2 and DES3 with the timestamp value before clearing the Own bit in DES0.

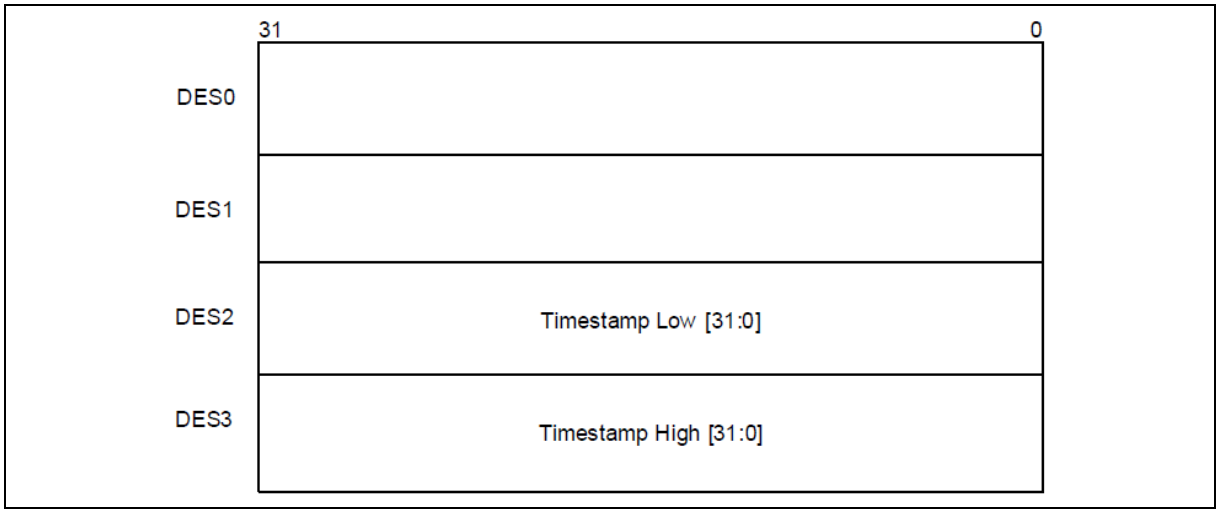


Figure 6.24-16 Descriptor Fields when DMA Clears the Own Bit

DES2 is updated with TSL and DES3 is updated with TSH. This ensures that the 64-bit timestamp can be processed as it is without any swapping between the two 32-bit words.

The following sections describe the details specific to receive and transmit descriptors in this mode.

Transmit Descriptor

In addition to the changes described in “Descriptor Format With IEEE 1588-2005 Timestamps Enabled”, the Transmit descriptor has additional control and status bits (TTSE and TTSS, respectively) for timestamping, as shown in Figure 6.24-17. Software sets the TTSE bit (when the Own bit is set), instructing the core to generate a timestamp for the corresponding Ethernet frame being transmitted. The DMA sets the TTSS bit if the timestamp has been updated in the TDES2 and TDES3 fields when the descriptor is closed (Own bit is cleared).

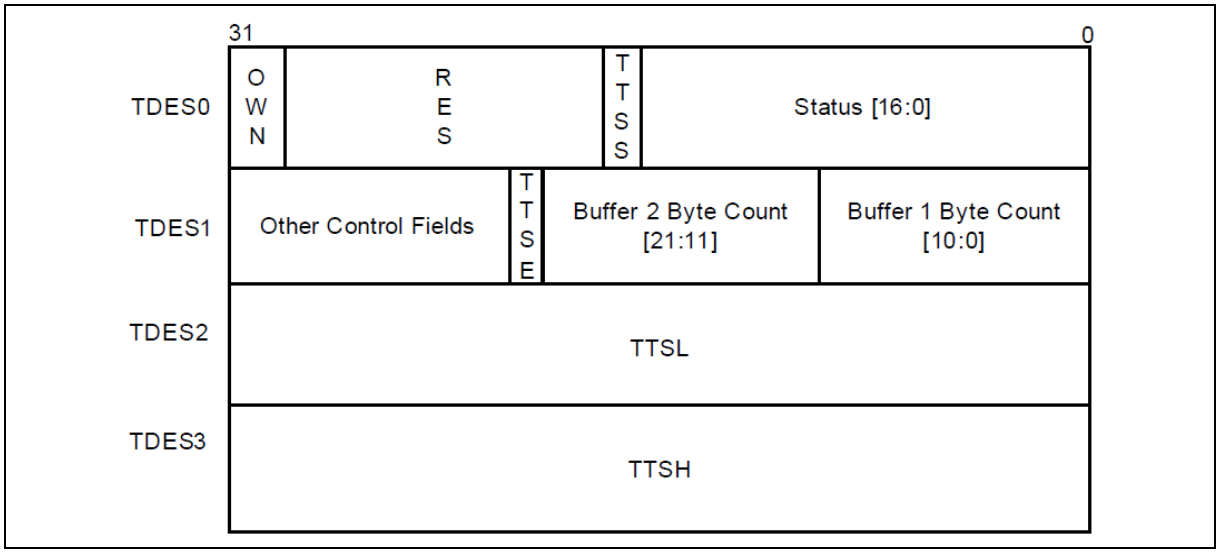


Figure 6.24-17 Descriptor Fields when DMA Clears the Own Bit

Transmit Descriptor: Transmit Timestamp Control and Status Fields

The position of these fields is different for normal transmit descriptor and enhanced format transmit descriptor. The value of this field in both the cases shall be preserved by the DMA at the time of closing the descriptor.

Updates to TDES0 and TDES1 for the default (normal) descriptor format are described in following tables.

Bit	Description
17	TTSS: Transmit Timestamp Status This field is a status bit indicating that a timestamp was captured for the corresponding transmit frame. When this bit is set, both TDES2 and TDES3 have a timestamp value that was captured for the transmit frame. This field is valid only when the Last Segment control bit (TDES1[30] in the descriptor) is set.

Table 6.24-22 Transmit Timestamp Status – Normal Descriptor Format Case (TDES0)

Bit	Description
22	TTSE: Transmit Timestamp Enable When set, this field enables IEEE1588 hardware timestamping for the transmit frame described by the descriptor. This field is valid only when the First Segment control bit (TDES1[29] in the descriptor) is set.

Table 6.24-23 Transmit Timestamp Control – Normal Descriptor Format Case (TDES1)

Transmit Descriptor: Transmit Timestamp Field

The transmit descriptor format and field descriptions remain unchanged when they are created by software (when the Own bit is set). However, when the DMA closes the last descriptor (marked, in the alternative descriptor format, by the LS bit in TDES1 or TDES0) and IEEE 1588 functionality is enabled (the Own bit is cleared), the TDES2 and TDES3 descriptor fields are updated with the timestamp, if taken, for that frame.

The fields in TDES2 and TDES3 are swapped when the core operates in 64-bit or 128-bit and with the descriptor in reverse-endian mode

Tables Table 6.24-24 and Table 6.24-25 describe the fields that have different meaning when the descriptor is closed.

Bit	Description
31:0	TTSL: Transmit Frame Timestamp Low This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment control bit (LS) in the descriptor is set.

Table 6.24-24 Transmit Descriptor Fields (TDES2)

Bit	Description
31:0	TTSH: Transmit Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment control bit (LS) in the descriptor is set.

Table 6.24-25 Transmit Descriptor Fields (TDES3)

Receive Descriptor

Receive Descriptor: Receive Timestamp

Figure 6.24-18 shows the format of receive descriptor when timestamping is enabled.

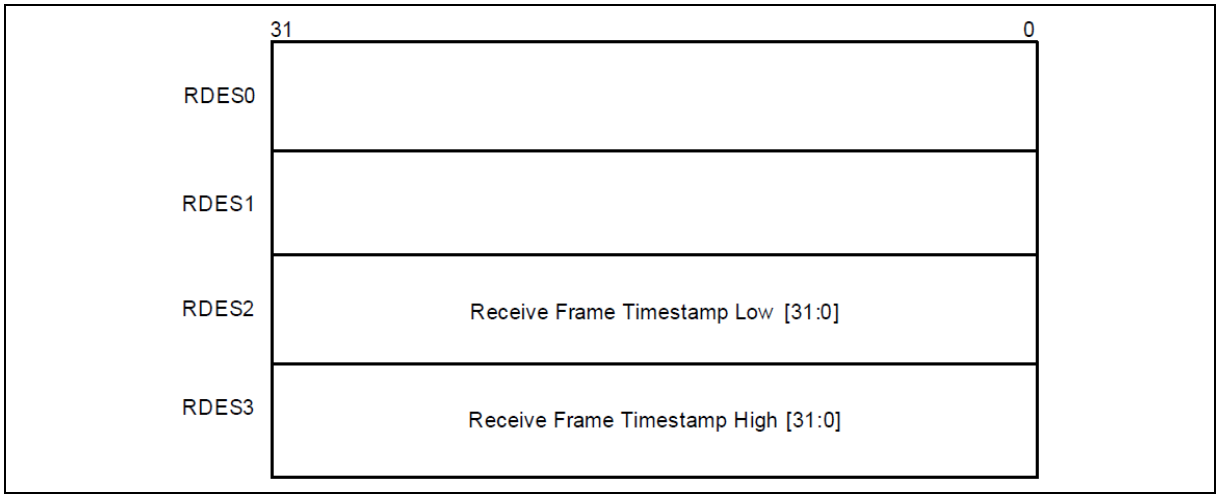


Figure 6.24-18 Receive Timestamp Fields

Table 6.24-26 and Table 6.24-27 describe the fields that have different meaning for RDES2 and RDES3 when the receive descriptor is closed and timestamping is enabled.

When software disables the Timestamp feature (the TSENA bit in Register 448 is low), the DMA does *not* update the RDES2 or RDES3 fields of descriptor before closing the RDES0.

Bit	Description
31:0	RTSL: Receive Frame Timestamp Low The DMA updates this field with the least significant 32 bits of the timestamp captured for the corresponding receive frame. The DMA updates this field only for the last descriptor of the receive frame indicated by Last Descriptor status bit (RDES0[8]). When this field and the RTSH field in RDES3 show an all-ones value, the timestamp must be treated as corrupt.

Table 6.24-26 Receive Descriptor Fields (RDES2)

Bit	Description

31:0	RTSH: Receive Frame Timestamp High The DMA updates this field with the most significant 32 bits of the timestamp captured for the corresponding receive frame. The DMA updates this field only for the last descriptor of the receive frame indicated by Last Descriptor status bit (RDES0[8]). When this field and RDES2's RTSL field show all-ones values, the timestamp must be treated as corrupt.

Table 6.24-27 Receive Descriptor Fields (RDES3)

6.24.6.4 Alternate or Enhanced Descriptors

The alternate (or enhanced) descriptor structure can have 8 DWORDS (32-bytes) instead of the 4 DWORDS as in the case of normal descriptor format. The alternate (or enhanced) descriptor structure is the only supported format when the IEEE 1588-2008 Advanced Timestamp feature, or SA, VLAN, and CRC Insertion feature is enabled. The features of the alternate descriptor structure are:

- The normal descriptor structure allows data buffers of up to 2,048 bytes. The alternative descriptor structure is implemented to support buffers of up to 8 Kbytes (useful for Jumbo frames).
- There is a re-assignment of control and status bits in TDES0, TDES1, RDES0 (Advanced timestamp or IPC full offload configuration), and RDES1.
- The transmit descriptor stores the timestamp in TDES6 and TDES7 when you select the Advanced Timestamp.
- This receive descriptor structure is also used for storing the extended status (RDES4) and timestamp (RDES6 and RDES7) when Advanced Timestamp or IPC Full Checksum Offload Engine feature is selected.
- You can select one of the following options for descriptor structure:
  - If timestamping is enabled in Register 448 (Timestamp Control Register) or Checksum Offload is enabled in Register 0 (MAC Configuration Register), the software needs to allocate 32-bytes (8 DWORDS) of memory for every descriptor. For this, the software should set Bit 7 (Alternate Descriptor Size) of Register 1000 (Bus Mode Register).
  - If timestamping or Checksum Offload is not enabled, the extended descriptors (DES4 to DES7) are not required. Therefore, the software can use alternate descriptors with the default size of 16-bytes (4 DWORDS). For this, the software should reset Bit 7 (Alternate Descriptor Size) of Register 1000 (Bus Mode Register) to 0.
- When alternate descriptor is selected without Timestamp or Receive IPC Full Checksum Offload Engine (Type 2) feature, the descriptor size is always 4 DWORDs (DES0-DES3). Therefore, the software can use alternate descriptors with the default size of 16 bytes.

Transmit Descriptor

The transmit descriptor structure is shown in Figure 6.24-19. The application software must program the control bits TDES0[31:18] during descriptor initialization. When the DMA updates the descriptor, it write backs all the control bits except the OWN bit (which it clears) and updates the status bits[7:0]. The contents of the transmitter descriptor word 0 (TDES0) through word 3 (TDES3) are given in Tables Table 6.24-28 through Table 6.24-31, respectively.

With the advance timestamp support, the snapshot of the timestamp to be taken can be enabled for a given frame by setting Bit 25 (TTSE) of TDES0. When the descriptor is closed (that is, when the OWN bit is cleared), the timestamp is written into TDES6 and TDES7. This is indicated by the status Bit 17 (TTSS) of TDES0 shown in Figure 6.24-19. The contents of TDES6 and TDES7 are mentioned in Table 6.24-32 and Table 6.24-33.

When Advanced Timestamp feature is enabled, the software should set Bit 7 of Register 1000 (Bus Mode Register), so that the DMA operates with extended descriptor size. When this control bit is reset, the TDES4-TDES7 descriptor space is not valid.

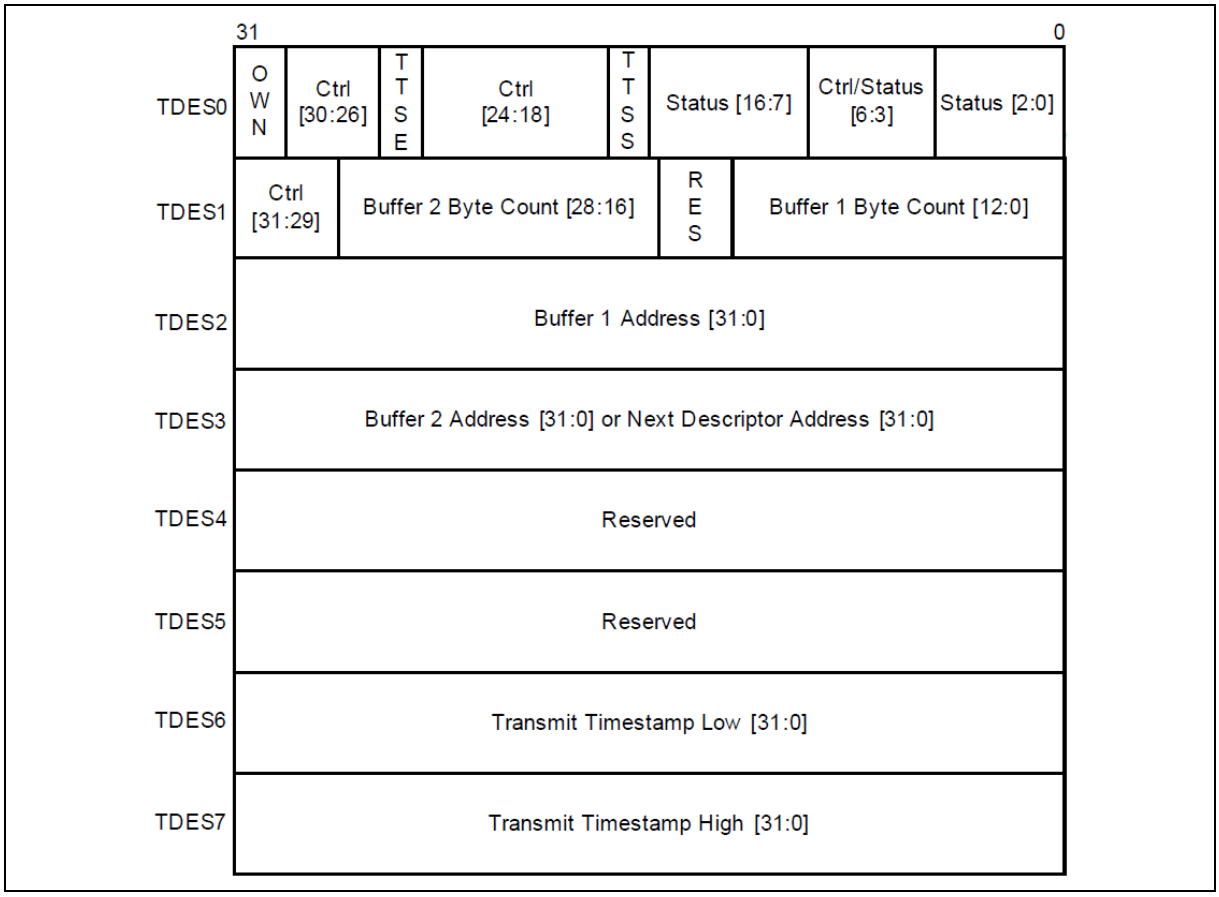


Figure 6.24-19 Transmit Descriptor Fields - Alternate (Enhanced) Format

The DMA always reads or fetches 4 DWORDS of the descriptor from system memory to obtain the buffer and control information as shown in Figure 6.24-20.

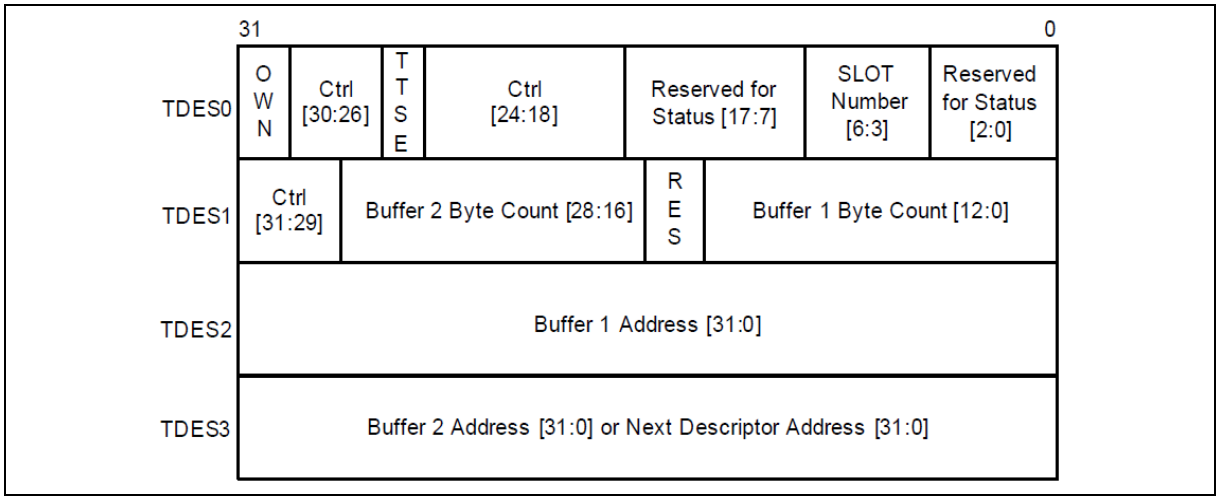


Figure 6.24-20 Transmit Descriptor Fetch (Read) for Alternate (Enhanced) Format

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA. When this bit is reset, it indicates that the descriptor

	is owned by the Host. The DMA clears this bit either when it completes the frame transmission or when the buffers allocated in the descriptor are read completely. The ownership bit of the frame's first descriptor must be set after all subsequent descriptors belonging to the same frame have been set. This avoids a possible race condition between fetching a descriptor and the driver setting an ownership bit.
30	IC: Interrupt on Completion When set, this bit sets the Transmit Interrupt (Register 5[0]) after the present frame has been transmitted. This bit is valid only when the last segment bit (TDES0[29]) is set.
29	LS: Last Segment When set, this bit indicates that the buffer contains the last segment of the frame. When this bit is set, the TBS1 or TBS2 field in TDES1 should have a non-zero value.
28	FS: First Segment When set, this bit indicates that the buffer contains the first segment of a frame.
27	DC: Disable CRC When this bit is set, the MAC does not append a cyclic redundancy check (CRC) to the end of the transmitted frame. This is valid only when the first segment (TDES0[28]) is set.
26	DP: Disable Pad When set, the MAC does not automatically add padding to a frame shorter than 64 bytes. When this bit is reset, the DMA automatically adds padding and CRC to a frame shorter than 64 bytes, and the CRC field is added despite the state of the DC (TDES0[27]) bit. This is valid only when the first segment (TDES0[28]) is set.
25	TTSE: Transmit Timestamp Enable When set, this bit enables IEEE1588 hardware timestamping for the transmit frame referenced by the descriptor. This field is valid only when the First Segment control bit (TDES0[28]) is set.
24	CRCR: CRC Replacement Control When set, the MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The host should ensure that the CRC bytes are present in the frame being transferred from the Transmit Buffer. This bit is valid when the First Segment control bit (TDES0[28]) is set. In addition, CRC replacement is done only when Bit 27 (DC) is set to 1.
23:22	CIC: Checksum Insertion Control These bits control the checksum calculation and insertion. The following list describes the bit encoding: ■ 2'b00: Checksum Insertion Disabled. ■ 2'b01: Only IP header checksum calculation and insertion are enabled. ■ 2'b10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo- header checksum is not calculated in hardware. ■ 2'b11: IP Header checksum and payload checksum calculation and insertion are enabled, and pseudo- header checksum is calculated in hardware. This field is valid when the First Segment control bit (TDES0[28]) is set.
21	TER: Transmit End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.
20	TCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When TDES0[20] is set, TBS2 (TDES1[28:16]) is a "don't care" value. TDES0[21] takes precedence over TDES0[20].
19:18	VLIC: VLAN Insertion Control When set, these bits request the MAC to perform VLAN tagging or untagging before transmitting the frames. If the frame is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. The following list describes the values of these bits: ■ 2'b00: Do not add a VLAN tag. ■ 2'b01: Remove the VLAN tag from the frames before transmission. This option should be used only with the VLAN frames.

	■ 2'b10: Insert a VLAN tag with the tag value programmed in Register 353 (VLAN Tag Inclusion or Replacement Register). ■ 2'b11: Replace the VLAN tag in frames with the Tag value programmed in Register 353 (VLAN Tag Inclusion or Replacement Register). This option should be used only with the VLAN frames. These bits are valid when the First Segment control bit (TDES0[28]) is set.
17	TTSS: Transmit Timestamp Status This field is used as a status bit to indicate that a timestamp was captured for the described transmit frame. When this bit is set, TDES2 and TDES3 have a timestamp value captured for the transmit frame. This field is only valid when the descriptor's Last Segment control bit (TDES0[29]) is set.
16	IHE: IP Header Error When set, this bit indicates that the MAC transmitter detected an error in the IP datagram header. The transmitter checks the header length in the IPv4 packet against the number of header bytes received from the application and indicates an error status if there is a mismatch. For IPv6 frames, a header error is reported if the main header length is not 40 bytes. Furthermore, the Ethernet Length/Type field value for an IPv4 or IPv6 frame must match the IP header version received with the packet. For IPv4 frames, an error status is also indicated if the Header Length field has a value less than 0x5. This bit is valid only when the Tx Checksum Offload is enabled. Otherwise, it is reserved. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload.
15	ES: Error Summary Indicates the logical OR of the following bits: ■ TDES0[14]: Jabber Timeout ■ TDES0[13]: Frame Flush ■ TDES0[11]: Loss of Carrier ■ TDES0[10]: No Carrier ■ TDES0[9]: Late Collision ■ TDES0[8]: Excessive Collision ■ TDES0[2]: Excessive Deferral ■ TDES0[1]: Underflow Error ■ TDES0[16]: IP Header Error ■ TDES0[12]: IP Payload Error
14	JT: Jabber Timeout When set, this bit indicates the MAC transmitter has experienced a jabber timeout. This bit is only set when Bit 22 (Jabber Disable) of Register 0 (MAC Configuration Register) is not set.
13	FF: Frame Flushed When set, this bit indicates that the DMA or MTL flushed the frame because of a software Flush command given by the CPU.
12	IPE: IP Payload Error When set, this bit indicates that MAC transmitter detected an error in the TCP, UDP, or ICMP IP datagram payload. The transmitter checks the payload length received in the IPv4 or IPv6 header against the actual number of TCP, UDP, or ICMP packet bytes received from the application and issues an error status in case of a mismatch.
11	LOC: Loss of Carrier When set, this bit indicates that a loss of carrier occurred during frame transmission (that is, the phy_crs_i signal was inactive for one or more transmit clock periods during frame transmission). This is valid only for the frames transmitted without collision when the MAC operates in the half-duplex mode.
10	NC: No Carrier When set, this bit indicates that the Carrier Sense signal from the PHY was not asserted during transmission. **Note:** When any optional PHY interface is enabled, the carrier errors (NC or LOC) cannot be generated. This is because the CRS signal is internally-generated based on the internal MII TX_EN and RX_DV derived from the optional PHY interface signals.
9	LC: Late Collision

	When set, this bit indicates that frame transmission is aborted because of a collision occurring after the collision window (64 byte-times, including preamble, in MII mode). This bit is not valid if the Underflow Error bit is set.
8	EC: Excessive Collision When set, this bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current frame. If Bit 9 (Disable Retry) bit in the Register 0 (MAC Configuration Register) is set, this bit is set after the first collision, and the transmission of the frame is aborted.
7	VF: VLAN Frame When set, this bit indicates that the transmitted frame is a VLAN-type frame.
6:3	CC: Collision Count (Status field) These status bits indicate the number of collisions that occurred before the frame was transmitted. This count is not valid when the Excessive Collisions bit (TDES0[8]) is set. The core updates this status field only in the half-duplex mode.
2	ED: Excessive Deferral When set, this bit indicates that the transmission has ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1,000-Mbps mode or if Jumbo Frame is enabled) if Bit 4 (Deferral Check) bit in Register 0 (MAC Configuration Register) is set high.
1	UF: Underflow Error When set, this bit indicates that the MAC aborted the frame because the data arrived late from the Host memory. Underflow Error indicates that the DMA encountered an empty transmit buffer while transmitting the frame. The transmission process enters the Suspended state and sets both Transmit Underflow (Register 5[5]) and Transmit Interrupt (Register 5[0]).
0	DB: Deferred Bit When set, this bit indicates that the MAC defers before transmission because of the presence of carrier. This bit is valid only in the half-duplex mode.

Table 6.24-28 Transmit Descriptor Word 0 (TDES0)

Bit	Description
31:29	SAIC: SA Insertion Control These bits request the MAC to add or replace the Source Address field in the Ethernet frame with the value given in the MAC Address 0 register. If the Source Address field is modified in a frame, the MAC automatically recalculates and replaces the CRC bytes. The Bit 31 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[30:29]: ■ 2'b00: Do not include the source address. ■ 2'b01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. ■ 2'b10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. ■ 2'b11: Reserved These bits are valid when the First Segment control bit (TDES0[28]) is set.
28:16	TBS2: Transmit Buffer 2 Size This field indicates the second data buffer size in bytes. This field is not valid if TDES0[20] is set.
15:13	Reserved
12:0	TBS1: Transmit Buffer 1 Size These bits indicate the first data buffer byte size, in bytes. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or the next descriptor, depending on the value of TCH (TDES0[20]).

Table 6.24-29 Transmit Descriptor Word 1 (TDES1)

Bit	Description

31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There is no limitation on the buffer address alignment. For more information about buffer address alignment, see “Host Data Buffer Alignment”.

Table 6.24-30 Transmit Descriptor 2 (TDES2)

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) Indicates the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (TDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. The buffer address pointer must be aligned to the bus width only when TDES1[24] is set. (LSBs are ignored internally.)

Table 6.24-31 Transmit Descriptor 3 (TDES3)

Bit	Description
31:0	TTSL: Transmit Frame Timestamp Low This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding transmit frame. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

Table 6.24-32 Transmit Descriptor 6 (TDES6)

Bit	Description
31:0	TTSH: Transmit Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

Table 6.24-33 Transmit Descriptor 7 (TDES7)

Receive Descriptor

The structure of the received descriptor is shown in Figure 6.24-21. This can have 32 bytes of descriptor data (8 DWORDs) when Advanced Timestamp or IPC Full Offload feature is selected.

When either of these features is enabled, the Software should set Bit 7 of Register 1000 (Bus Mode Register) so that the DMA operates with extended descriptor size. When this control bit is reset, the RDES0[0] is always cleared and the RDES4-RDES7 descriptor space is not valid.

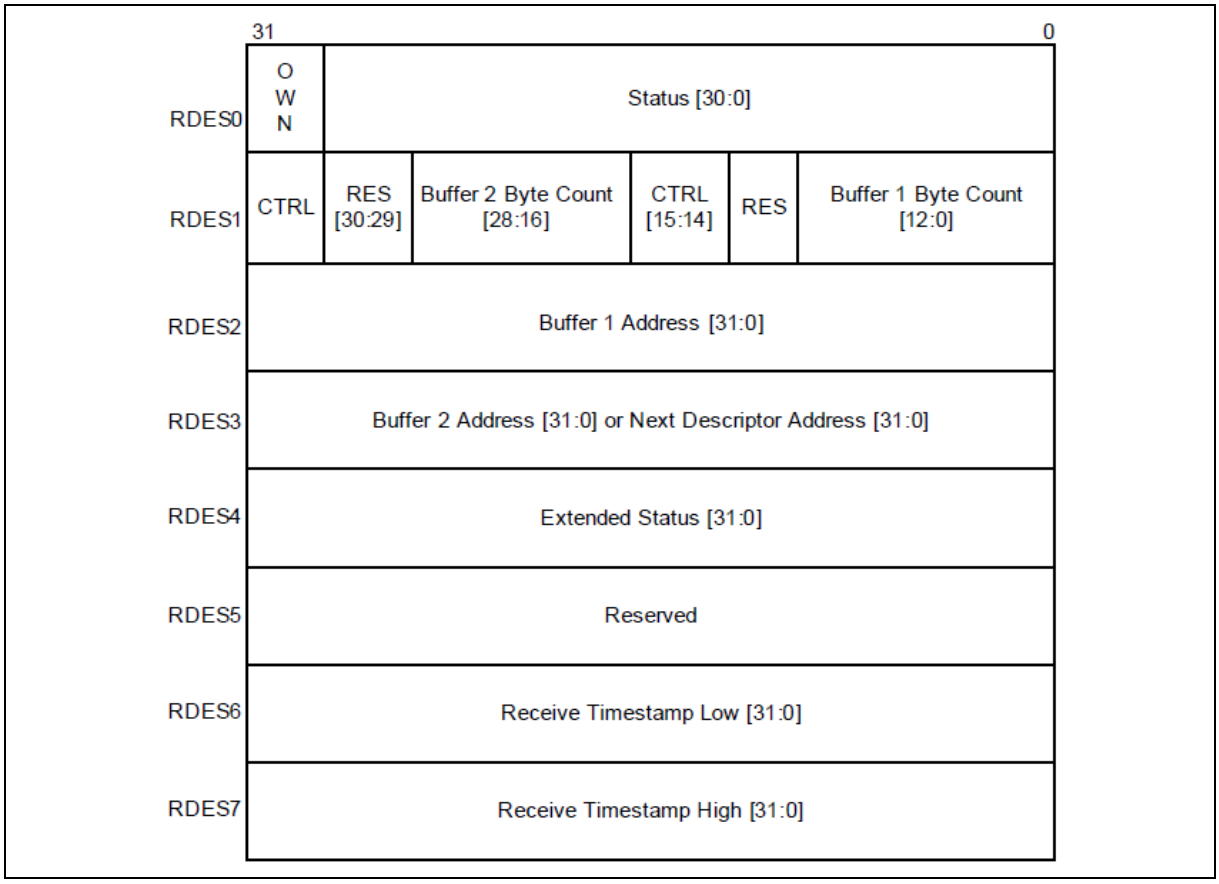


Figure 6.24-21 Receive Descriptor Fields - Alternate (Enhanced) Format

The contents of RDES0 are identified in Table 6.24-34. The contents of RDES1 through RDES3 are identified in Tables Table 6.24-35 through Table 6.24-37, respectively.

The status bits are valid when the LS bit is set. Therefore, when multiple descriptors are used for a single frame, status bits of only last descriptor (that is, descriptor with LS bit set) are valid.

Bit	Description
31	OWN: Own Bit When set, this bit indicates that the descriptor is owned by the DMA of the EMAC. When this bit is reset, this bit indicates that the descriptor is owned by the Host. The DMA clears this bit either when it completes the frame reception or when the buffers that are associated with this descriptor are full.
30	AFM: Destination Address Filter Fail When set, this bit indicates a frame that failed in the DA Filter in the MAC.
29:16	FL: Frame Length These bits indicate the byte length of the received frame that was transferred to host memory. This field is valid when Last Descriptor (RDES0[8]) is set and either the Descriptor Error (RDES0[14]) or Overflow Error bits are reset. The frame length also includes the two bytes appended to the Ethernet frame when IP checksum calculation (Type 1) is enabled and the received frame is not a MAC control frame. This field is valid when Last Descriptor (RDES0[8]) is set. When the Last Descriptor and Error Summary bits are not set, this field indicates the accumulated number of bytes that have been transferred for the current frame. The inclusion of CRC length in the frame length depends on the settings of Bit 7 and Bit 25 in Register 0 (MAC Configuration Register).
15	ES: Error Summary Indicates the logical OR of the following bits:

	■ RDES0[1]: CRC Error ■ RDES0[3]: Receive Error ■ RDES0[4]: Watchdog Timeout ■ RDES0[6]: Late Collision ■ RDES0[7]: Giant Frame ■ RDES4[4:3]: IP Header or Payload Error ■ RDES0[11]: Overflow Error ■ RDES0[14]: Descriptor Error This field is valid only when the Last Descriptor (RDES0[8]) is set.
14	DE: Descriptor Error When set, this bit indicates a frame truncation caused by a frame that does not fit within the current descriptor buffers, and that the DMA does not own the Next Descriptor. The frame is truncated. This field is valid only when the Last Descriptor (RDES0[8]) is set.
13	SAF: Source Address Filter Fail When set, this bit indicates that the SA field of frame failed the SA Filter in the MAC.
12	LE: Length Error When set, this bit indicates that the actual length of the frame received and that the Length/ Type field does not match. This bit is valid only when the Frame Type (RDES0[5]) bit is reset.
11	OE: Overflow Error When set, this bit indicates that the received frame was damaged because of buffer overflow in MTL. **Note:** This bit is set only when the DMA transfers a partial frame to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial frames are dropped completely in Rx FIFO.
10	VLAN: VLAN Tag When set, this bit indicates that the frame to which this descriptor is pointing is a VLAN frame tagged by the MAC. The VLAN tagging depends on checking the VLAN fields of received frame based on the Register 7 (VLAN Tag Register) setting.
9	FS: First Descriptor When set, this bit indicates that this descriptor contains the first buffer of the frame. If the size of the first buffer is 0, the second buffer contains the beginning of the frame. If the size of the second buffer is also 0, the next Descriptor contains the beginning of the frame.
8	LS: Last Descriptor When set, this bit indicates that the buffers pointed to by this descriptor are the last buffers of the frame
7	Timestamp Available, IP Checksum Error (Type1), or Giant Frame When Advanced Timestamp feature is present, when set, this bit indicates that a snapshot of the Timestamp is written in descriptor words 6 (RDES6) and 7 (RDES7). This is valid only when the Last Descriptor bit (RDES0[8]) is set. When IP Checksum Engine (Type 1) is selected, this bit, when set, indicates one of the following: ■ The 16-bit IPv4 header checksum calculated by the core did not match the received checksum bytes. ■ The header checksum checking is bypassed for non-IPv4 frames. Otherwise, this bit, when set, indicates the Giant Frame Status. Giant frames are larger than 1,518-byte (or 1,522-byte for VLAN or 2,000-byte when Bit 27 of MAC Configuration register is set) normal frames and larger than 9,018-byte (9,022-byte for VLAN) frame when Jumbo Frame processing is enabled.
6	LC: Late Collision When set, this bit indicates that a late collision has occurred while receiving the frame in the half-duplex mode.
5	FT: Frame Type When set, this bit indicates that the Receive Frame is an Ethernet-type frame (the LT field is greater than or equal to 1,536). When this bit is reset, it indicates that the received frame is an IEEE 802.3 frame. This bit is not valid for Runt frames less than 14 bytes.
4	RWT: Receive Watchdog Timeout

	When set, this bit indicates that the Receive Watchdog Timer has expired while receiving the current frame and the current frame is truncated after the Watchdog Timeout.
3	RE: Receive Error When set, this bit indicates that the gmii_rxer_i signal is asserted while gmii_rxdv_i is asserted during frame reception. This error also includes carrier extension error in half-duplex mode. Error can be of less or no extension, or error (rxd ≠ 0f) during extension.
2	DE: Dribble Bit Error When set, this bit indicates that the received frame has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
1	CE: CRC Error When set, this bit indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received frame. This field is valid only when the Last Descriptor (RDES0[8]) is set.
0	Extended Status Available/Rx MAC Address When either Advanced Timestamp or IP Checksum Offload (Type 2) is present, this bit, when set, indicates that the extended status is available in descriptor word 4 (RDES4). This is valid only when the Last Descriptor bit (RDES0[8]) is set. This bit is invalid when Bit 30 is set. When IP Checksum Offload (Type 2) is present, this bit is set even when IP Checksum Offload engine bypasses the processing of received frame. The bypassing may be because of non-IP frame or IP frame with non-TCP/UDP/ICMP payload. When Advance Timestamp Feature or IPC Full Offload is not selected, this bit indicates Rx MAC Address status. When set, this bit indicates that the Rx MAC Address registers value (1 to 15) matched the frame's DA field. When reset, this bit indicates that the Rx MAC Address Register 0 value matched the DA field.

Table 6.24-34 Receive Descriptor Fields (RDES0)

Bit	Description
31	DIC: Disable Interrupt on Completion When set, this bit prevents setting the Status Register's RI bit (CSR5[6]) for the received frame ending in the buffer indicated by this descriptor. This, in turn, disables the assertion of the interrupt to Host because of RI for that frame. **Note:** This bit is valid only when the last descriptor bit (RDES0[8]) is set.
30:29	Reserved
28:16	RBS2: Receive Buffer 2 Size These bits indicate the second data buffer size, in bytes. The buffer size must be a multiple of 4, 8, or 16, depending on the bus widths (32, 64, or 128, respectively), even if the value of RDES3 (buffer2 address pointer) is not aligned to bus width. If the buffer size is not an appropriate multiple of 4, 8, or 16, the resulting behavior is undefined. This field is not valid if RDES1[14] is set.
15	RER: Receive End of Ring When set, this bit indicates that the descriptor list reached its final descriptor. The DMA returns to the base address of the list, creating a descriptor ring.
14	RCH: Second Address Chained When set, this bit indicates that the second address in the descriptor is the Next Descriptor address rather than the second buffer address. When this bit is set, RBS2 (RDES1[28:16]) is a "don't care" value. RDES1[15] takes precedence over RDES1[14].
13	Reserved
12:0	RBS1: Receive Buffer 1 Size Indicates the first data buffer size in bytes. The buffer size must be a multiple of 4, 8, or 16, depending upon the bus widths (32, 64, or 128), even if the value of RDES2 (buffer1 address pointer) is not aligned. When the buffer size is not a multiple of 4, 8, or 16, the resulting behavior is undefined. If this field is 0, the DMA ignores this buffer and uses Buffer 2 or next descriptor depending on the value of RCH (Bit 14).

Table 6.24-35 Receive Descriptor Fields 1 (RDES1)

Bit	Description
31:0	Buffer 1 Address Pointer These bits indicate the physical address of Buffer 1. There are no limitations on the buffer address alignment except for the following condition: The DMA uses the configured value for its address generation when the RDES2 value is used to store the start of frame. The DMA performs a write operation with the RDES2[3:0, 2:0, or 1:0] bits as 0 during the transfer of the start of frame but the frame data is shifted as per the actual Buffer address pointer. The DMA ignores RDES2[3:0, 2:0, or 1:0] (corresponding to bus width of 128, 64, or 32) if the address pointer is to a buffer where the middle or last part of the frame is stored.

Table 6.24-36 Receive Descriptor Fields 2 (RDES2)

Bit	Description
31:0	Buffer 2 Address Pointer (Next Descriptor Address) These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. If the Second Address Chained (RDES1[24]) bit is set, this address contains the pointer to the physical memory where the Next Descriptor is present. If RDES1[24] is set, the buffer (Next Descriptor) address pointer must be bus width-aligned (RDES3[3, 2, or 1:0] = 0, corresponding to a bus width of 128, 64, or 32. LSBs are ignored internally.) However, when RDES1[24] is reset, there are no limitations on the RDES3 value, except for the following condition: The DMA uses the configured value for its buffer address generation when the RDES3 value is used to store the start of frame. The DMA ignores RDES3 [3, 2, or 1:0] (corresponding to a bus width of 128, 64, or 32) if the address pointer is to a buffer where the middle or last part of the frame is stored.

Table 6.24-37 Receive Descriptor Fields 3 (RDES3)

The extended status written is as shown in Table 6.24-38. The extended status is written only when there is status related to IPC or timestamp available. The availability of extended status is indicated by Bit 0 of RDES0. This status is available only when the Advance Timestamp or IPC Full Offload feature is selected.

Bit	Description
31:28	Reserved
27:26	Layer 3 and Layer 4 Filter Number Matched These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received frame. ■ 00: Filter 0 ■ 01: Filter 1 ■ 10: Filter 2 ■ 11: Filter 3 This field is valid only when Bit 24 or Bit 25 is set high. When more than one filter matches, these bits give only the lowest filter number.
25	Layer 4 Filter Match When set, this bit indicates that the received frame matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: ■ Layer 3 fields are not enabled and all enabled Layer 4 fields match. ■ All enabled Layer 3 and Layer 4 filter fields match. When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by Bits [27:26].
24	Layer 3 Filter Match When set, this bit indicates that the received frame matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: ■ All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed. ■ All enabled filter fields match.

	When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by Bits [27:26].
23:21	Reserved
15	Reserved
14	Timestamp Dropped When set, this bit indicates that the timestamp was captured for this frame but got dropped in the MTL Rx FIFO because of overflow. This bit is available only when you select the Advanced Timestamp feature. Otherwise, this bit is reserved.
13	PTP Version When set, this bit indicates that the received PTP message is having the IEEE 1588 version 2 format. When reset, it has the version 1 format. This bit is available only when you select the Advanced Timestamp feature. Otherwise, this bit is reserved.
12	PTP Frame Type When set, this bit indicates that the PTP message is sent directly over Ethernet. When this bit is not set and the message type is non-zero, it indicates that the PTP message is sent over UDP-IPv4 or UDP-IPv6. The information about IPv4 or IPv6 can be obtained from Bits 6 and 7. This bit is available only when you select the Advanced Timestamp feature.
11:8	Message Type These bits are encoded to give the type of the message received. ■ 0000: No PTP message received ■ 0001: SYNC (all clock types) ■ 0010: Follow_Up (all clock types) ■ 0011: Delay_Req (all clock types) ■ 0100: Delay_Resp (all clock types) ■ 0101: Pdelay_Req (in peer-to-peer transparent clock) ■ 0110: Pdelay_Resp (in peer-to-peer transparent clock) ■ 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) ■ 1000: Announce ■ 1001: Management ■ 1010: Signaling ■ 1011-1110: Reserved ■ 1111: PTP packet with Reserved message type These bits are available only when you select the Advance Timestamp feature.
7	IPv6 Packet Received When set, this bit indicates that the received packet is an IPv6 packet. This bit is updated only when Bit 10 (IPC) of Register 0 (MAC Configuration Register) is set. This bit is available when you select the Enable Receive Full TCP/IP Checksum (Type 2) feature.
6	IPv4 Packet Received When set, this bit indicates that the received packet is an IPv4 packet. This bit is updated only when Bit 10 (IPC) of Register 0 (MAC Configuration Register) is set. This bit is available when you select the Enable Receive Full TCP/IP Checksum (Type 2) feature.
5	IP Checksum Bypassed When set, this bit indicates that the checksum offload engine is bypassed. This bit is available when you select the Enable Receive Full TCP/IP Checksum (Type 2) feature.
4	IP Payload Error When set, this bit indicates that the 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) that the core calculated does not match the corresponding checksum field in the received segment. It is also set when the TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. This bit is valid when

	either Bit 7 or Bit 6 is set. This bit is available when you select the Enable Receive Full TCP/IP Checksum (Type 2) feature.
3	IP Header Error When set, this bit indicates that either the 16-bit IPv4 header checksum calculated by the core does not match the received checksum bytes, or the IP datagram version is not consistent with the Ethernet Type value. This bit is valid when either Bit 7 or Bit 6 is set. This bit is available when you select the Enable Receive Full TCP/IP Checksum (Type 2) feature.
2:0	IP Payload Type These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE). The COE also sets these bits to 2'b00 if it does not process the IP datagram's payload due to an IP header error or fragmented IP. ■ 3'b000: Unknown or did not process IP payload ■ 3'b001: UDP ■ 3'b010: TCP ■ 3'b011: ICMP ■ 3'b1xx: Reserved This bit is valid when either Bit 7 or Bit 6 is set. This bit is available when you select the Enable Receive Full TCP/IP Checksum (Type 2) feature.

Table 6.24-38 Receive Descriptor Fields 4 (RDES4)

RDES6 and RDES7 contain the snapshot of the timestamp. The availability of the snapshot of the timestamp in RDES6 and RDES7 is indicated by Bit 7 in the RDES0 descriptor. The contents of RDES6 and RDES7 are identified in Table 6.24-39 and Table 6.24-40, respectively.

Bit	Description
31:0	RTSL: Receive Frame Timestamp Low This field is updated by DMA with the least significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).

Table 6.24-39 Receive Descriptor Fields 6 (RDES6)

Bit	Description
31:0	RTSH: Receive Frame Timestamp High This field is updated by DMA with the most significant 32 bits of the timestamp captured for the corresponding receive frame. This field is updated by DMA only for the last descriptor of the receive frame which is indicated by Last Descriptor status bit (RDES0[8]).

Table 6.24-40 Receive Descriptor Fields 7 (RDES7)

6.24.7 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
EMAC Base Address: **EMAC0_BA = 0x4001_2000**				
MAC_Configuration	EMAC0_BA+0x0000	R/W	Register 0 (MAC Configuration Register)	0x00008000
MAC_Frame_Filter	EMAC0_BA+0x0004	R/W	Register 1 (MAC Frame Filter)	0x00000000
GMII_Address	EMAC0_BA+0x0010	R/W	Register 4 (GMII Address Register)	0x00000000
GMII_Data	EMAC0_BA+0x0014	R/W	Register 5 (GMII Data Register)	0x00000000
Flow_Control	EMAC0_BA+0x0018	R/W	Register 6 (Flow Control Register)	0x00000000
VLAN_Tag	EMAC0_BA+0x001c	R/W	Register 7 (VLAN Tag Register)	0x00000000
Version	EMAC0_BA+0x0020	R	Register 8 (Version Register)	0x00001037
Debug	EMAC0_BA+0x0024	R	Register 9 (Debug Register)	0x00000000
PMT_Control_Status	EMAC0_BA+0x002c	R/W	Register 11 (PMT Control and Status Register)	0x00000000
Interrupt_Status	EMAC0_BA+0x0038	R	Register 14 (Interrupt Register)	0x00000000
Interrupt_Mask	EMAC0_BA+0x003c	R/W	Register 15 (Interrupt Mask Register)	0x00000000
MAC_Address0_High	EMAC0_BA+0x0040	R/W	Register 16 (MAC Address0 High Register)	0x8000FFFF
MAC_Address0_Low	EMAC0_BA+0x0044	R/W	Register 17 (MAC Address0 Low Register)	0xFFFFFFFF
MAC_Address1_High	EMAC0_BA+0x0048	R/W	Register 18 (MAC Address1 High Register)	0x0000FFFF
MAC_Address1_Low	EMAC0_BA+0x004c	R/W	Register 19 (MAC Address1 Low Register)	0xFFFFFFFF
MAC_Address2_High	EMAC0_BA+0x0050	R/W	Register 20 (MAC Address2 High Register)	0x0000FFFF
MAC_Address2_Low	EMAC0_BA+0x0054	R/W	Register 21 (MAC Address2 Low Register)	0xFFFFFFFF
MAC_Address3_High	EMAC0_BA+0x0058	R/W	Register 22 (MAC Address3 High Register)	0x0000FFFF
MAC_Address3_Low	EMAC0_BA+0x005c	R/W	Register 23 (MAC Address3 Low Register)	0xFFFFFFFF
MAC_Address4_High	EMAC0_BA+0x0060	R/W	Register 24 (MAC Address4 High Register)	0x0000FFFF
MAC_Address4_Low	EMAC0_BA+0x0064	R/W	Register 25 (MAC Address4 Low Register)	0xFFFFFFFF
MAC_Address5_High	EMAC0_BA+0x0068	R/W	Register 26 (MAC Address5 High Register)	0x0000FFFF
MAC_Address5_Low	EMAC0_BA+0x006c	R/W	Register 27 (MAC Address5 Low Register)	0xFFFFFFFF

MAC_Address6_High	EMAC0_BA+0x0070	R/W	Register 28 (MAC Address6 High Register)	0x0000FFFF
MAC_Address6_Low	EMAC0_BA+0x0074	R/W	Register 29 (MAC Address6 Low Register)	0xFFFFFFFF
MAC_Address7_High	EMAC0_BA+0x0078	R/W	Register 30 (MAC Address7 High Register)	0x0000FFFF
MAC_Address7_Low	EMAC0_BA+0x007c	R/W	Register 31 (MAC Address7 Low Register)	0xFFFFFFFF
MAC_Address8_High	EMAC0_BA+0x0080	R/W	Register 32 (MAC Address8 High Register)	0x0000FFFF
MAC_Address8_Low	EMAC0_BA+0x0084	R/W	Register 33 (MAC Address8 Low Register)	0xFFFFFFFF
WDog_Timeout	EMAC0_BA+0x00dc	R/W	Register 55 (Watchdog Timeout Register)	0x00000000
VLAN_Incl_Reg	EMAC0_BA+0x0584	R/W	Register 353 (VLAN Tag Inclusion or Replacement Register)	0x00000000
Timestamp_Control	EMAC0_BA+0x0700	R/W	Register 448 (Timestamp Control Register)	0x00002000
Sub_Second_Increment	EMAC0_BA+0x0704	R/W	Register 449 (Sub-Second Increment Register)	0x00000000
System_Time_Seconds	EMAC0_BA+0x0708	R	Register 450 (System Time - Seconds Register)	0x00000000
System_Time_Nanoseconds	EMAC0_BA+0x070c	R	Register 451 (System Time - Nanoseconds Register)	0x00000000
System_Time_Seconds_Update	EMAC0_BA+0x0710	R/W	Register 452 (System Time - Seconds Update Register)	0x00000000
System_Time_Nanoseconds_Update	EMAC0_BA+0x0714	R/W	Register 453 (System Time - Nanoseconds Update Register)	0x00000000
Timestamp_Addend	EMAC0_BA+0x0718	R/W	Register 454 (Timestamp Addend Register)	0x00000000
Target_Time_Seconds	EMAC0_BA+0x071c	R/W	Register 455 (Target Time Seconds Register)	0x00000000
Target_Time_Nanoseconds	EMAC0_BA+0x0720	R/W	Register 456 (Target Time Nanoseconds Register)	0x00000000
System_Time_Higher_Word_Seconds	EMAC0_BA+0x0724	R/W	Register 457 (System Time - Higher Word Seconds Register)	0x00000000
Timestamp_Status	EMAC0_BA+0x0728	R	Register 458 (Timestamp Status Register)	0x00000000
PPS_Control	EMAC0_BA+0x072c	R/W	Register 459 (PPS Control Register)	0x00000000
PPS0_Interval	EMAC0_BA+0x0760	R/W	Register 472 (PPS0 Interval Register)	0x00000000
PPS0_Width	EMAC0_BA+0x0764	R/W	Register 473 (PPS0 Width Register)	0x00000000
Bus_Mode	EMAC0_BA+0x1000	R/W	Register 1000 (Bus Mode Register)	0x00020100

Transmit_Poll_Demand	EMAC0_BA+0x1004	R/W	Register 1001 (Transmit Poll Demand Register)	0x00000000
Receive_Poll_Demand	EMAC0_BA+0x1008	R/W	Register 1002 (Receive Poll Demand Register)	0x00000000
Receive_Descriptor_List_Address	EMAC0_BA+0x100c	R/W	Register 1003 (Receive Descriptor List Address Register)	0x00000000
Transmit_Descriptor_List_Address	EMAC0_BA+0x1010	R/W	Register 1004 (Transmit Descriptor List Address Register)	0x00000000
Status	EMAC0_BA+0x1014	R/W	Register 1005 (Status Register)	0x00000000
Operation_Mode	EMAC0_BA+0x1018	R/W	Register 1006 (Operation Mode Register)	0x00000000
Interrupt_Enable	EMAC0_BA+0x101c	R/W	Register 1007 (Interrupt Enable Register)	0x00000000
Missed_Frame_And_Buffer_Overflow_Counter	EMAC0_BA+0x1020	R	Register 1008 (Missed Frame and Buffer Overflow Counter Register)	0x00000000
Receive_Interrupt_Watchdog_Timer	EMAC0_BA+0x1024	R/W	Register 1009 (Receive Interrupt Watchdog Timer Register)	0x00000000
AHB_Status	EMAC0_BA+0x102c	R	Register 1011 (AHB Status Register)	0x00000000
Current_Host_Transmit_Descriptor	EMAC0_BA+0x1048	R	Register 1018 (Current Host Transmit Descriptor Register)	0x00000000
Current_Host_Receive_Descriptor	EMAC0_BA+0x104c	R	Register 1019 (Current Host Receive Descriptor Register)	0x00000000
Current_Host_Transmit_Buffer_Address	EMAC0_BA+0x1050	R	Register 1020 (Current Host Transmit Buffer Address Register)	0x00000000
Current_Host_Receive_Buffer_Address	EMAC0_BA+0x1054	R	Register 1021 (Current Host Receive Buffer Address Register)	0x00000000
HW_Feature	EMAC0_BA+0x1058	R	Register 1022 (HW Feature Register)	0x4F0D6525

6.24.8 Register Description

EMAC Register 0 MAC Configuration Register (MAC_Configuration)

Register	Offset	R/W	Description	Reset Value
MAC_Configuration	EMAC0_BA+0x0000	R/W	Register 0 (MAC Configuration Register)	0x00008000

31	30	29	28	27	26	25	24
Reserved	SARC			TWOKPE	Reserved	CST	Reserved
23	22	21	20	19	18	17	16
WD	JD	Reserved	JE	IFG			DCRS
15	14	13	12	11	10	9	8
Reserved	FES	DO	LM	DM	IPC	DR	Reserved
7	6	5	4	3	2	1	0
ACS	BL		DC	TE	RE	PRELEN	

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**SARC**	**Source Address Insertion or Replacement Control** This field controls the source address insertion or replacement for all transmitted frames. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits [29:28]: * 2'b0x = The input signals control the SA field generation. * 2'b10: • If Bit 30 is set to 0, the MAC inserts the content of the MAC Address 0 registers (registers 16 and 17) in the SA field of all transmitted frames. • If Bit 30 is set to 1, the MAC inserts the content of the MAC Address 1 registers (registers 18 and 19) in the SA field of all transmitted frames. * 2'b11 =. • If Bit 30 is set to 0, the MAC replaces the content of the MAC Address 0 registers (registers 16 and 17) in the SA field of all transmitted frames. • If Bit 30 is set to 1, the MAC replaces the content of the MAC Address 1 registers (registers 18 and 19) in the SA field of all transmitted frames. **Note:** • Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value.
[27]	**TWOKPE**	**IEEE 802.3as Support for 2K Packets** When set, the MAC considers all frames, with up to 2,000 bytes length, as normal packets. When Bit 20 (JE) is not set, the MAC considers all received frames of size more than 2 Kbytes as Giant frames. When this bit is reset and Bit 20 (JE) is not set, the MAC considers all received frames of size more than 1,518 bytes (1,522 bytes for tagged) as Giant frames. When Bit 20 is set, setting this bit has no effect on Giant Frame status.
[26]	**Reserved**	Reserved.

[25]	**CST**	**CRC Stripping for Type Frames** When this bit is set, the last 4 bytes (FCS) of all frames of Ether type (Length/Type field greater than or equal to 1,536) are stripped and dropped before forwarding the frame to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled.
[24]	**Reserved**	Reserved.
[23]	**WD**	**Watchdog Disable** When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive frames of up to 16,384 bytes. When this bit is reset, the MAC does not allow a receive frame which more than 2,048 bytes (10,240 if JE is set high) or the value programmed in Register 55 (Watchdog Timeout Register). The MAC cuts off any bytes received after the watchdog limit number of bytes.
[22]	**JD**	**Jabber Disable** When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer frames of up to 16,384 bytes. When this bit is reset, the MAC cuts off the transmitter if the application sends out more than 2,048 bytes of data (10,240 if JE is set high) during transmission.
[21]	**Reserved**	Reserved.
[20]	**JE**	**Jumbo Frame Enable** When this bit is set, the MAC allows Jumbo frames of 9,018 bytes (9,022 bytes for VLAN tagged frames) without reporting a giant frame error in the receive frame status.
[19:17]	**IFG**	**Inter-Frame Gap** These bits control the minimum IFG between frames during transmission. 000 = 96 bit times. 001 = 88 bit times. 010 = 80 bit times. ... 111 = 40 bit times. In the half-duplex mode, the minimum IFG can be configured only for 64 bit times (IFG = 100). Lower values are not considered. In the 1000-Mbps mode, the minimum IFG supported is 80 bit times (and above).
[16]	**DCRS**	**Disable Carrier Sense During Transmission** When set high, this bit makes the MAC transmitter ignore the (G)MII CRS signal during frame transmission in the half-duplex mode. This request results in no errors generated because of Loss of Carrier or No Carrier during such transmission. When this bit is low, the MAC transmitter generates such errors because of Carrier Sense and can even abort the transmissions.
[15]	**Reserved**	Reserved.
[14]	**FES**	**Speed** This bit selects the speed in the RMII interface 0 = 10 Mbps. 1 = 100 Mbps.
[13]	**DO**	**Disable Receive Own** When this bit is set, the MAC disables the reception of frames when the phy_txen_o is asserted in the half-duplex mode. When this bit is reset, the MAC receives all packets that are given by the PHY while transmitting. This bit is not applicable if the MAC is operating in the full-duplex mode.
[12]	**LM**	**Loopback Mode** When this bit is set, the MAC operates in the loopback mode at MII. The (G)MII Receive clock input is required for the loopback to work properly, because the Transmit clock is not looped-back internally.
[11]	**DM**	**Duplex Mode** When this bit is set, the MAC operates in the full-duplex mode where it can transmit and receive simultaneously.

[10]	**IPC**	**Checksum Offload** When this bit is set, the MAC calculates the 16-bit one's complement of the one's complement sum of all received Ethernet frame payloads. It also checks whether the IPv4 Header checksum (assumed to be bytes 2526 or 2930 (VLAN-tagged) of the received Ethernet frame) is correct for the received frame and gives the status in the receive status word. The MAC also appends the 16-bit checksum calculated for the IP header datagram payload (bytes after the IPv4 header) and appends it to the Ethernet frame transferred to the application (when Type 2 COE is deselected). When this bit is reset, this function is disabled. When Type 2 COE is selected, this bit, when set, enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled and the corresponding PCE and IP HCE status bits are always cleared.
[9]	**DR**	**Disable Retry** When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current frame transmission and reports a Frame Abort with excessive collision error in the transmit frame status. When this bit is reset, the MAC attempts retries based on the settings of the BL field (Bits [6:5]). This bit is applicable only in the half-duplex mode.
[8]	**Reserved**	Reserved.
[7]	**ACS**	**Automatic Pad or CRC Stripping** When this bit is set, the MAC strips the Pad or FCS field on the incoming frames only if the value of the length field is less than 1,536 bytes. All received frames with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field. When this bit is reset, the MAC passes all incoming frames, without modifying them, to the Host.
[6:5]	**BL**	**Back-Off Limit** The Back-Off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision. This bit is applicable only in the half-duplex mode. 00 = k is the min of (n, 10). 01 = k is the min of (n, 8). 10 = k is the min of (n, 4). 11 = k is the min of (n, 1). where *n* = retransmission attempt. The random integer *r* takes the value in the range 0 <= r < kth power of 2.
[4]	**DC**	**Deferral Check** When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Frame Abort status, along with the excessive deferral error bit set in the transmit frame status, when the transmit state machine is deferred for more than 24,288 bit times in the 10 or 100 Mbps mode. If the MAC is configured for 1000 Mbps operation or if the Jumbo frame mode is enabled in the 10 or 100 Mbps mode, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on MII. The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and then the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0 and it is restarted. When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive. This bit is applicable only in the half-duplex mode.
[3]	**TE**	**Transmitter Enable** When this bit is set, the transmit state machine of the MAC is enabled for transmission on the MII. When this bit is reset, the MAC transmit state machine is disabled after the completion of the transmission of the current frame, and does not transmit any further frames.
[2]	**RE**	**Receiver Enable** When this bit is set, the receiver state machine of the MAC is enabled for receiving frames from the MII. When this bit is reset, the MAC receive state machine is disabled after the completion of the reception of the current frame, and does not receive any further frames from the MII.

[1:0]	**PRELEN**	**Preamble Length for Transmit Frames** These bits control the number of preamble bytes that are added to the beginning of every Transmit frame. The preamble reduction occurs only when the MAC is operating in the full-duplex mode. 2'b00 = 7 bytes of preamble. 2'b01 = 5 byte of preamble. 2'b10 = 3 bytes of preamble. 2'b11 = reserved.

EMAC Register 1 MAC Frame Filter)Register (MAC_Frame_Filter)

Register	Offset	R/W	Description	Reset Value
MAC_Frame_Filter	EMAC0_BA+0x0004	R/W	Register 1 (MAC Frame Filter)	0x00000000

31	30	29	28	27	26	25	24
RA	Reserved						
23	22	21	20	19	18	17	16
Reserved							VTFE
15	14	13	12	11	10	9	8
Reserved						SAF	SAIF
7	6	5	4	3	2	1	0
PCF		DBF	PM	DAIF	Reserved		PR

Bits	Description	
[31]	**RA**	**Receive All** When this bit is set, the MAC Receiver module passes all received frames, irrespective of whether they pass the address filter or not, to the Application. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bits in the Receive Status Word. When this bit is reset, the Receiver module passes only those frames to the Application that pass the SA or DA address filter.
[30:17]	**Reserved**	Reserved.
[16]	**VTFE**	**VLAN Tag Filter Enable** When set, this bit enables the MAC to drop VLAN tagged frames that do not match the VLAN Tag comparison. When reset, the MAC forwards all frames irrespective of the match status of the VLAN Tag.
[15:10]	**Reserved**	Reserved.
[9]	**SAF**	**Source Address Filter Enable** When this bit is set, the MAC compares the SA field of the received frames with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the frame. When this bit is reset, the MAC forwards the received frame to the application with updated SAF bit of the Rx Status depending on the SA address comparison. **Note:** According to the IEEE specification, Bit 47 of the SA is reserved and set to 0. However, in EMAC, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.
[8]	**SAIF**	**SA Inverse Filtering** When this bit is set, the Address Check block operates in inverse filtering mode for the SA address comparison. The frames whose SA matches the SA registers are marked as failing the SA Address filter. When this bit is reset, frames whose SA does not match the SA registers are marked as failing the SA Address filter.

[7:6]	**PCF**	**Pass Control Frames** These bits control the forwarding of all control frames (including unicast and multicast PAUSE frames). 00 = MAC filters all control frames from reaching the application. 01 = MAC forwards all control frames except PAUSE control frames to application even if they fail the Address filter. 10 = MAC forwards all control frames to application even if they fail the Address Filter. 11 = MAC forwards control frames that pass the Address Filter. The following conditions should be true for the PAUSE control frames processing: • Condition 1: The MAC is in the full-duplex mode and flow control is enabled by setting Bit 2 (RFE) of Register 6 (Flow Control Register) to 1. • Condition 2: The destination address (DA) of the received frame matches the special multicast address or the MAC Address 0 when Bit 3 (UP) of the Register 6 (Flow Control Register) is set. • Condition 3: The Type field of the received frame is 0x8808 and the OPCODE field is 0x0001. **Note:** This field should be set to 01 only when the Condition 1 is true, that is, the MAC is programmed to operate in the full-duplex mode and the RFE bit is enabled. Otherwise, the PAUSE frame filtering may be inconsistent. When Condition 1 is false, the PAUSE frames are considered as generic control frames. Therefore, to pass all control frames (including PAUSE control frames) when the full-duplex mode and flow control is not enabled, you should set the PCF field to 10 or 11 (as required by the application).
[5]	**DBF**	**Disable Broadcast Frames** When this bit is set, the AFM module filters all incoming broadcast frames. In addition, it overrides all other filter settings. When this bit is reset, the AFM module passes all received broadcast frames.
[4]	**PM**	**Pass All Multicast** When set, this bit indicates that all received frames with a multicast destination address (first bit in the destination address field is '1') are passed. When reset, filtering of multicast frame depends on HMC bit.
[3]	**DAIF**	**DA Inverse Filtering** When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast frames. When reset, normal filtering of frames is performed.
[2:1]	**Reserved**	Reserved.
[0]	**PR**	**Promiscuous Mode** When this bit is set, the Address Filter module passes all incoming frames regardless of its destination or source address. The SA or DA Filter Fails status bits of the Receive Status Word are always cleared when PR is set.

EMAC Register 4 GMII Address Register (GMII_Address)

Register	Offset	R/W	Description	Reset Value
GMII_Address	EMAC0_BA+0x0010	R/W	Register 4 (GMII Address Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
PA					GR		
7	6	5	4	3	2	1	0
GR		CR				GW	GB

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:11]	**PA**	**Physical Layer Address** This field indicates which of the 32 possible PHY devices are being accessed.
[10:6]	**GR**	**GMII Register** These bits select the desired GMII register in the selected PHY device.
[5:2]	**CR**	**CSR Clock Range** The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency used in your design. The suggested range of CSR clock frequency applicable for each value (when Bit[5] = 0) ensures that the MDC clock is approximately between the frequency range 1.0 MHz - 2.5 MHz. 0000 = The frequency of the CSR clock is 60-100 MHz and the MDC clock is CSR clock/42. 0001 = The frequency of the CSR clock is 100-150 MHz and the MDC clock is CSR clock/62. 0010 = The frequency of the CSR clock is 20-35 MHz and the MDC clock is CSR clock/16. 0011 = The frequency of the CSR clock is 35-60 MHz and the MDC clock is CSR clock/26. 0100 = The frequency of the CSR clock is 150-250 MHz and the MDC clock is CSR clock/102. 0101 = The frequency of the CSR clock is 250-300 MHz and the MDC clock is CSR clock/124. 0110 and 0111 = Reserved. When Bit 5 is set, you can achieve MDC clock of frequency higher than the IEEE 802.3 specified frequency limit of 2.5 MHz and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits as 1010, then the resultant MDC clock is of 12.5 MHz which is outside the limit of IEEE 802.3 specified range. Program the following values only if the interfacing chips support faster MDC clocks: 1000 = CSR clock/4. 1001 = CSR clock/6. 1010 = CSR clock/8. 1011 = CSR clock/10. 1100 = CSR clock/12. 1101 = CSR clock/14. 1110 = CSR clock/16. 1111 = CSR clock/18.

[1]	**GW**	**GMII Write** When set, this bit indicates to the PHY that this is a Write operation using the GMII Data register. If this bit is not set, it indicates that this is a Read operation, that is, placing the data in the GMII Data register.
[0]	**GB**	**GMII Busy** This bit should read logic 0 before writing to Register 4 and Register 5. During a PHY register access, the software sets this bit to 1'b1 to indicate that a Read or Write access is in progress. Register 5 is invalid until this bit is cleared by the MAC. Therefore, Register 5 (GMII Data) should be kept valid until the MAC clears this bit during a PHY Write operation. Similarly for a read operation, the contents of Register 5 are not valid until this bit is cleared. The subsequent read or write operation should happen only after the previous operation is complete. Because there is no acknowledgment from the PHY to MAC after a read or write operation is completed, there is no change in the functionality of this bit even when the PHY is not present.

EMAC Register 5 GMII Data Register (GMII_Data)

Register	Offset	R/W	Description	Reset Value
GMII_Data	EMAC0_BA+0x0014	R/W	Register 5 (GMII Data Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
GD							
7	6	5	4	3	2	1	0
GD							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	GD	**GMII Data** This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

EMAC Register 6 Flow Control Register (Flow_Control)

Register	Offset	R/W	Description	Reset Value
Flow_Control	EMAC0_BA+0x0018	R/W	Register 6 (Flow Control Register)	0x00000000

31	30	29	28	27	26	25	24
PT							
23	22	21	20	19	18	17	16
PT							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DZPQ	Reserved	PLT		UP	RFE	TFE	FCA_BPA

Bits	Description	
[31:16]	**PT**	**Pause Time** This field holds the value to be used in the Pause Time field in the transmit control frame. Consecutive writes to this register should be performed only after at least four clock cycles in the destination clock domain.
[15:8]	**Reserved**	Reserved.
[7]	**DZPQ**	**Disable Zero-Quanta Pause** When this bit is set, it disables the automatic generation of the Zero-Quanta Pause Control frames on the de-assertion of the flow-control signal from the FIFO layer. When this bit is reset, normal operation with automatic Zero-Quanta Pause Control frame generation is enabled.
[6]	**Reserved**	Reserved.
[5:4]	**PLT**	**Pause Low Threshold** This field configures the threshold of the PAUSE timer at which the input flow control signal is checked for automatic retransmission of PAUSE Frame. The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot-times), and PLT = 01, then a second PAUSE frame is automatically transmitted if the mti_flowctrl_i signal is asserted at 228 (256 - 28) slot times after the first PAUSE frame is transmitted. The following list provides the threshold values for different values: 00 = The threshold is Pause time minus 4 slot times (PT - 4 slot times). 01 = The threshold is Pause time minus 28 slot times (PT - 28 slot times). 10 = The threshold is Pause time minus 144 slot times (PT - 144 slot times). 11 = The threshold is Pause time minus 256 slot times (PT - 256 slot times). The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface.
[3]	**UP**	**Unicast Pause Frame Detect** A pause frame is processed when it has the unique multicast address specified in the IEEE Std 802.3. When this bit is set, the MAC can also detect Pause frames with unicast address of the station. This unicast address should be as specified in the MAC Address0 High Register and MAC Address0 Low Register. When this bit is reset, the MAC only detects Pause frames with unique multicast address. **Note:** The MAC does not process a Pause frame if the multicast address of received frame is different from the unique multicast address.

[2]	**RFE**	**Receive Flow Control Enable** When this bit is set, the MAC decodes the received Pause frame and disables its transmitter for a specified (Pause) time. When this bit is reset, the decode function of the Pause frame is disabled.
[1]	**TFE**	**Transmit Flow Control Enable** In the full-duplex mode, when this bit is set, the MAC enables the flow control operation to transmit Pause frames. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause frames. In half-duplex mode, when this bit is set, the MAC enables the back-pressure operation. When this bit is reset, the back-pressure feature is disabled.
[0]	**FCA_BPA**	**Flow Control Busy or Backpressure Activate** This bit initiates a Pause Control frame in the full-duplex mode and activates the backpressure function in the half-duplex mode if the TFE bit is set. In the full-duplex mode, this bit should be read as 1'b0 before writing to the Flow Control register. To initiate a Pause control frame, the Application must set this bit to 1'b1. During a transfer of the Control Frame, this bit continues to be set to signify that a frame transmission is in progress. After the completion of Pause control frame transmission, the MAC resets this bit to 1'b0. The Flow Control register should not be written to until this bit is cleared. In the half-duplex mode, when this bit is set (and TFE is set), then backpressure is asserted by the MAC. During backpressure, when the MAC receives a new frame, the transmitter starts sending a JAM pattern resulting in a collision. This control register bit is logically ORed with the mti_flowctrl_i input signal for the backpressure function. When the MAC is configured for the full-duplex mode, the BPA is automatically disabled.

EMAC Register 7 VLAN Tag Register (VLAN_Tag)

Register	Offset	R/W	Description	Reset Value
VLAN_Tag	EMAC0_BA+0x001c	R/W	Register 7 (VLAN Tag Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					ESVL	VTIM	ETV
15	14	13	12	11	10	9	8
VL							
7	6	5	4	3	2	1	0
VL							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**ESVL**	**Enable S-VLAN** When this bit is set, the MAC transmitter and receiver also consider the S-VLAN (Type = 0x88A8) frames as valid VLAN tagged frames.
[17]	**VTIM**	**VLAN Tag Inverse Match Enable** When set, this bit enables the VLAN Tag inverse matching. The frames that do not have matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The frames with matched VLAN Tag are marked as matched.
[16]	**ETV**	**Enable 12-Bit VLAN Tag Comparison** When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits [11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged frame. Similarly, when enabled, only 12 bits of the VLAN tag in the received frame are used for hash-based VLAN filtering. When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN frame are used for comparison and VLAN hash filtering.
[15:0]	**VL**	**VLAN Tag Identifier for Receive Frames** This field contains the 802.1Q VLAN tag to identify the VLAN frames and is compared to the 15th and 16th bytes of the frames being received for VLAN frames. The following list describes the bits of this field: • Bits [15:13]: User Priority • Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI) • Bits [11:0]: VLAN tag's VLAN Identifier (VID) field When the ETV bit is set, only the VID (Bits [11:0]) is used for comparison. If VL (VL[11:0] if ETV is set) is all zeros, the MAC does not check the fifteenth and 16th bytes for VLAN tag comparison, and declares all frames with a Type field value of 0x8100 or 0x88a8 as VLAN frames.

EMAC Register 9 Debug Register (Debug)

Register	Offset	R/W	Description	Reset Value
Debug	EMAC0_BA+0x0024	R	Register 9 (Debug Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved						TXSTSFSTS	TXFSTS
23	22	21	20	19	18	17	16
Reserved	TWCSTS	TRCSTS		TXPAUSED	TFCSTS		TPESTS
15	14	13	12	11	10	9	8
Reserved						RXFSTS	
7	6	5	4	3	2	1	0
Reserved	RRCSTS		RWCSTS	Reserved	RFCFCSTS		RPESTS

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**TXSTSFSTS**	**MTL TxStatus FIFO Full Status** When high, this bit indicates that the MTL TxStatus FIFO is full. Therefore, the MTL cannot accept any more frames for transmission.
[24]	**TXFSTS**	**MTL Tx FIFO Not Empty Status** When high, this bit indicates that the MTL Tx FIFO is not empty and some data is left for transmission.
[23]	**Reserved**	Reserved.
[22]	**TWCSTS**	**MTL Tx FIFO Write Controller Active Status** When high, this bit indicates that the MTL Tx FIFO Write Controller is active and transferring data to the Tx FIFO.
[21:20]	**TRCSTS**	**MTL Tx FIFO Read Controller Status** This field indicates the state of the Tx FIFO Read Controller: 00 = IDLE state. 01 = READ state (transferring data to MAC transmitter). 10 = Waiting for TxStatus from MAC transmitter. 11 = Writing the received TxStatus or flushing the Tx FIFO.
[19]	**TXPAUSED**	**MAC transmitter in PAUSE** When high, this bit indicates that the MAC transmitter is in the PAUSE condition (in the full-duplex only mode) and hence does not schedule any frame for transmission.
[18:17]	**TFCSTS**	**MAC Transmit Frame Controller Status** This field indicates the state of the MAC Transmit Frame Controller module: 00 = IDLE state. 01 = Waiting for Status of previous frame or IFG or backoff period to be over. 10 = Generating and transmitting a PAUSE control frame (in the full-duplex mode). 11 = Transferring input frame for transmission.

[16]	**TPESTS**	**MAC MII Transmit Protocol Engine Status** When high, this bit indicates that the MAC MII transmit protocol engine is actively transmitting data and is not in the IDLE state.
[15:10]	**Reserved**	Reserved.
[9:8]	**RXFSTS**	**MTL Rx FIFO Fill-level Status** This field gives the status of the fill-level of the Rx FIFO: 00 = Rx FIFO Empty. 01 = Rx FIFO fill level is below the flow-control deactivate threshold. 10 = Rx FIFO fill level is above the flow-control activate threshold. 11 = Rx FIFO Full.
[7]	**Reserved**	Reserved.
[6:5]	**RRCSTS**	**MTL Rx FIFO Read Controller State** This field gives the state of the Rx FIFO read Controller: 00 = IDLE state. 01 = Reading frame data. 10 = Reading frame status (or timestamp). 11 = Flushing the frame data and status.
[4]	**RWCSTS**	**MTL Rx FIFO Write Controller Active Status** When high, this bit indicates that the MTL Rx FIFO Write Controller is active and is transferring a received frame to the FIFO.
[3]	**Reserved**	Reserved.
[2:1]	**RFCFCSTS**	**MAC Receive Frame Controller FIFO Status** When high, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Frame Controller Module.
[0]	**RPESTS**	**MAC MII Receive Protocol Engine Status** When high, this bit indicates that the MAC MII receive protocol engine is actively receiving data and not in IDLE state.

EMAC Register 11 PMT Control and Status Register (PMT_Control_Status)

Register	Offset	R/W	Description	Reset Value
PMT_Control_Status	EMAC0_BA+0x002c	R/W	Register 11 (PMT Control and Status Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		MGKPRCVD	Reserved			MGKPKTEN	PWRDWN

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**MGKPRCVD**	**Magic Packet Received (read only)** When set, this bit indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared by a Read into this register.
[4:2]	**Reserved**	Reserved.
[1]	**MGKPKTEN**	**Magic Packet Enable** When set, enables generation of a power management event because of magic packet reception.
[0]	**PWRDWN**	**Power Down** When set, the MAC receiver drops all received frames until it receives the expected magic packet or wake-up frame. This bit is then self-cleared and the power-down mode is disabled. The Software can also clear this bit before the expected magic packet or wake-up frame is received. The frames, received by the MAC after this bit is cleared, are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Wake-Up Frame Enable bit is set high. **Note:** You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.

EMAC Register 14 Interrupt Register (Interrupt_Status)

Register	Offset	R/W	Description	Reset Value
Interrupt_Status	EMAC0_BA+0x0038	R	Register 14 (Interrupt Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						TSIS	Reserved
7	6	5	4	3	2	1	0
Reserved				PMTIS	Reserved		

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**TSIS**	**Timestamp Interrupt Status** When the Advanced Timestamp feature is enabled, this bit is set when any of the following conditions is true: • The system time value equals or exceeds the value specified in the Target Time High and Low registers. • There is an overflow in the seconds register. • The Auxiliary snapshot trigger is asserted. This bit is cleared on reading Bit 0 of the Register 458 (Timestamp Status Register). If default Timestamping is enabled, when set, this bit indicates that the system time value is equal to or exceeds the value specified in the Target Time registers. In this mode, this bit is cleared after the completion of the read of this bit. In all other modes, this bit is reserved.
[8:4]	**Reserved**	Reserved.
[3]	**PMTIS**	**PMT Interrupt Status** This bit is set when a Magic packet or Wake-on-LAN frame is received in the power-down mode (see Bits 5 in the PMT Control and Status Register). This bit is cleared when Bit 5 is cleared because of a read operation to the PMT Control and Status register.
[2:0]	**Reserved**	Reserved.

EMAC Register 15 Interrupt Mask Register (Interrupt_Mask)

Register	Offset	R/W	Description	Reset Value
Interrupt_Mask	EMAC0_BA+0x003c	R/W	Register 15 (Interrupt Mask Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						TSIM	Reserved
7	6	5	4	3	2	1	0
Reserved				PMTIM	Reserved		

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**TSIM**	**Timestamp Interrupt Mask** When set, this bit disables the assertion of the interrupt signal because of the setting of Timestamp Interrupt Status bit in Register 14 (Interrupt Status Register). This bit is valid only when IEEE1588 timestamping is enabled. In all other modes, this bit is reserved.
[8:4]	**Reserved**	Reserved.
[3]	**PMTIM**	**PMT Interrupt Mask** When set, this bit disables the assertion of the interrupt signal because of the setting of PMT Interrupt Status bit in Register 14 (Interrupt Status Register).
[2:0]	**Reserved**	Reserved.

EMAC Register 16 MAC Address0 High Register (MAC_Address0_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address0_High	EMAC0_BA+0x0040	R/W	Register 16 (MAC Address0 High Register)	0x8000FFFF

31	30	29	28	27	26	25	24
AE	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable (read only)** This bit is always set to 1.
[30:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address0 [47:32]** This field contains the upper 16 bits (47:32) of the first 6-byte MAC address. The MAC uses this field for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

EMAC Register 17 MAC Address0 Low Register (MAC_Address0_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address0_Low	EMAC0_BA+0x0044	R/W	Register 17 (MAC Address0 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address0 [31:0]** This field contains the lower 32 bits of the first 6-byte MAC address. This is used by the MAC for filtering the received frames and inserting the MAC address in the Transmit Flow Control (PAUSE) Frames.

EMAC Register 18 MAC Address1 High Register (MAC_Address1_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address1_High	EMAC0_BA+0x0048	R/W	Register 18 (MAC Address1 High Register)	0x0000FFFF

31	30	29	28	27	26	25	24
AE	SA	MBC					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable** When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
[30]	**SA**	**Source Address** When this bit is set, the MAC Address1[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address1[47:0] is used to compare with the DA fields of the received frame.
[29:24]	**MBC**	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 18[15:8] • Bit 28: Register 18[7:0] • Bit 27: Register 19[31:24] • ... • Bit 24: Register 19[7:0] You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.
[23:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address1 [47:32]** This field contains the upper 16 bits (47:32) of the second 6-byte MAC address.

EMAC Register 19 MAC Address1 Low Register (MAC_Address1_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address1_Low	EMAC0_BA+0x004c	R/W	Register 19 (MAC Address1 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address1 [31:0]** This field contains the lower 32 bits of the second 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

EMAC Register 20 MAC Address2 High Register (MAC_Address2_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address2_High	EMAC0_BA+0x0050	R/W	Register 20 (MAC Address2 High Register)	0x0000FFFF

31	30	29	28	27	26	25	24
AE	SA	MBC					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable** When this bit is set, the address filter module uses the third MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
[30]	**SA**	**Source Address** When this bit is set, the MAC Address2[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address2[47:0] is used to compare with the DA fields of the received frame.
[29:24]	**MBC**	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address2 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 18[15:8] • Bit 28: Register 18[7:0] • Bit 27: Register 19[31:24] • ... • Bit 24: Register 19[7:0]
[23:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address2 [47:32]** This field contains the upper 16 bits (47:32) of the third 6-byte MAC address.

EMAC Register 21 MAC Address2 Low Register (MAC_Address2_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address2_Low	EMAC0_BA+0x0054	R/W	Register 21 (MAC Address2 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address2 [31:0]** This field contains the lower 32 bits of the third 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

EMAC Register 22 MAC Address3 High Register (MAC_Address3_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address3_High	EMAC0_BA+0x0058	R/W	Register 22 (MAC Address3 High Register)	0x0000FFFF

31	30	29	28	27	26	25	24
AE	SA	MBC					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable** When this bit is set, the address filter module uses the fourth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
[30]	**SA**	**Source Address** When this bit is set, the MAC Address3[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address3[47:0] is used to compare with the DA fields of the received frame.
[29:24]	**MBC**	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address3 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 18[15:8] • Bit 28: Register 18[7:0] • Bit 27: Register 19[31:24] • ... • Bit 24: Register 19[7:0]
[23:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address3 [47:32]** This field contains the upper 16 bits (47:32) of the fourth 6-byte MAC address.

EMAC Register 23 MAC Address3 Low Register (MAC_Address3_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address3_Low	EMAC0_BA+0x005c	R/W	Register 23 (MAC Address3 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address3 [31:0]** This field contains the lower 32 bits of the fourth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

EMAC Register 24 MAC Address4 High Register (MAC_Address4_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address4_High	EMAC0_BA+0x0060	R/W	Register 24 (MAC Address4 High Register)	0x0000FFFF

31	30	29	28	27	26	25	24
AE	SA	MBC					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable** When this bit is set, the address filter module uses the fifth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
[30]	**SA**	**Source Address** When this bit is set, the MAC Address4[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address4[47:0] is used to compare with the DA fields of the received frame.
[29:24]	**MBC**	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address4 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 18[15:8] • Bit 28: Register 18[7:0] • Bit 27: Register 19[31:24] • ... • Bit 24: Register 19[7:0]
[23:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address4 [47:32]** This field contains the upper 16 bits (47:32) of the fifth 6-byte MAC address.

EMAC Register 25 MAC Address4 Low Register (MAC_Address4_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address4_Low	EMAC0_BA+0x0064	R/W	Register 25 (MAC Address4 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address4 [31:0]** This field contains the lower 32 bits of the fifth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

EMAC Register 26 MAC Address5 High Register (MAC_Address5_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address5_High	EMAC0_BA+0x0068	R/W	Register 26 (MAC Address5 High Register)	0x0000FFFF

31	30	29	28	27	26	25	24
AE	SA	MBC					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable** When this bit is set, the address filter module uses the sixth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
[30]	**SA**	**Source Address** When this bit is set, the MAC Address5[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address5[47:0] is used to compare with the DA fields of the received frame.
[29:24]	**MBC**	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address5 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 18[15:8] • Bit 28: Register 18[7:0] • Bit 27: Register 19[31:24] • ... • Bit 24: Register 19[7:0]
[23:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address5 [47:32]** This field contains the upper 16 bits (47:32) of the sixth 6-byte MAC address.

EMAC Register 27 MAC Address5 Low Register (MAC_Address5_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address5_Low	EMAC0_BA+0x006c	R/W	Register 27 (MAC Address5 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address5 [31:0]** This field contains the lower 32 bits of the sixth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

EMAC Register 28 MAC Address6 High Register (MAC_Address6_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address6_High	EMAC0_BA+0x0070	R/W	Register 28 (MAC Address6 High Register)	0x0000FFFF

31	30	29	28	27	26	25	24
AE	SA	MBC					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable** When this bit is set, the address filter module uses the seventh MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
[30]	**SA**	**Source Address** When this bit is set, the MAC Address6[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address6[47:0] is used to compare with the DA fields of the received frame.
[29:24]	**MBC**	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 18[15:8] • Bit 28: Register 18[7:0] • Bit 27: Register 19[31:24] • ... • Bit 24: Register 19[7:0]
[23:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address6 [47:32]** This field contains the upper 16 bits (47:32) of the seventh 6-byte MAC address.

EMAC Register 29 MAC Address6 Low Register (MAC_Address6_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address6_Low	EMAC0_BA+0x0074	R/W	Register 29 (MAC Address6 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address6 [31:0]** This field contains the lower 32 bits of the seventh 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

EMAC Register 30 MAC Address7 High Register (MAC_Address7_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address7_High	EMAC0_BA+0x0078	R/W	Register 30 (MAC Address7 High Register)	0x0000FFFF

31	30	29	28	27	26	25	24
AE	SA	MBC					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable** When this bit is set, the address filter module uses the eighth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
[30]	**SA**	**Source Address** When this bit is set, the MAC Address7[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address7[47:0] is used to compare with the DA fields of the received frame.
[29:24]	**MBC**	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address6 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 18[15:8] • Bit 28: Register 18[7:0] • Bit 27: Register 19[31:24] • ... • Bit 24: Register 19[7:0]
[23:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address7 [47:32]** This field contains the upper 16 bits (47:32) of the eighth 6-byte MAC address.

EMAC Register 31 MAC Address7 Low Register (MAC_Address7_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address7_Low	EMAC0_BA+0x007c	R/W	Register 31 (MAC Address7 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address7 [31:0]** This field contains the lower 32 bits of the eighth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

EMAC Register 32 MAC Address8 High Register (MAC_Address8_High)

Register	Offset	R/W	Description	Reset Value
MAC_Address8_High	EMAC0_BA+0x0080	R/W	Register 32 (MAC Address8 High Register)	0x0000FFFF

31	30	29	28	27	26	25	24
AE	SA	MBC					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ADDRHI							
7	6	5	4	3	2	1	0
ADDRHI							

Bits	Description	
[31]	**AE**	**Address Enable** When this bit is set, the address filter module uses the nineth MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.
[30]	**SA**	**Source Address** When this bit is set, the MAC Address8[47:0] is used to compare with the SA fields of the received frame. When this bit is reset, the MAC Address8[47:0] is used to compare with the DA fields of the received frame.
[29:24]	**MBC**	**Mask Byte Control** These bits are mask control bits for comparison of each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address8 registers. Each bit controls the masking of the bytes as follows: • Bit 29: Register 18[15:8] • Bit 28: Register 18[7:0] • Bit 27: Register 19[31:24] • ... • Bit 24: Register 19[7:0]
[23:16]	**Reserved**	Reserved.
[15:0]	**ADDRHI**	**MAC Address8 [47:32]** This field contains the upper 16 bits (47:32) of the nineth 6-byte MAC address.

EMAC Register 33 MAC Address8 Low Register (MAC_Address8_Low)

Register	Offset	R/W	Description	Reset Value
MAC_Address8_Low	EMAC0_BA+0x0084	R/W	Register 33 (MAC Address8 Low Register)	0xFFFFFFFF

31	30	29	28	27	26	25	24
ADDRLO							
23	22	21	20	19	18	17	16
ADDRLO							
15	14	13	12	11	10	9	8
ADDRLO							
7	6	5	4	3	2	1	0
ADDRLO							

Bits	Description	
[31:0]	ADDRLO	**MAC Address8 [31:0]** This field contains the lower 32 bits of the nineth 6-byte MAC address. The content of this field is undefined until loaded by the Application after the initialization process.

EMAC Register 55 Watchdog Timeout Register (WDog_Timeout)

Register	Offset	R/W	Description	Reset Value
WDog_Timeout	EMAC0_BA+0x00dc	R/W	Register 55 (Watchdog Timeout Register)	0x00000000

Note: This register controls the watchdog timeout for received frames.

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							PWE
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		WTO					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WTO							

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**PWE**	**Programmable Watchdog Enable** When this bit is set and Bit 23 (WD) of Register 0 (MAC Configuration Register) is reset, the WTO field (Bits[13:0]) is used as watchdog timeout for a received frame. When this bit is cleared, the watchdog timeout for a received frame is controlled by the setting of Bit 23 (WD) and Bit 20 (JE) in Register 0 (MAC Configuration Register).
[15:14]	**Reserved**	Reserved.
[13:0]	**WTO**	**Watchdog Timeout** When Bit 16 (PWE) is set and Bit 23 (WD) of Register 0 (MAC Configuration Register) is reset, this field is used as watchdog timeout for a received frame. If the length of a received frame exceeds the value of this field, such frame is terminated and declared as an error frame. **Note:** When Bit 16 (PWE) is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE Std 802.3-specified valid tagged frames are declared as error frames and are dropped.

EMAC Register 353 VLAN Tag Inclusion or Replacement Register (VLAN_Incl_Reg)

Register	Offset	R/W	Description	Reset Value
VLAN_Incl_Reg	EMAC0_BA+0x0584	R/W	Register 353 (VLAN Tag Inclusion or Replacement Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				CSVL	VLP	VLC	
15	14	13	12	11	10	9	8
VLT							
7	6	5	4	3	2	1	0
VLT							

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19]	**CSVL**	**C-VLAN or S-VLAN** When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted frames. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the transmitted frames.
[18]	**VLP**	**VLAN Priority Control** When this bit is set, the control Bits [17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, the mti_vlan_ctrl_i control input is used, and Bits [17:16] are ignored.
[17:16]	**VLC**	**VLAN Tag Control in Transmit Frames** 2'b00 = No VLAN tag deletion, insertion, or replacement. 2'b01 = VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted frames with VLAN tags. 2'b10 = VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the frame after inserting the Type value (0x8100/0x88a8) in bytes 13 and 14. This operation is performed on all transmitted frames, irrespective of whether they already have a VLAN tag. 2'b11 = VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted frames (Bytes 13 and 14 are 0x8100/0x88a8). **Note:** Changes to this field take effect only on the start of a frame. If you write this register field when a frame is being transmitted, only the subsequent frame can use the updated value, that is, the current frame does not use the updated value
[15:0]	**VLT**	**VLAN Tag for Transmit Frames** This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase. Bits [15:13] are the User Priority, Bit 12 is the CFI/DEI, and Bits [11:0] are the VLAN tag's VID field.

EMAC Register 448 Timestamp Control Register (Timestamp_Control)

Register	Offset	R/W	Description	Reset Value
Timestamp_Control	EMAC0_BA+0x0700	R/W	Register 448 (Timestamp Control Register)	0x00002000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					TSENMACADDR	SNAPTYPSEL	
15	14	13	12	11	10	9	8
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL
7	6	5	4	3	2	1	0
Reserved		TSADDREG	TSTRIG	TSUPDT	TSINIT	TSCFUPDT	TSENA

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**TSENMACADDR**	**Enable MAC address for PTP Frame Filtering** When set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP frames when PTP is directly sent over Ethernet.
[17:16]	**SNAPTYPSEL**	**Select PTP packets for Taking Snapshots** These bits along with Bits 15 and 14 decide the set of PTP packet types for which snapshot needs to be taken.
[15]	**TSMSTRENA**	**Enable Snapshot for Messages Relevant to Master** When set, the snapshot is taken only for the messages relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.
[14]	**TSEVNTENA**	**Enable Timestamp Snapshot for Event Messages** When set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When reset, the snapshot is taken for all messages except Announce, Management, and Signaling.
[13]	**TSIPV4ENA**	**Enable Processing of PTP Frames Sent over IPv4-UDP** When set, the MAC receiver processes the PTP packets encapsulated in UDP over IPv4 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv4 packets. This bit is set by default.
[12]	**TSIPV6ENA**	**Enable Processing of PTP Frames Sent Over IPv6-UDP** When set, the MAC receiver processes PTP packets encapsulated in UDP over IPv6 packets. When this bit is clear, the MAC ignores the PTP transported over UDP-IPv6 packets.
[11]	**TSIPENA**	**Enable Processing of PTP over Ethernet Frames** When set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet frames. When this bit is clear, the MAC ignores the PTP over Ethernet packets.
[10]	**TSVER2ENA**	**Enable PTP packet Processing for Version 2 Format** When set, the PTP packets are processed using the 1588 version 2 format. Otherwise, the PTP packets are processed using the version 1 format.

[9]	**TSCTRLSSR**	**Timestamp Digital or Binary Rollover Control** When set, the Timestamp Low register rolls over after 0x3B9A_C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When reset, the rollover value of sub-second register is 0x7FFF_FFFF. The sub-second increment has to be programmed correctly depending on the PTP reference clock frequency and the value of this bit.
[8]	**TSENALL**	**Enable Timestamp for All Frames** When set, the timestamp snapshot is enabled for all frames received by the MAC.
[7:6]	**Reserved**	Reserved.
[5]	**TSADDREG**	**Addend Reg Update** When set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This is cleared when the update is completed. This register bit should be zero before setting it.
[4]	**TSTRIG**	**Timestamp Interrupt Trigger Enable** When set, the timestamp interrupt is generated when the System Time becomes greater than the value written in the Target Time register. This bit is reset after the generation of the Timestamp Trigger Interrupt.
[3]	**TSUPDT**	**Timestamp Update** When set, the system time is updated (added or subtracted) with the value specified in Register 452 (System Time - Seconds Update Register) and Register 453 (System Time - Nanoseconds Update Register). This bit should be read zero before updating it. This bit is reset when the update is completed in hardware. The Timestamp Higher Word register is not updated.
[2]	**TSINIT**	**Timestamp Initialize** When set, the system time is initialized (overwritten) with the value specified in the Register 452 (System Time - Seconds Update Register) and Register 453 (System Time - Nanoseconds Update Register). This bit should be read zero before updating it. This bit is reset when the initialization is complete. The Timestamp Higher Word register can only be initialized.
[1]	**TSCFUPDT**	**Timestamp Fine or Coarse Update** When set, this bit indicates that the system times update should be done using the fine update method. When reset, it indicates the system timestamp update should be done using the Coarse method.
[0]	**TSENA**	**Timestamp Enable** When set, the timestamp is added for the transmit and receive frames. When disabled, timestamp is not added for the transmit and receive frames and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode. On the receive side, the MAC processes the 1588 frames only if this bit is set.

EMAC Register 449 Sub-second Increment Register (Sub_Second_Increment)

Register	Offset	R/W	Description	Reset Value
Sub_Second_Increment	EMAC0_BA+0x0704	R/W	Register 449 (Sub-Second Increment Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
SSINC							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**SSINC**	**Sub-second Increment Value** The value programmed in this field is accumulated every clock cycle with the contents of the sub-second register. For example, when PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time-Nanoseconds register has an accuracy of 1 ns (TSCTRLSSR bit is set). When TSCTRLSSR is clear, the Nanoseconds register has a resolution of ~0.465ns. In this case, you should program a value of 43 (0x2B) that is derived by 20ns/0.465.

EMAC Register 450 System Time - Seconds Register (System_Time_Seconds)

Register	Offset	R/W	Description	Reset Value
System_Time_Seconds	EMAC0_BA+0x0708	R	Register 450 (System Time - Seconds Register)	0x00000000

31	30	29	28	27	26	25	24
TSS							
23	22	21	20	19	18	17	16
TSS							
15	14	13	12	11	10	9	8
TSS							
7	6	5	4	3	2	1	0
TSS							

Bits	Description	
[31:0]	TSS	**Timestamp Second** The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

EMAC Register 451 System Time - Nanoseconds Register (System_Time_Nanoseconds)

Register	Offset	R/W	Description	Reset Value
System_Time_Nanoseconds	EMAC0_BA+0x070c	R	Register 451 (System Time - Nanoseconds Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved	TSSS						
23	22	21	20	19	18	17	16
TSSS							
15	14	13	12	11	10	9	8
TSSS							
7	6	5	4	3	2	1	0
TSSS							

Bits	Description	
[31]	Reserved	Reserved.
[30:0]	TSSS	**Timestamp Sub Seconds** The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When bit 9 (TSCTRLSSR) is set in Register 448 (Timestamp Control Register), each bit represents 1 ns and the maximum value is 0x3B9A_C9FF, after which it rolls-over to zero.

EMAC Register 452 System Time - Seconds Update Register (System_Time_Seconds_Update)

Register	Offset	R/W	Description	Reset Value
System_Time_Seconds_Update	EMAC0_BA+0x0710	R/W	Register 452 (System Time - Seconds Update Register)	0x00000000

31	30	29	28	27	26	25	24
TSS							
23	22	21	20	19	18	17	16
TSS							
15	14	13	12	11	10	9	8
TSS							
7	6	5	4	3	2	1	0
TSS							

Bits	Description	
[31:0]	TSS	**Timestamp Second** The value in this field indicates the time in seconds to be initialized or added to the system time.

EMAC Register 453 System Time - Nanoseconds Update Register (System_Time_Nanoseconds_Update)

Register	Offset	R/W	Description	Reset Value
System_Time_Nanoseconds_Update	EMAC0_BA+0x0714	R/W	Register 453 (System Time - Nanoseconds Update Register)	0x00000000

31	30	29	28	27	26	25	24
ADDSUB	TSSS						
23	22	21	20	19	18	17	16
TSSS							
15	14	13	12	11	10	9	8
TSSS							
7	6	5	4	3	2	1	0
TSSS							

Bits	Description	
[31]	ADDSUB	**Add or subtract time** When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.
[30:0]	TSSS	**Timestamp Sub Second** The value in this field has the sub second representation of time, with an accuracy of 0.46 ns. When bit 9 (TSCTRLSSR) is set in Register 448 (Timestamp Control Register), each bit represents 1 ns and the programmed value should not exceed 0x3B9A_C9FF.

EMAC Register 454 Timestamp Addend Register (Timestamp_Addend)

Register	Offset	R/W	Description	Reset Value
Timestamp_Addend	EMAC0_BA+0x0718	R/W	Register 454 (Timestamp Addend Register)	0x00000000

31	30	29	28	27	26	25	24
TSAR							
23	22	21	20	19	18	17	16
TSAR							
15	14	13	12	11	10	9	8
TSAR							
7	6	5	4	3	2	1	0
TSAR							

Bits	Description	
[31:0]	**TSAR**	**Timestamp Addend Register** This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

EMAC Register 455 Target Time Seconds Register (Target_Time_Seconds)

Register	Offset	R/W	Description	Reset Value
Target_Time_Seconds	EMAC0_BA+0x071c	R/W	Register 455 (Target Time Seconds Register)	0x00000000

31	30	29	28	27	26	25	24
TSTR							
23	22	21	20	19	18	17	16
TSTR							
15	14	13	12	11	10	9	8
TSTR							
7	6	5	4	3	2	1	0
TSTR							

Bits	Description	
[31:0]	TSTR	**Target Time Seconds Register** This register stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, then based on Bits [6:5] of Register 459 (PPS Control Register), the MAC starts or stops the PPS signal output and generates an interrupt (if enabled).

EMAC Register 456 Target Time Nanoseconds Register (Target_Time_Nanoseconds)

Register	Offset	R/W	Description	Reset Value
Target_Time_Nanoseconds	EMAC0_BA+0x0720	R/W	Register 456 (Target Time Nanoseconds Register)	0x00000000

31	30	29	28	27	26	25	24
TRGTBUSY	TTSLO						
23	22	21	20	19	18	17	16
TTSLO							
15	14	13	12	11	10	9	8
TTSLO							
7	6	5	4	3	2	1	0
TTSLO							

Bits	Description	
[31]	**TRGTBUSY**	**Target Time Register Busy (read only)** The MAC sets this bit when the PPSCMD field (Bits [3:0]) in Register 459 (PPS Control Register) is programmed to 010 or 011. Programming the PPSCMD field to 010 or 011, instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers to the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted. This bit is reserved when the Enable Flexible Pulse-Per-Second Output feature is not selected.
[30:0]	**TTSLO**	**Target Timestamp Low Register** This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the both Target Timestamp registers, then based on the TRGTMODSEL0 field (Bits [6:5]) in Register 459 (PPS Control Register), the MAC starts or stops the PPS signal output and generates an interrupt (if enabled). This value should not exceed 0x3B9A_C9FF when TSCTRLSSR is set in the Timestamp control register. The actual start or stop time of the PPS signal output may have an error margin up to one unit of sub-second increment value.

EMAC Register 457 System Time - Higher Word Seconds Register (System_Time_Higher_Word_Seconds)

Register	Offset	R/W	Description	Reset Value
System_Time_Higher_Word_Seconds	EMAC0_BA+0x0724	R/W	Register 457 (System Time - Higher Word Seconds Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TSHWR							
7	6	5	4	3	2	1	0
TSHWR							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	TSHWR	**Timestamp Higher Word Register** This field contains the most significant 16-bits of the timestamp seconds value. The register is directly written to initialize the value. This register is incremented when there is an overflow from the 32-bits of the System Time - Seconds register.

EMAC Register 458 Timestamp Status Register (Timestamp_Status)

Register	Offset	R/W	Description	Reset Value
Timestamp_Status	EMAC0_BA+0x0728	R	Register 458 (Timestamp Status Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				ATSSTN			
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				TSTRGTERR	Reserved	TSTARGT	TSSOVF

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:16]	**ATSSTN**	**Auxiliary Timestamp Snapshot Trigger Identifier** These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list: • Bit 16: Auxiliary trigger 0 • Bit 17: Auxiliary trigger 1 • Bit 18: Auxiliary trigger 2 • Bit 19: Auxiliary trigger 3 The software can read this register to find the triggers that are set when the timestamp is taken.
[15:4]	**Reserved**	Reserved.
[3]	**TSTRGTERR**	**Timestamp Target Time Error** This bit is set when the target time, being programmed in Target Time Registers, is already elapsed. This bit is cleared when read by the application.
[2]	**Reserved**	Reserved.
[1]	**TSTARGT**	**Timestamp Target Time Reached** When set, this bit indicates that the value of system time is greater or equal to the value specified in the Register 455 (Target Time Seconds Register) and Register 456 (Target Time Nanoseconds Register).
[0]	**TSSOVF**	**Timestamp Seconds Overflow** When set, this bit indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 32'hFFFF_FFFF.

EMAC Register 459 PPS Control Register (PPS_Control)

Register	Offset	R/W	Description	Reset Value
PPS_Control	EMAC0_BA+0x072c	R/W	Register 459 (PPS Control Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	TRGTMODSEL0		PPSEN0	PPSCTRL_PPSCMD			

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:5]	**TRGTMODSEL0**	**Target Time Register Mode for PPS0 Output** This field indicates the Target Time registers (register 455 and 456) mode for PPS0 output signal: 00 = Indicates that the Target Time registers are programmed only for generating the interrupt event. 01 = Reserved. 10 = Indicates that the Target Time registers are programmed for generating the interrupt event and starting or stopping the generation of the PPS0 output signal. 11 = Indicates that the Target Time registers are programmed only for starting or stopping the generation of the PPS0 output signal. No interrupt is asserted.
[4]	**PPSEN0**	**Flexible PPS Output Mode Enable** When set low, Bits [3:0] function as PPSCTRL (backward compatible). When set high, Bits [3:0] function as PPSCMD.

[3:0]	**PPSCTRL_PPSCMD**	**PPSCTRL0 or PPSCMD0** PPSCTRL0: PPS0 Output Frequency Control This field controls the frequency of the PPS0 output signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies: 0000 = The binary rollover is 1 Hz, and the digital rollover is 0.5 Hz. 0001 = The binary rollover is 2 Hz, and the digital rollover is 1 Hz. 0010 = The binary rollover is 4 Hz, and the digital rollover is 2 Hz. 0011 = The binary rollover is 8 Hz, and the digital rollover is 4 Hz. 0100 = The binary rollover is 16 Hz, and the digital rollover is 8 Hz. ... 1111 = The binary rollover is 32.768 kHz, and the digital rollover is 16.384 kHz. **Note:** In the binary rollover mode, the PPS output has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example: * When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms. * When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of:. • One clock of 50 percent duty cycle and 537 ms period • Second clock of 463 ms period (268 ms low and 195 ms high) * When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of:. • Three clocks of 50 percent duty cycle and 268 ms period • Fourth clock of 195 ms period (134 ms low and 61 ms high) This behavior is because of the non-linear toggling of bits in the digital rollover mode in Register 451 (System Time - Nanoseconds Register). Flexible PPS0 Output Control Programming these bits with a non-zero value instructs the MAC to initiate an event. Once the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The Software should ensure that these bits are programmed only when they are all-zero. The following list describes the values of PPSCMD0: 0000 = No Command. 0001 = START Single Pulse This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS0 Width Register. 0010 = START Pulse Train This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS0 Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by 'STOP Pulse train at time' or 'STOP Pulse Train immediately' commands. 0011 = Cancel START. This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time. 0100 = STOP Pulse train at time This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010) after the time programmed in the Target Time registers elapses. 0101 = STOP Pulse Train immediately This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD = 0010). 0110 = Cancel STOP Pulse Train. This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command. 0111-1111 = Reserved.

EMAC Register 472 PPS0 Interval Register (PPS0_Interval)

Register	Offset	R/W	Description	Reset Value
PPS0_Interval	EMAC0_BA+0x0760	R/W	Register 472 (PPS0 Interval Register)	0x00000000

31	30	29	28	27	26	25	24
PPSINT							
23	22	21	20	19	18	17	16
PPSINT							
15	14	13	12	11	10	9	8
PPSINT							
7	6	5	4	3	2	1	0
PPSINT							

Bits	Description	
[31:0]	PPSINT	**PPS0 Output Signal Interval** These bits store the interval between the rising edges of PPS0 signal output in terms of units of sub-second increment value. You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20ns), and desired interval between rising edges of PPS0 signal output is 100ns (that is, five units of sub-second increment value), then you should program value 4 (5 -1) in this register.

EMAC Register 473 PPS0 Width Register (PPS0_Width)

Register	Offset	R/W	Description	Reset Value
PPS0_Width	EMAC0_BA+0x0764	R/W	Register 473 (PPS0 Width Register)	0x00000000

31	30	29	28	27	26	25	24
PPSWIDTH							
23	22	21	20	19	18	17	16
PPSWIDTH							
15	14	13	12	11	10	9	8
PPSWIDTH							
7	6	5	4	3	2	1	0
PPSWIDTH							

Bits	Description	
[31:0]	PPSWIDTH	**PPS0 Output Signal Width** These bits store the width between the rising edge and corresponding falling edge of the PPS0 signal output in terms of units of sub-second increment value. You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20ns), and desired width between the rising and corresponding falling edges of PPS0 signal output is 80ns (that is, four units of sub-second increment value), then you should program value 3 (4-1) in this register. **Note:** The value programmed in this register must be lesser than the value programmed in Register 472 (PPS0 Interval Register).

EMAC Register 1000 Bus Mode Register (Bus_Mode)

Register	Offset	R/W	Description	Reset Value
Bus_Mode	EMAC0_BA+0x1000	R/W	Register 1000 (Bus Mode Register)	0x00020100

31	30	29	28	27	26	25	24
Reserved						AAL	PBLx8
23	22	21	20	19	18	17	16
USP	RPBL						FB
15	14	13	12	11	10	9	8
Reserved		PBL					
7	6	5	4	3	2	1	0
ATDS	DSL					Reserved	SWR

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**AAL**	**Address Aligned Beats** When this bit is set high and the FB bit is equal to 1, the AHB interface generates all bursts aligned to the start address LS bits. If the FB bit is equal to 0, the first burst (accessing the data buffer's start address) is not aligned, but subsequent bursts are aligned to the address.
[24]	**PBLx8**	**PBLx8 Mode** When set high, this bit multiplies the programmed PBL value (Bits [22:17] and Bits [13:8]) eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.
[23]	**USP**	**Use Separate PBL** When set high, this bit configures the Rx DMA to use the value configured in Bits [22:17] as PBL. The PBL value in Bits [13:8] is applicable only to the Tx DMA operations. When reset to low, the PBL value in Bits [13:8] is applicable for both DMA engines.
[22:17]	**RPBL**	**Rx DMA PBL** This field indicates the maximum number of beats to be transferred in one Rx DMA transaction. This is the maximum value that is used in a single block Read or Write. The Rx DMA always attempts to burst as specified in the RPBL bit each time it starts a Burst transfer on the host bus. You can program RPBL with values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. This field is valid and applicable only when USP is set high.
[16]	**FB**	**Fixed Burst** This bit controls whether the AHB Master interface performs fixed burst transfers or not. When set, the AHB interface uses only SINGLE, INCR4, INCR8, or INCR16 during start of the normal burst transfers. When reset, the AHB interface uses SINGLE and INCR burst transfer operations.
[15:14]	**Reserved**	Reserved.

[13:8]	**PBL**	**Programmable Burst Length** These bits indicate the maximum number of beats to be transferred in one DMA transaction. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a Burst transfer on the host bus. PBL can be programmed with permissible values of 1, 2, 4, 8, 16, and 32. Any other value results in undefined behavior. When USP is set high, this PBL value is applicable only for Tx DMA transactions. If the number of beats to be transferred is more than 32, then perform the following steps: 1. Set the PBLx8 mode. 2. Set the PBL. For example, if the maximum number of beats to be transferred is 64, then first set PBLx8 to 1 and then set PBL to 8. The PBL values have the following limitation: The maximum number of possible beats (PBL) is limited by the size of the Tx FIFO and Rx FIFO in the MTL layer and the data bus width on the DMA. The FIFO has a constraint that the maximum beat supported is half the depth of the FIFO, except when specified. For different data bus widths and FIFO sizes, the valid PBL range (including x8 mode) is provided in the following list. If the PBL is common for both transmit and receive DMA, the minimum Rx FIFO and Tx FIFO depths must be considered. **Note:** In the half-duplex mode, the valid PBL range specified in the following list is applicable only for Tx FIFO. * 32-Bit Data Bus Width • 128 Bytes FIFO Depth: In the full-duplex mode, the valid PBL range is 16 or less. In the half-duplex mode, the valid PBL range is 8 or less for the 10 or 100 Mbps mode. • 256 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 32 or less. • 512 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 64 or less. • 1 Kbytes FIFO Depth: In the full-duplex mode, the valid PBL range is 128 or less. In the half-duplex mode, the valid PBL range is 128 or less in the 10 or 100 Mbps mode and 64 or less in the 1000 Mbps mode. • 2 Kbytes and Higher FIFO Depth: All PBL values are supported in the full-duplex mode and half-duplex modes. * 64-Bit Data Bus Width • 128 Bytes FIFO Depth: In the full-duplex mode, the valid PBL range is 8 or less. In the half-duplex mode, the valid PBL range is 4 or less for the 10 or 100 Mbps mode. • 256 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 16 or less. • 512 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 32 or less. • 1 Kbytes FIFO Depth: In the full-duplex mode, the valid PBL range is 64 or less. In the half-duplex mode, the valid PBL range is 64 or less in the 10 or 100 Mbps mode and 32 or less in the 1000-Mbps mode. • 2 Kbytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 128 or less. • 4 Kbytes and Higher FIFO Depth: All PBL values are supported in the full-duplex and half-duplex modes. * 128-Bit Data Bus Width • 128 Bytes FIFO Depth: In the full-duplex mode, the valid PBL range is 4 or less. In the half-duplex mode, the valid PBL range is 2 or less for the 10 or 100 Mbps mode. • 256 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 8 or less. • 512 Bytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 16 or less. • 1 Kbytes FIFO Depth: In the full-duplex mode, the valid PBL range is 32 or less. In the half-duplex mode, the valid PBL range is 32 or less in the 10 or 100 Mbps mode and 16 or less in the 1000-Mbps mode. • 2 Kbytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 64 or less.

		• 4 Kbytes FIFO Depth: In the full-duplex mode and the half-duplex (10 or 100 Mbps) modes, the valid PBL range is 128 or less. • 8 Kbytes and Higher FIFO Depth: All PBL values are supported in the full-duplex and half-duplex modes.
[7]	**ATDS**	**Alternate Descriptor Size** When set, the size of the alternate descriptor increases to 32 bytes (8 DWORDS). This is required when the Advanced Timestamp feature or the IPC Full Offload Engine (Type 2) is enabled in the receiver. The enhanced descriptor is not required if the Advanced Timestamp and IPC Full Checksum Offload (Type 2) features are not enabled. In such cases, you can use the 16 bytes descriptor to save 4 bytes of memory. When reset, the descriptor size reverts back to 4 DWORDs (16 bytes). This bit preserves the backward compatibility for the descriptor size. In versions prior to 3.50a, the descriptor size is 16 bytes for both normal and enhanced descriptors. In version 3.50a, descriptor size is increased to 32 bytes because of the Advanced Timestamp and IPC Full Checksum Offload Engine (Type 2) features.
[6:2]	**DSL**	**Descriptor Skip Length** This bit specifies the number of Word, Dword, or Lword (depending on the 32-bit, 64-bit, or 128-bit bus) to skip between two unchained descriptors. The address skipping starts from the end of current descriptor to the start of next descriptor. When the DSL value is equal to zero, the descriptor table is taken as contiguous by the DMA in Ring mode.
[1]	**Reserved**	Reserved.
[0]	**SWR**	**Software Reset** When this bit is set, the MAC DMA Controller resets the logic and all internal registers of the MAC. It is cleared automatically after the reset operation has completed in all of the EMAC clock domains. Before reprogramming any register of the EMAC, you should read a zero (0) value in this bit. **Note:** • The Software reset function is driven only by this bit. • The reset operation is completed only when all resets in all active clock domains are de-asserted. Therefore, it is essential that all the PHY inputs clocks (applicable for the selected PHY interface) are present for the software reset completion.

EMAC Register 1001 Transmit Poll Demand Register (Transmit_Poll_Demand)

Register	Offset	R/W	Description	Reset Value
Transmit_Poll_Demand	EMAC0_BA+0x1004	R/W	Register 1001 (Transmit Poll Demand Register)	0x00000000

31	30	29	28	27	26	25	24
TPD							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TPD							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TPD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TPD							

Bits	Description	
[31:0]	**TPD**	**Transmit Poll Demand** When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 1018 (Current Host Transmit Descriptor Register). If that descriptor is not available (owned by the Host), the transmission returns to the Suspend state and the Bit 2 (TU) of Register 1005 (Status Register) is asserted. If the descriptor is available, the transmission resumes.

EMAC Register 1002 Receive Poll Demand Register (Receive_Poll_Demand)

Register	Offset	R/W	Description	Reset Value
Receive_Poll_Demand	EMAC0_BA+0x1008	R/W	Register 1002 (Receive Poll Demand Register)	0x00000000

31	30	29	28	27	26	25	24
RPD							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RPD							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RPD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RPD							

Bits	Description	
[31:0]	**RPD**	**Receive Poll Demand** When these bits are written with any value, the DMA reads the current descriptor pointed to by Register 1019 (Current Host Receive Descriptor Register). If that descriptor is not available (owned by the Host), the reception returns to the Suspended state and the Bit 7 (RU) of Register 1005 (Status Register) is not asserted. If the descriptor is available, the Rx DMA returns to the active state.

EMAC Register 1003 Receive Descriptor List Address Register (Receive_Descriptor_List_Address)

Register	Offset	R/W	Description	Reset Value
Receive_Descriptor_List_Address	EMAC0_BA+0x100c	R/W	Register 1003 (Receive Descriptor List Address Register)	0x00000000

31	30	29	28	27	26	25	24
RDESLA_64-bit							
23	22	21	20	19	18	17	16
RDESLA_64-bit							
15	14	13	12	11	10	9	8
RDESLA_64-bit							
7	6	5	4	3	2	1	0
RDESLA_64-bit					Reserved		

Bits	Description	
[31:3]	RDESLA_64-bit	**Start of Receive List** This field contains the base address of the first descriptor in the Receive Descriptor list. The LSB bits (2:0) for 64-bit bus width are ignored and are internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).
[2:0]	Reserved	Reserved.

EMAC Register 1004 Transmit Descriptor List Address Register (Transmit_Descriptor_List_Address)

Register	Offset	R/W	Description	Reset Value
Transmit_Descriptor_List_Address	EMAC0_BA+0x1010	R/W	Register 1004 (Transmit Descriptor List Address Register)	0x00000000

31	30	29	28	27	26	25	24
TDESLA_64-bit							
23	22	21	20	19	18	17	16
TDESLA_64-bit							
15	14	13	12	11	10	9	8
TDESLA_64-bit							
7	6	5	4	3	2	1	0
TDESLA_64-bit					Reserved		

Bits	Description	
[31:3]	**TDESLA_64-bit**	**Start of Transmit List** This field contains the base address of the first descriptor in the Transmit Descriptor list. The LSB bits (2:0) for 64-bit bus width are ignored and are internally taken as all-zero by the DMA. Therefore, these LSB bits are read-only (RO).
[2:0]	**Reserved**	Reserved.

EMAC Register 1005 Status Register (Status)

Register	Offset	R/W	Description	Reset Value
Status	EMAC0_BA+0x1014	R/W	Register 1005 (Status Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved		TTI	GPI	Reserved		EB	
23	22	21	20	19	18	17	16
EB	TS			RS			NIS
15	14	13	12	11	10	9	8
AIS	ERI	FBI	Reserved		ETI	RWT	RPS
7	6	5	4	3	2	1	0
RU	RI	UNF	OVF	TJT	TU	TPS	TI

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**TTI**	**Timestamp Trigger Interrupt (read only)** This bit indicates an interrupt event in the Timestamp Generator block of EMAC. The software must read the corresponding registers in the EMAC to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. When this bit is high, the interrupt signal from the EMAC subsystem is high. This bit is applicable only when the IEEE 1588 Timestamp feature is enabled. Otherwise, this bit is reserved.
[28]	**GPI**	**GMAC PMT Interrupt (read only)** This bit indicates an interrupt event in the PMT module of the EMAC. The software must read the PMT Control and Status Register in the MAC to get the exact cause of interrupt and clear its source to reset this bit to 1'b0. The interrupt signal from the EMAC subsystem is high when this bit is high. This bit is applicable only when the Power Management feature is enabled. Otherwise, this bit is reserved.
[27:26]	**Reserved**	Reserved.
[25:23]	**EB**	**Error Bits (read only)** This field indicates the type of error that caused a Bus Error, for example, error response on the AHB interface. This field is valid only when Bit 13 (FBI) is set. This field does not generate an interrupt. 000 = Error during Rx DMA Write Data Transfer. 011 = Error during Tx DMA Read Data Transfer. 100 = Error during Rx DMA Descriptor Write Access. 101 = Error during Tx DMA Descriptor Write Access. 110 = Error during Rx DMA Descriptor Read Access. 111 = Error during Tx DMA Descriptor Read Access. **Note:** 001 and 010 are reserved.

[22:20]	**TS**	**Transmit Process State (read only)** This field indicates the Transmit DMA FSM state. This field does not generate an interrupt. 3'b000 = Stopped; Reset or Stop Transmit Command issued. 3'b001 = Running; Fetching Transmit Transfer Descriptor. 3'b010 = Running; Waiting for status. 3'b011 = Running; Reading Data from host memory buffer and queuing it to transmit buffer (Tx FIFO). 3'b100 = TIME_STAMP write state. 3'b101 = Reserved for future use. 3'b110 = Suspended; Transmit Descriptor Unavailable or Transmit Buffer Underflow. 3'b111 = Running; Closing Transmit Descriptor.
[19:17]	**RS**	**Received Process State (read only)** This field indicates the Receive DMA FSM state. This field does not generate an interrupt. 3'b000 = Stopped: Reset or Stop Receive Command issued. 3'b001 = Running: Fetching Receive Transfer Descriptor. 3'b010 = Reserved for future use. 3'b011 = Running: Waiting for receive packet. 3'b100 = Suspended: Receive Descriptor Unavailable. 3'b101 = Running: Closing Receive Descriptor. 3'b110 = TIME_STAMP write state. 3'b111 = Running: Transferring the receive packet data from receive buffer to host memory.
[16]	**NIS**	**Normal Interrupt Summary** Normal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register 1007 (Interrupt Enable Register): • Register 1005[0]: Transmit Interrupt • Register 1005[2]: Transmit Buffer Unavailable • Register 1005[6]: Receive Interrupt • Register 1005[14]: Early Receive Interrupt Only unmasked bits (interrupts for which interrupt enable is set in Register 1007) affect the Normal Interrupt Summary bit. This is a sticky bit and must be cleared (by writing 1 to this bit) each time a corresponding bit, which causes NIS to be set, is cleared.
[15]	**AIS**	**Abnormal Interrupt Summary** Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in Register 1007 (Interrupt Enable Register): • Register 1005[1]: Transmit Process Stopped • Register 1005[3]: Transmit Jabber Timeout • Register 1005[4]: Receive FIFO Overflow • Register 1005[5]: Transmit Underflow • Register 1005[7]: Receive Buffer Unavailable • Register 1005[8]: Receive Process Stopped • Register 1005[9]: Receive Watchdog Timeout • Register 1005[10]: Early Transmit Interrupt • Register 1005[13]: Fatal Bus Error Only unmasked bits affect the Abnormal Interrupt Summary bit. This is a sticky bit and must be cleared each time a corresponding bit, which causes AIS to be set, is cleared.

[14]	**ERI**	**Early Receive Interrupt** This bit indicates that the DMA filled the first data buffer of the packet. This bit is cleared when the software writes 1 to this bit or Bit 6 (RI) of this register is set (whichever occurs earlier).
[13]	**FBI**	**Fatal Bus Error Interrupt** This bit indicates that a bus error occurred, as described in Bits [25:23]. When this bit is set, the corresponding DMA engine disables all of its bus accesses.
[12:11]	**Reserved**	Reserved.
[10]	**ETI**	**Early Transmit Interrupt** This bit indicates that the frame to be transmitted is fully transferred to the MTL Transmit FIFO.
[9]	**RWT**	**Receive Watchdog Timeout** When set, this bit indicates that the Receive Watchdog Timer expired while receiving the current frame and the current frame is truncated after the watchdog timeout.
[8]	**RPS**	**Receive Process Stopped** This bit is asserted when the Receive Process enters the Stopped state.
[7]	**RU**	**Receive Buffer Unavailable** This bit indicates that the host owns the Next Descriptor in the Receive List and the DMA cannot acquire it. The Receive Process is suspended. To resume processing Receive descriptors, the host should change the ownership of the descriptor and issue a Receive Poll Demand command. If no Receive Poll Demand is issued, the Receive Process resumes when the next recognized incoming frame is received. This bit is set only when the previous Receive Descriptor is owned by the DMA.
[6]	**RI**	**Receive Interrupt** This bit indicates that the frame reception is complete. When reception is complete, the Bit 31 of RDES1 (Disable Interrupt on Completion) is reset in the last Descriptor, and the specific frame status information is updated in the descriptor. The reception remains in the Running state.
[5]	**UNF**	**Transmit Underflow** This bit indicates that the Transmit Buffer had an Underflow during frame transmission. Transmission is suspended and an Underflow Error TDES0[1] is set.
[4]	**OVF**	**Receive Overflow** This bit indicates that the Receive Buffer had an Overflow during frame reception. If the partial frame is transferred to the application, the overflow status is set in RDES0[11].
[3]	**TJT**	**Transmit Jabber Timeout** This bit indicates that the Transmit Jabber Timer expired, which happens when the frame size exceeds 2,048 (10,240 bytes when the Jumbo frame is enabled). When the Jabber Timeout occurs, the transmission process is aborted and placed in the Stopped state. This causes the Transmit Jabber Timeout TDES0[14] flag to assert.
[2]	**TU**	**Transmit Buffer Unavailable** This bit indicates that the host owns the Next Descriptor in the Transmit List and the DMA cannot acquire it. Transmission is suspended. Bits [22:20] explain the Transmit Process state transitions. To resume processing Transmit descriptors, the host should change the ownership of the descriptor by setting TDES0[31] and then issue a Transmit Poll Demand command.
[1]	**TPS**	**Transmit Process Stopped** This bit is set when the transmission is stopped.
[0]	**TI**	**Transmit Interrupt** This bit indicates that the frame transmission is complete. When transmission is complete, Bit 31 (OWN) of TDES0 is reset, and the specific frame status information is updated in the descriptor.

EMAC Register 1006 Operation Mode Register (Operation_Mode)

Register	Offset	R/W	Description	Reset Value
Operation_Mode	EMAC0_BA+0x1018	R/W	Register 1006 (Operation Mode Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved					DT	RSF	DFF
23	22	21	20	19	18	17	16
Reserved		TSF	FTF	Reserved			TTC
15	14	13	12	11	10	9	8
TTC		ST	RFD		RFA		EFC
7	6	5	4	3	2	1	0
FEF	FUF	DGF	RTC		OSF	SR	Reserved

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26]	**DT**	**Disable Dropping of TCP/IP Checksum Error Frames** When this bit is set, the MAC does not drop the frames which only have errors detected by the Receive Checksum Offload engine. Such frames do not have any errors (including FCS error) in the Ethernet frame received by the MAC but have errors only in the encapsulated payload. When this bit is reset, all error frames are dropped if the FEF bit is reset. If the IPC Full Checksum Offload Engine (Type 2) is disabled, this bit is reserved (RO with value 1'b0).
[25]	**RSF**	**Receive Store and Forward** When this bit is set, the MTL reads a frame from the Rx FIFO only after the complete frame has been written to it, ignoring the RTC bits. When this bit is reset, the Rx FIFO operates in the cut-through mode, subject to the threshold specified by the RTC bits
[24]	**DFF**	**Disable Flushing of Received Frames** When this bit is set, the Rx DMA does not flush any frames because of the unavailability of receive descriptors or buffers as it does normally when this bit is reset.
[23:22]	**Reserved**	Reserved.
[21]	**TSF**	**Transmit Store and Forward** When this bit is set, transmission starts when a full frame resides in the MTL Transmit FIFO. When this bit is set, the TTC values specified in Bits [16:14] are ignored. This bit should be changed only when the transmission is stopped.
[20]	**FTF**	**Flush Transmit FIFO** When this bit is set, the transmit FIFO controller logic is reset to its default values and thus all data in the Tx FIFO is lost or flushed. This bit is cleared internally when the flushing operation is completed. The Operation Mode register should not be written to until this bit is cleared. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt frame transmission. **Note:** The flush operation is complete only when the Tx FIFO is emptied of its contents and all the pending Transmit Status of the transmitted frames are accepted by the host. To complete this flush operation, the PHY transmit clock is required to be active.
[19:17]	**Reserved**	Reserved.

[16:14]	**TTC**	**Transmit Threshold Control** These bits control the threshold level of the MTL Transmit FIFO. Transmission starts when the frame size within the MTL Transmit FIFO is larger than the threshold. In addition, full frames with a length less than the threshold are also transmitted. These bits are used only when Bit 21 (TSF) is reset. 000 = 64 bytes. 001 = 128 bytes. 010 = 192 bytes. 011 = 256 bytes. 100 = 40 bytes. 101 = 32 bytes. 110 = 24 bytes. 111 = 16 bytes.
[13]	**ST**	**Start or Stop Transmission Command** When this bit is set, transmission is placed in the Running state, and the DMA checks the Transmit List at the current position for a frame to be transmitted. Descriptor acquisition is attempted either from the current position in the list, which is the Transmit List Base Address set by Register 1004 (Transmit Descriptor List Address Register), or from the position retained when transmission was stopped previously. If the DMA does not own the current descriptor, transmission enters the Suspended state and Bit 2 (Transmit Buffer Unavailable) of Register 1005 (Status Register) is set. The Start Transmission command is effective only when transmission is stopped. If the command is issued before setting Register 1004 (Transmit Descriptor List Address Register), then the DMA behavior is unpredictable. When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current frame. The Next Descriptor position in the Transmit List is saved, and it becomes the current position when transmission is restarted. To change the list address, you need to program Register 1004 (Transmit Descriptor List Address Register) with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current frame is complete or the transmission is in the Suspended state.
[12:11]	**RFD**	**Threshold for Deactivating Flow Control** These bits control the threshold (Fill-level of Rx FIFO) at which the flow control is de-asserted after activation. 00 = Full minus 1 Kbytes, that is, FULL - 1 Kbytes. 01 = Full minus 2 Kbytes, that is, FULL - 2 Kbytes. 10 = Full minus 3 Kbytes, that is, FULL - 3 Kbytes. 11 = Full minus 4 Kbytes, that is, FULL - 4 Kbytes. The de-assertion is effective only after flow control is asserted. If the Rx FIFO is 8 Kbytes or more, an additional bit (RFD_2) is used for more threshold levels as described in Bit 22. These bits are reserved and read-only when the Rx FIFO depth is less than 4 Kbytes. **Note:** For proper flow control, the value programmed in the "RFD_2, RFD" fields should be equal to or more than the value programmed in the "RFA_2, RFA" fields.
[10:9]	**RFA**	**Threshold for Activating Flow Control** These bits control the threshold (Fill level of Rx FIFO) at which the flow control is activated. 00 = Full minus 1 Kbytes, that is, FULL - 1 Kbytes. 01 = Full minus 2 Kbytes, that is, FULL - 2 Kbytes. 10 = Full minus 3 Kbytes, that is, FULL - 3 Kbytes. 11 = Full minus 4 Kbytes, that is, FULL - 4 Kbytes. These values are applicable only to Rx FIFOs of 4 Kbytes or more and when Bit 8 (EFC) is set high. If the Rx FIFO is 8 Kbytes or more, an additional Bit (RFA_2) is used for more threshold levels as described in Bit 23. These bits are reserved and read-only when the depth of Rx FIFO is less than 4 Kbytes. **Note:** When FIFO size is exactly 4 Kbytes, although the EMAC allows you to program the value of these bits to 11, the software should not program these bits to 2'b11. The value 2'b11 means flow control on FIFO empty condition.

[8]	**EFC**	**Enable HW Flow Control** When this bit is set, the flow control signal operation based on the fill-level of Rx FIFO is enabled. When reset, the flow control operation is disabled. This bit is not used (reserved and always reset) when the Rx FIFO is less than 4 Kbytes.
[7]	**FEF**	**Forward Error Frames** When this bit is reset, the Rx FIFO drops frames with error status (CRC error, collision error, GMII_ER, giant frame, watchdog timeout, or overflow). However, if the start byte (write) pointer of a frame is already transferred to the read controller side (in Threshold mode), then the frame is not dropped. When the FEF bit is set, all frames except runt error frames are forwarded to the DMA. If the Bit 25 (RSF) is set and the Rx FIFO overflows when a partial frame is written, then the frame is dropped irrespective of the FEF bit setting. However, if the Bit 25 (RSF) is reset and the Rx FIFO overflows when a partial frame is written, then a partial frame may be forwarded to the DMA.
[6]	**FUF**	**Forward Undersized Good Frames** When set, the Rx FIFO forwards Undersized frames (frames with no Error and length less than 64 bytes) including pad-bytes and CRC. When reset, the Rx FIFO drops all frames of less than 64 bytes, unless a frame is already transferred because of the lower value of Receive Threshold, for example, RTC = 01.
[5]	**DGF**	**Drop Giant Frames** When set, the MAC drops the received giant frames in the Rx FIFO, that is, frames that are larger than the computed giant frame limit. When reset, the MAC does not drop the giant frames in the Rx FIFO.
[4:3]	**RTC**	**Receive Threshold Control** These two bits control the threshold level of the MTL Receive FIFO. Transfer (request) to DMA starts when the frame size within the MTL Receive FIFO is larger than the threshold. In addition, full frames with length less than the threshold are transferred automatically. These bits are valid only when the RSF bit is zero, and are ignored when the RSF bit is set to 1. 00 = 64 bytes. 01 = 32 bytes. 10 = 96 bytes. 11 = 128 bytes.
[2]	**OSF**	**Operate on Second Frame** When this bit is set, it instructs the DMA to process the second frame of the Transmit data even before the status for the first frame is obtained.
[1]	**SR**	**Start or Stop Receive** When this bit is set, the Receive process is placed in the Running state. The DMA attempts to acquire the descriptor from the Receive list and processes the incoming frames. The descriptor acquisition is attempted from the current position in the list, which is the address set by Register 1003 (Receive Descriptor List Address Register) or the position retained when the Receive process was previously stopped. If the DMA does not own the descriptor, reception is suspended and Bit 7 (Receive Buffer Unavailable) of Register 1005 (Status Register) is set. The Start Receive command is effective only when the reception has stopped. If the command is issued before setting Register 1003 (Receive Descriptor List Address Register), the DMA behavior is unpredictable. When this bit is cleared, the Rx DMA operation is stopped after the transfer of the current frame. The next descriptor position in the Receive list is saved and becomes the current position after the Receive process is restarted. The Stop Receive command is effective only when the Receive process is in either the Running (waiting for receive packet) or in the Suspended state.
[0]	**Reserved**	Reserved.

EMAC Register 1007 Interrupt Enable Register (Interrupt_Enable)

Register	Offset	R/W	Description	Reset Value
Interrupt_Enable	EMAC0_BA+0x101c	R/W	Register 1007 (Interrupt Enable Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							NIE
15	14	13	12	11	10	9	8
AIE	ERE	FBE	Reserved		ETE	RWE	RSE
7	6	5	4	3	2	1	0
RUE	RIE	UNE	OVE	TJE	TUE	TSE	TIE

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**NIE**	**Normal Interrupt Summary Enable** When this bit is set, normal interrupt summary is enabled. When this bit is reset, normal interrupt summary is disabled. This bit enables the following interrupts in Register 1005 (Status Register): • Register 1005[0]: Transmit Interrupt • Register 1005[2]: Transmit Buffer Unavailable • Register 1005[6]: Receive Interrupt • Register 1005[14]: Early Receive Interrupt
[15]	**AIE**	**Abnormal Interrupt Summary Enable** When this bit is set, abnormal interrupt summary is enabled. When this bit is reset, the abnormal interrupt summary is disabled. This bit enables the following interrupts in Register 1005 (Status Register): • Register 1005[1]: Transmit Process Stopped • Register 1005[3]: Transmit Jabber Timeout • Register 1005[4]: Receive Overflow • Register 1005[5]: Transmit Underflow • Register 1005[7]: Receive Buffer Unavailable • Register 1005[8]: Receive Process Stopped • Register 1005[9]: Receive Watchdog Timeout • Register 1005[10]: Early Transmit Interrupt • Register 1005[13]: Fatal Bus Error
[14]	**ERE**	**Early Receive Interrupt Enable** When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Early Receive Interrupt is enabled. When this bit is reset, the Early Receive Interrupt is disabled.
[13]	**FBE**	**Fatal Bus Error Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Fatal Bus Error Interrupt is enabled. When this bit is reset, the Fatal Bus Error Enable Interrupt is disabled.

[12:11]	**Reserved**	Reserved.
[10]	**ETE**	**Early Transmit Interrupt Enable** When this bit is set with an Abnormal Interrupt Summary Enable (Bit 15), the Early Transmit Interrupt is enabled. When this bit is reset, the Early Transmit Interrupt is disabled.
[9]	**RWE**	**Receive Watchdog Timeout Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Watchdog Timeout Interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout Interrupt is disabled.
[8]	**RSE**	**Receive Stopped Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped Interrupt is disabled.
[7]	**RUE**	**Receive Buffer Unavailable Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Buffer Unavailable Interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable Interrupt is disabled.
[6]	**RIE**	**Receive Interrupt Enable** When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.
[5]	**UNE**	**Underflow Interrupt Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Underflow Interrupt is enabled. When this bit is reset, the Underflow Interrupt is disabled.
[4]	**OVE**	**Overflow Interrupt Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Receive Overflow Interrupt is enabled. When this bit is reset, the Overflow Interrupt is disabled.
[3]	**TJE**	**Transmit Jabber Timeout Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmit Jabber Timeout Interrupt is enabled. When this bit is reset, the Transmit Jabber Timeout Interrupt is disabled.
[2]	**TUE**	**Transmit Buffer Unavailable Enable** When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Buffer Unavailable Interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable Interrupt is disabled.
[1]	**TSE**	**Transmit Stopped Enable** When this bit is set with Abnormal Interrupt Summary Enable (Bit 15), the Transmission Stopped Interrupt is enabled. When this bit is reset, the Transmission Stopped Interrupt is disabled.
[0]	**TIE**	**Transmit Interrupt Enable** When this bit is set with Normal Interrupt Summary Enable (Bit 16), the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.

EMAC Register 1008 Missed Frame and Buffer Overflow Counter Register (Missed_Frame_And_Buffer_Overflow_Counter)

Register	Offset	R/W	Description	Reset Value
Missed_Frame_And_Buffer_Overflow_Counter	EMAC0_BA+0x1020	R	Register 1008 (Missed Frame and Buffer Overflow Counter Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved			OVFCNTOVF	OVFFRMCNT			
23	22	21	20	19	18	17	16
OVFFRMCNT							MISCNTOVF
15	14	13	12	11	10	9	8
MISFRMCNT							
7	6	5	4	3	2	1	0
MISFRMCNT							

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28]	**OVFCNTOVF**	**Overflow Bit for FIFO Overflow Counter** This bit is set every time the Overflow Frame Counter (Bits [27:17]) overflows, that is, the Rx FIFO overflows with the overflow frame counter at maximum value. In such a scenario, the overflow frame counter is reset to all-zeros and this bit indicates that the rollover happened.
[27:17]	**OVFFRMCNT**	**Overflow Frame Counter** This field indicates the number of frames missed by the application. This counter is incremented each time the MTL FIFO overflows. The counter is cleared when this register is read with mci_be_i[2] at 1'b1.
[16]	**MISCNTOVF**	**Overflow Bit for Missed Frame Counter** This bit is set every time Missed Frame Counter (Bits [15:0]) overflows, that is, the DMA discards an incoming frame because of the Host Receive Buffer being unavailable with the missed frame counter at maximum value. In such a scenario, the Missed frame counter is reset to all-zeros and this bit indicates that the rollover happened.
[15:0]	**MISFRMCNT**	**Missed Frame Counter** This field indicates the number of frames missed by the controller because of the Host Receive Buffer being unavailable. This counter is incremented each time the DMA discards an incoming frame. The counter is cleared when this register is read with mci_be_i[0] at 1'b1.

EMAC Register 1009 Receive Interrupt Watchdog Timer Register (Receive_Interrupt_Watchdog_Timer)

Register	Offset	R/W	Description	Reset Value
Receive_Interrupt_Watchdog_Timer	EMAC0_BA+0x1024	R/W	Register 1009 (Receive Interrupt Watchdog Timer Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RIWT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**RIWT**	**RI Watchdog Timer Count** This bit indicates the number of system clock cycles multiplied by 256 for which the watchdog timer is set. The watchdog timer gets triggered with the programmed value after the Rx DMA completes the transfer of a frame for which the RI status bit is not set because of the setting in the corresponding descriptor RDES1[31]. When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per RDES1[31] of any received frame.

EMAC Register 1011 AHB Status Register (AHB_Status)

Register	Offset	R/W	Description	Reset Value
AHB_Status	EMAC0_BA+0x102c	R	Register 1011 (AHB Status Register)	0x00000000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**AXIRDSTS**	**AXWHSTS**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**AXWHSTS**	**AHB Master Write Channel Status** When high, it indicates that AHB Master's write channel is active and transferring data.

EMAC Register 1018 Current Host Transmit Descriptor Register (Current_Host_Transmit_Descriptor)

Register	Offset	R/W	Description	Reset Value
Current_Host_Transmit_Descriptor	EMAC0_BA+0x1048	R	Register 1018 (Current Host Transmit Descriptor Register)	0x00000000

31	30	29	28	27	26	25	24
CURTDESAPTR							
23	22	21	20	19	18	17	16
CURTDESAPTR							
15	14	13	12	11	10	9	8
CURTDESAPTR							
7	6	5	4	3	2	1	0
CURTDESAPTR							

Bits	Description	
[31:0]	CURTDESAPTR	**Host Transmit Descriptor Address Pointer** Cleared on Reset. Pointer updated by the DMA during operation.

EMAC Register 1019 Current Host Receive Descriptor Register (Current_Host_Receive_Descriptor)

Register	Offset	R/W	Description	Reset Value
Current_Host_Receive_Descriptor	EMAC0_BA+0x104c	R	Register 1019 (Current Host Receive Descriptor Register)	0x00000000

31	30	29	28	27	26	25	24
CURRDESAPTR							
23	22	21	20	19	18	17	16
CURRDESAPTR							
15	14	13	12	11	10	9	8
CURRDESAPTR							
7	6	5	4	3	2	1	0
CURRDESAPTR							

Bits	Description	
[31:0]	CURRDESAPTR	**Host Receive Descriptor Address Pointer** Cleared on Reset. Pointer updated by the DMA during operation.

EMAC Register 1020 Current Host Transmit Buffer Address Register (Current_Host_Transmit_Buffer_Address)

Register	Offset	R/W	Description	Reset Value
Current_Host_Transmit_Buffer_Address	EMAC0_BA+0x1050	R	Register 1020 (Current Host Transmit Buffer Address Register)	0x00000000

31	30	29	28	27	26	25	24
CURTBUFAPTR							
23	22	21	20	19	18	17	16
CURTBUFAPTR							
15	14	13	12	11	10	9	8
CURTBUFAPTR							
7	6	5	4	3	2	1	0
CURTBUFAPTR							

Bits	Description	
[31:0]	**CURTBUFAPTR**	**Host Transmit Buffer Address Pointer** Cleared on Reset. Pointer updated by the DMA during operation.

EMAC Register 1021 Current Host Receive Buffer Address Register (Current_Host_Receive_Buffer_Address)

Register	Offset	R/W	Description	Reset Value
Current_Host_Receive_Buffer_Address	EMAC0_BA+0x1054	R	Register 1021 (Current Host Receive Buffer Address Register)	0x00000000

31	30	29	28	27	26	25	24
CURRBUFAPTR							
23	22	21	20	19	18	17	16
CURRBUFAPTR							
15	14	13	12	11	10	9	8
CURRBUFAPTR							
7	6	5	4	3	2	1	0
CURRBUFAPTR							

Bits	Description	
[31:0]	CURRBUFAPTR	**Host Receive Buffer Address Pointer** Cleared on Reset. Pointer updated by the DMA during operation.

EMAC Register 1022 HW Feature Register (HW_Feature)

Register	Offset	R/W	Description	Reset Value
HW_Feature	EMAC0_BA+0x1058	R	Register 1022 (HW Feature Register)	0x4F0D6525

31	30	29	28	27	26	25	24
Reserved	ACTPHYIF			SAVLANINS	FLEXIPPSEN	INTTSEN	ENHDESSEL
23	22	21	20	19	18	17	16
TXCHCNT		RXCHCNT		RXFIFOSIZE	RXTYP2COE	RXTYP1COE	TXCOESEL
15	14	13	12	11	10	9	8
Reserved	EEESEL	TSVER2SEL	TSVER1SEL	MMCSEL	MGKSEL	RWKSEL	SMASEL
7	6	5	4	3	2	1	0
L3L4FLTREN	PCSSEL	ADDMACADRSEL	HASHSEL	EXTHASHEN	HDSEL	Reserved	MIISEL

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**ACTPHYIF**	**Active or Selected PHY interface** When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion 0100 = RMII. All Others = Reserved.
[27]	**SAVLANINS**	Source Address or VLAN Insertion
[26]	**FLEXIPPSEN**	Flexible Pulse-Per-Second Output
[25]	**INTTSEN**	Timestamping with Internal System Time
[24]	**ENHDESSEL**	Alternate (Enhanced Descriptor)
[23:22]	**TXCHCNT**	Number of additional Tx channels
[21:20]	**RXCHCNT**	Number of additional Rx channels
[19]	**RXFIFOSIZE**	Rx FIFO > 2,048 Bytes
[18]	**RXTYP2COE**	IP Checksum Offload (Type 2) in Rx
[17]	**RXTYP1COE**	IP Checksum Offload (Type 1) in Rx
[16]	**TXCOESEL**	Checksum Offload in Tx
[15]	**Reserved**	Reserved.
[14]	**EEESEL**	Energy Efficient Ethernet
[13]	**TSVER2SEL**	IEEE 1588-2008 Advanced Timestamp
[12]	**TSVER1SEL**	Only IEEE 1588-2002 Timestamp
[11]	**MMCSEL**	RMON Module
[10]	**MGKSEL**	PMT Magic Packet

[9]	**RWKSEL**	PMT Remote Wakeup
[8]	**SMASEL**	SMA (MDIO) Interface
[7]	**L3L4FLTREN**	Layer 3 and Layer 4 Filter Feature
[6]	**PCSSEL**	PCS registers (TBI, SGMII, or RTBI PHY interface)
[5]	**ADDMACADRSEL**	Multiple MAC Address Registers
[4]	**HASHSEL**	HASH Filter
[3]	**EXTHASHEN**	Expanded DA Hash Filter
[2]	**HDSEL**	Half-Duplex support
[1]	**Reserved**	Reserved.
[0]	**MIISEL**	10 or 100 Mbps support

6.25 Smart Card Host Interface (SC)

6.25.1 Overview

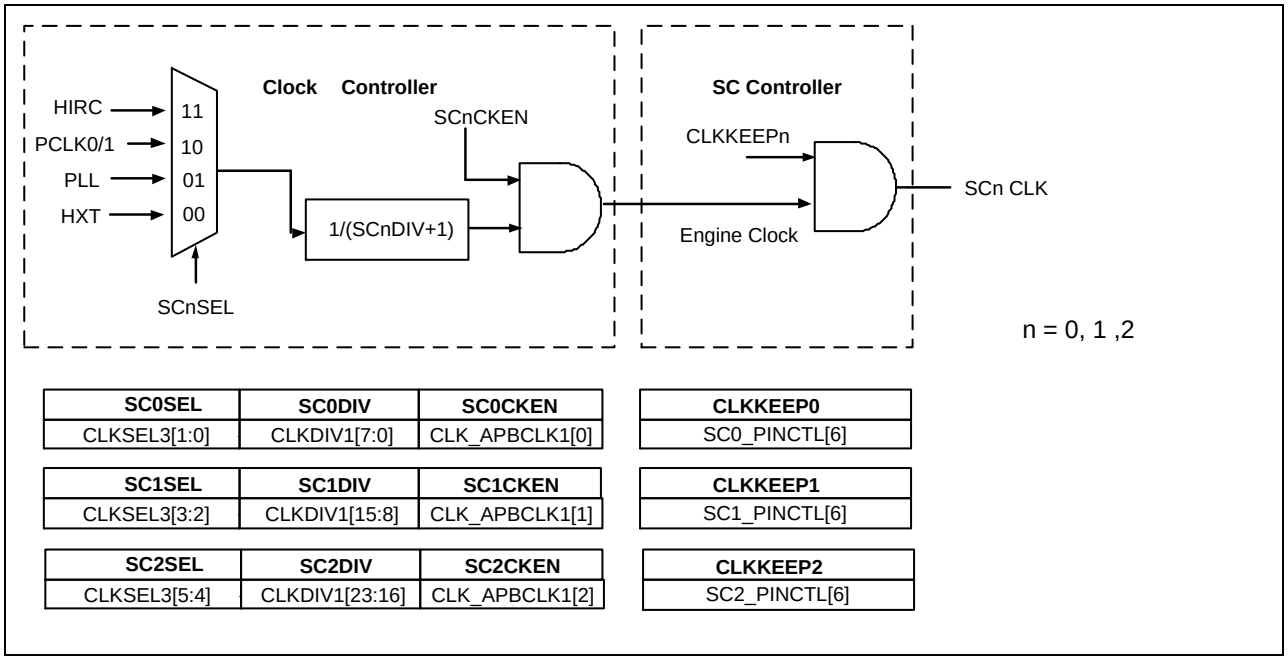
The Smart Card Interface controller (SC controller) is based on ISO/IEC 7816-3 standard and fully compliant with PC/SC Specifications. It also provides status of card insertion/removal. It can also be set as UART mode to communicate with other device.

6.25.2 Features

- ISO 7816-3 T = 0, T = 1 compliant
- EMV2000 compliant
- Three ISO 7816-3 ports
- Separates receive/transmit 4 bytes' entry FIFO for data payloads
- Programmable transmission clock frequency
- Programmable receiver buffer trigger level
- Programmable guard time selection (11 ETU ~ 267 ETU)
- One 24-bit timer and two 8-bit timers for Answer to Request (ATR) and waiting times processing
- Supports auto direct / inverse convention function
- Supports transmitter and receiver error retry and error number limiting function
- Supports hardware activation sequence process, and the time between PWR on and CLK start is configurable
- Supports hardware warm reset sequence process
- Supports hardware deactivation sequence process
- Supports hardware auto deactivation sequence when detected the card removal
- Supports UART mode
  - Full duplex, asynchronous communications
  - Separates receiving / transmitting 4 bytes' entry FIFO for data payloads
  - Supports programmable baud rate generator
  - Supports programmable receiver buffer trigger level
  - Programmable transmitting data delay time between the last stop bit leaving the TX-FIFO and the de-assertion by setting EGT (SCn_EGT[7:0])
  - Programmable even, odd or no parity bit generation and detection
  - Programmable stop bit, 1- or 2- stop bit generation

6.25.3 Block Diagram

The SC clock control and block diagram are shown in SC Clock Control Diagram (8-bit Pre-scale Counter in Clock Controller) and SC Controller Block Diagram. The SC controller is completely asynchronous design with two clock domains, PCLK and engine clock. Note that the PCLK should be higher than or equal to the frequency of engine clock.



SC0SEL	SC0DIV	SC0CKEN	CLKKEEP0
CLKSEL3[1:0]	CLKDIV1[7:0]	CLK_APBCLK1[0]	SC0_PINCTL[6]
SC1SEL	**SC1DIV**	**SC1CKEN**	**CLKKEEP1**
CLKSEL3[3:2]	CLKDIV1[15:8]	CLK_APBCLK1[1]	SC1_PINCTL[6]
SC2SEL	**SC2DIV**	**SC2CKEN**	**CLKKEEP2**
CLKSEL3[5:4]	CLKDIV1[23:16]	CLK_APBCLK1[2]	SC2_PINCTL[6]

Figure 6.25-1 SC Clock Control Diagram (8-bit Pre-scale Counter in Clock Controller)

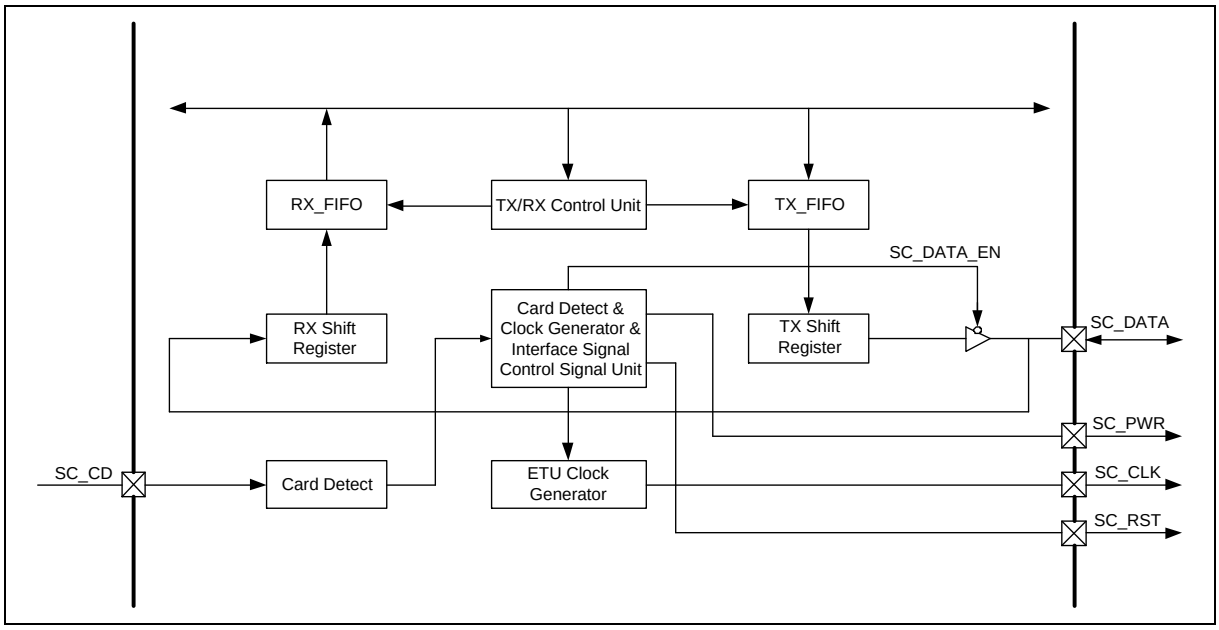


Figure 6.25-2 SC Controller Block Diagram

6.25.4 Basic Configuration

SC Host Controller Pin description is shown in:

Pin	Type	Description
SCn_DATA	Bi-direction	SC Host Controller DATA
SCn_CD	Input	SC Host Controller Card Detect
SCn_PWR	Output	SC Host Controller Power ON/OFF
SCn_CLK	Output	SC Host Controller Clock
SCn_RST	Output	SC Host Controller Reset

Table 6.25-1 SC Host Controller Pin Description

UART Mode Pin description is shown in UART Pin Description:

Pin	Type	Description
SCn_DATA	Input	UART Receive Data
SCn_CLK	Output	UART Transmit Data

Table 6.25-2 UART Pin Description

6.25.4.1 SC0 Basic Configuration

- Clock Source Configuration
  - Select the source of SC0 peripheral clock on SC0SEL (CLK_CLKSEL3[1:0]).
  - Select the clock divider number of SC0 peripheral clock on SC0DIV (CLK_CLKDIV1[7:0]).
  - Enable SC0 peripheral clock in SC0CKEN (CLK_APBCLK1[0]).
- Reset Configuration
  - Reset SC0 controller in SC0RST (SYS_IPRST2[0]).

6.25.4.2 SC1 Basic Configuration

- Clock Source Configuration
  - Select the source of SC1 peripheral clock on SC1SEL (CLK_CLKSEL3[3:2]).
  - Select the clock divider number of SC1 peripheral clock on SC1DIV (CLK_CLKDIV1[15:8]).
  - Enable SC1 peripheral clock in SC1CKEN (CLK_APBCLK1[1]).
- Reset Configuration
  - Reset SC1 controller in SC1RST (SYS_IPRST2[1]).

6.25.4.3 SC2 Basic Configuration

- Clock Source Configuration
  - Select the source of SC2 peripheral clock on SC2SEL (CLK_CLKSEL3[5:4]).
  - Select the clock divider number of SC2 peripheral clock on SC2DIV (CLK_CLKDIV1[23:16]).
  - Enable SC2 peripheral clock in SC2CKEN (CLK_APBCLK1[2]).

- Reset Configuration
  - Reset SC2 controller in SC2RST (SYS_IPRST2[2]).

6.25.5 Functional Description

Basically, the smart card interface acts as a half-duplex asynchronous communication port and its data format is composed of ten consecutive bits which is shown in SC Data Character.

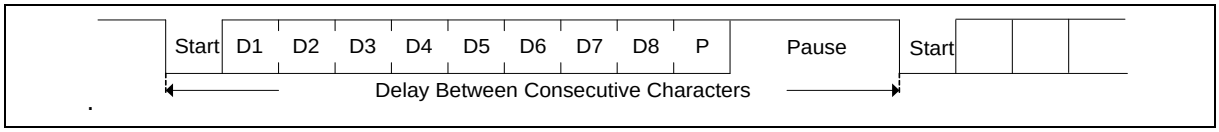


Figure 6.25-3 SC Data Character

The Smart Card Interface controller supports hardware activation, warm reset and deactivation sequence. The activation, warm reset and deactivation sequence are shown as follows.

6.25.5.1 Smart Card Pin Configuration

The Smart Card Interface pin status can be observed by polling following registers.

1. SCn_RST is a output pin. Its status can be observed by RSTSTS (SCn_PINCTL[18]). Programming RSTEN (SCn_PINCTL[1]) '0' or '1' can drive this output pin low or high.
2. SCn_PWR is a output pin. Its status can be observed by PWRSTS (SCn_PINCTL[17]). Programming PWREN (SCn_PINCTL[0]) to '0' or '1' can drive this output pin low or high. PWRINV (SCn_PINCTL[11]) can inverse the SCn_PWR output. User must select PWRINV (SCn_PINCTL[11]) before smart card is enabled by SCEN (SCn_CTL[0]).
3. SCn_DATA is a bidirectional pin, DATASTS (SCn_PINCTL[16]) shows the pin status when SC is receiving data. Programming SCDATA (SCn_PINCTL[9]) to '0' or '1' can drive this pin output low or high.
4. SCn_CLK is a output pin. It outputs Smart card clock SCn CLK. Programming CLKKEEP (SCn_PINCTL[6]) '0' or '1' to disable or enable this pin. Programming CSTOPLV (SCn_PINCTL[5]) can determine the SCn_CLK is stopped at high or low when this pin is disable.
5. SCn_CD(Card Detect Pin) state represent the status of the card is inserted or not. SCn_CD pin status can be observed by CDPINSTS (SCn_STATUS[13]). SCn_CD related function can be set by CDLV (SCn_CTL[26]), CDDBSEL (SCn_CTL[25:24]), CDIF (SCn_INTSTS[7]), CDIEN (SCn_INTEN[7]).
6. CDLV (SCn_CTL[26]) determines what kind of pin level change represents the card insertion. CDDBSEL (SCn_CTL[25:24]) determines the de-bounce cycles. When the card status CINSERT (SCn_STATUS[12]) or CREMOVE (SCn_STATUS[11]) is detected, CDIF will set to 1. If CDIEN is enable, SC will deliver an interrupt to CPU when CDIF is set to 1. Card Detect Pin is recommend setting before enable SC.

6.25.5.2 Activation, Warm Reset and Deactivation Sequence

Activation

The activation sequence is shown in Figure 6.25-4:

1. Set SCn_RST to low by programming RSTEN (SCn_PINCTL[1]) to '0', and wait SYNC (SCn_PINCTL[31]) is cleared to 0.
2. Set SCn_PWR at high level by programming PWREN (SCn_PINCTL[0]) to '1' and SCn_DATA at high level (reception mode) by programming SCDATA (SCn_PINCTL[9]) to '1', and wait SYNC (SCn_PINCTL[31]) is cleared to 0.
3. Enable SCn_CLK clock by programming CLKKEEP (SCn_PINCTL[6]) to '1', and wait SYNC

(SCn_PINCTL[31]) is cleared to 0.

4. De-assert SCn_RST to high by programming RSTEN (SCn_PINCTL[1]) to '1', and wait SYNC (SCn_PINCTL[31]) is cleared to 0.

The activation sequence can be controlled in two ways. The procedure is shown as follows:

- Software Timing Control:

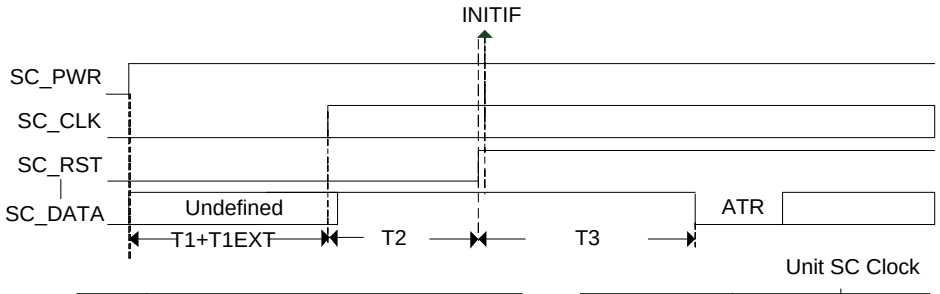
  Set SCn_PINCTL and SCn_TMRCTLx (x = 0, 1, 2) to process the activation sequence. SCn_PWR, SCn_CLK, SCn_RST and SCn_DATA pin state can be programmed by SCn_PINCTL. The programming method is shown in activation sequence. The activation sequence timing can be controlled by setting SCn_TMRCTLx (x = 0, 1, 2). This programming procedure provides user with a flexible timing setting for activation sequence.

- Hardware Timing Control:

  Set ACTEN (SCn_ALTCTL[3]) to '1' and the interface will perform the activation sequence by hardware. The SCn_PWR to SCn_CLK start (T1) and SCn_CLK_start to SCn_RST assert (T2) can be selected by programming INITSEL (SCn_ALTCTL[9:8]). The SCn_PWR to SCn_CLK length can be configure by setting T1EXT (SCn_ACTCTL[4:0]). This programming procedure provides user with a simple setting for activation sequence. During the hardware activation, RX receive is disabled and cannot receive data.

The following is activation control sequence in hardware activation mode:

1. Set activation timing by setting INITSEL (SCn_ALTCTL[9:8]).
2. Timer0 can be selected by setting TMRSEL (SCn_CTL[14:13]) is 11.
3. Set operation mode OPMODE (SCn_TMRCTL0[27:24]) to 0011 and give an Answer to Request (ATR) value by setting CNT (SCn_TMRCTL0[23:0]) register.
4. When hardware de-asserts SCn_RST to high, hardware will generator an interrupt INITIF (SCn_INTSTS[8]) to CPU at the same time if INITIEN (SCn_INTEN[8]) is 1.
5. If the Timer0 decreases the counter to "0" (started from SCn_RST de-assert) and the card does not response ATR before that time, hardware will generate an interrupt flag TMR0IF (SCn_INTSTS[3]).



Time	Comment
T1	SC_PWR to SC_CLK Start
T1EXT	Configurable cycles for T1
T2	SC_CLK Start to SC_RST Assert
T3	SC_CLK Start to ATR Appear

INITSEL (SC_ALTCTL[9:8])	T1	T2
00	83.5	491
01	133	537
10	165	569
11	165	42060

Note : The values are measured by chip I / O pin and the real value will depend on system design

Figure 6.25-4 SC Activation Sequence

Warm Reset

The warm reset sequence is shown in Figure 6.25-5 :

1. Set SCn_RST to low by programming RSTEN (SCn_PINCTL[1]) to '0', and wait SYNC (SCn_PINCTL[31]) is cleared to 0.
2. Set SCn_DATA to high by programming SCDATA (SCn_PINCTL[9]) to '1', and wait SYNC (SCn_PINCTL[31]) is cleared to 0.
3. Set SCn_RST to high by programming RSTEN (SCn_PINCTL[1]) to '1', and wait SYNC (SCn_PINCTL[31]) is cleared to 0.

The warm reset sequence can be controlled in two ways. The procedure is shown as follows.

- Software Timing Control:

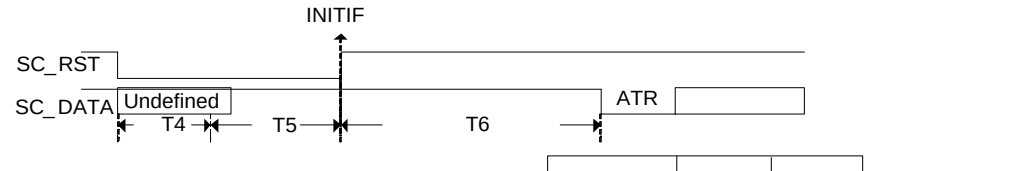
  Set SCn_PINCTL and SCn_TMRCTLx (x = 0, 1, 2) to process the warm reset sequence. The SCn_RST and SCn_DATA pin state can be programmed by SCn_PINCTL. The warm reset sequence timing can be controlled by setting SCn_TMRCTLx (x = 0, 1, 2). This programming procedure provides user with a flexible timing setting for warm reset sequence.

- Hardware Timing Control:

  Set WARSTEN (SCn_ALTCTL[4]) to '1' and the interface will perform the warm reset sequence by hardware. The SCn_RST to SCn_DATA reception mode (T4) and SCn_DATA reception mode to SCn_RST assert (T5) can be selected by programming INITSEL (SCn_ALTCTL[9:8]). This programming procedure provides user with a simple setting for warm reset sequence. During the hardware warm reset, RX receive is disabled and cannot receive data.

The following is the warm reset control sequence by hardware:

1. Set warm reset timing by setting INITSEL (SCn_ALTCTL[9:8]).
2. Select Timer0 by setting TMRSEL (SCn_CTL[14:13]) to 11.
3. Set operation mode OPMODE (SCn_TMRCTL0[27:24]) to 0011 and give an Answer to Request (ATR) value by setting CNT (SCn_TMRCTL0[23:0]) register.
4. Set CNTEN0 (SCn_ALTCTL[5]) and WARSTEN (SCn_ALTCTL[4]) to start counting.
5. When hardware de-asserts SCn_RST to high, hardware will generate an interrupt INITIF (SCn_INTSTS[8]) to CPU at the same time if INITIEN (SCn_INTEN[8]) is 1.
6. If the Timer0 decreases the counter to '0' (start from SCn_RST) and the card does not response ATR before that time, hardware will generate an interrupt flag TMR0IF (SCn_INTSTS[3]).



Time	Comment
T4	SC_RST to SC_DATA Reception Mode
T5	SC_DATA Reception Mode to SC_RST Assert
T6	SC_CLK Start to ATR Appear

INITSEL (SC_ ALTCTL[9:8])	T4	T5
00	81	483
01	129	531
10	161	563
11	161	42106

Unit: SC Clock

Note: This value is measured by chip IO pin and the real value will depend on system design

Figure 6.25-5 SC Warm Reset Sequence

Deactivation

The deactivation sequence is shown in Figure 6.25-6.

1. Set SCn_RST to low by programming RSTEN (SCn_PINCTL[1]) to '0', wait SYNC (SCn_PINCTL[31]) is cleared to 0.
2. Stop SCn_CLK by programming CLKKEEP (SCn_PINCTL[6]) to '0', wait SYNC (SCn_PINCTL[31]) is cleared to 0.
3. Set SCn_DATA to low by programming SCDATA (SCn_PINCTL[9]) to '0', wait SYNC (SCn_PINCTL[31]) is cleared to 0.
4. Deactivate SCn_PWR by programming PWREN (SCn_PINCTL[0]) to '0', wait SYNC (SCn_PINCTL[31]) is cleared to 0.

The deactivation sequence can be controlled in two ways. The procedure is shown as follows.

- Software Timing Control:

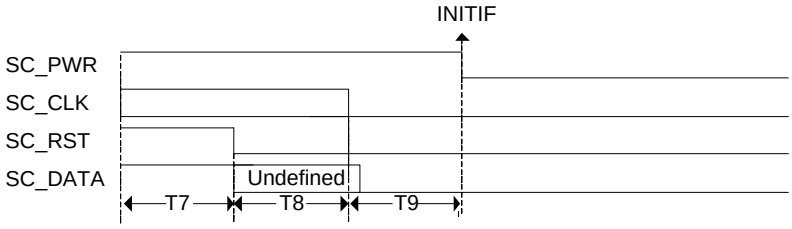
  Set SCn_PINCTL and SCn_TMRCTL0 to process the deactivation sequence. SCn_PWR, SCn_CLK, SCn_RST and SCn_DATA pin state can be programmed by SCn_PINCTL. The deactivation sequence timing can be controlled by setting SCn_TMRCTL0. This programming procedure provides user with a flexible timing setting for deactivation sequence.

- Hardware Timing Control:

  DACTEN (SCn_ALTCTL[2]) to '1' and the interface will perform the deactivation sequence by hardware. The Deactivation Trigger to SCn_RST low (T7), SMC_RST low to SCn_CLK (T8) and stop SCn_CLK to stop SCn_PWR (T9) time can be selected by programming INITSEL (SCn_ALTCTL[9:8]). This programming procedure provides user with a simple setting for deactivation sequence.

When hardware de-asserts SCn_PWR to low, the SC controller will generate an interrupt INITIF (SCn_INTSTS[8]) to CPU at the same time if INITIEN (SCn_INTEN[8]) is 1.

The SC controller also supports auto deactivation sequence when the card removal detection is enabled by setting ADACEN (SCn_ALTCTL[11]).



Time	Comment
T7	Deactivation Trigger to SC_RST Low
T8	SMC_RST Low to Stop SC_CLK
T9	Stop SC_CLK to Stop SC_PWR

INITSEL (SC_ALTCTL[9:8])	T7	T8	T9
00	97	83	87
01	145	131	135
10	177	163	167
11	177	163	167

Unit: SC Clock

Figure 6.25-6 SC Deactivation Sequence

6.25.5.3 Basic Operation Flow

Basic operation flow of smartcard can be referenced from ISO 7816-3 & EMV.

The Program Sequence Flow is shown as follows:

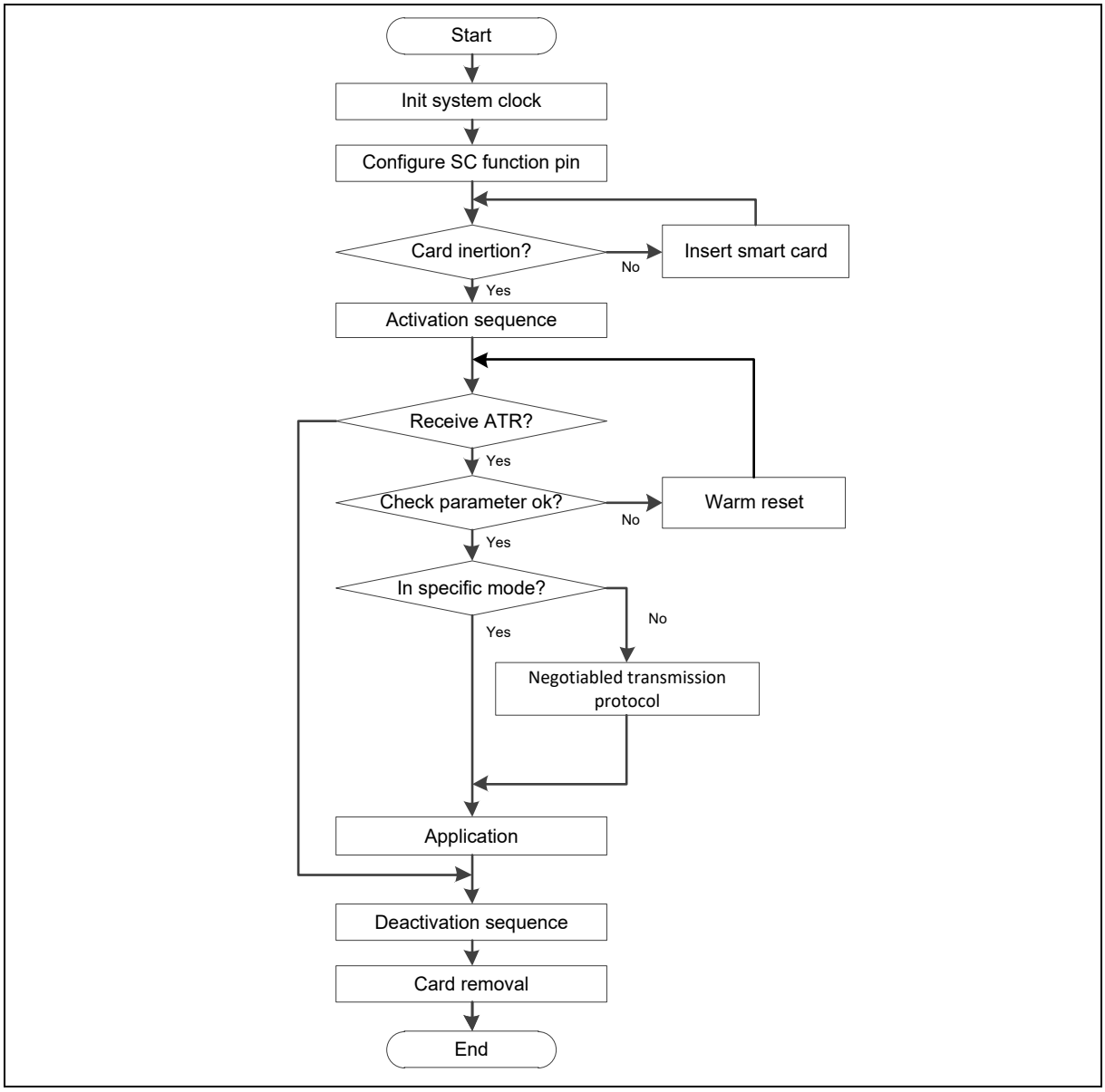


Figure 6.25-7 Basic Operation Flow

6.25.5.4 Initial Character TS

According to ISO 7816-3, the initial character TS has two possible patterns shown in Figure 6.25-8. If the TS pattern is 1100_0000, it is inverse convention. When decoded by inverse convention, the conveyed byte is equal to 0x3F. If the TS pattern is 1101_1100, it is direct convention. When decoded by direct convention, the conveyed byte is equal to 0x3B. User can set AUTOCEN (SCn_CTL[3]) and then the operating convention will be decided by hardware. User can also set the CONSEL (SCn_CTL[5:4]) register (set to '00' or '11') to change the operating convention after SC received TS of answer to request (ATR).

If auto convention function is enabled by setting AUTOCEN (SCn_CTL[3]) register, the setting step must be done before Answer to Request (ATR) state and the received first data must be 0x3B or 0x3F. After hardware received first data and stored it at SCn_DAT, the hardware will decided the convention and change the CONSEL (SCn_CTL[5:4]) register automatically. If the received first data is neither 0x3B nor 0x3F, ACERRIF (SCn_INTSTS[10] Auto Convention Error Interrupt Status Flag) will be set and the hardware will generate an interrupt to CPU if ACERRIEN (SCn_INTEN[10]) is 1.

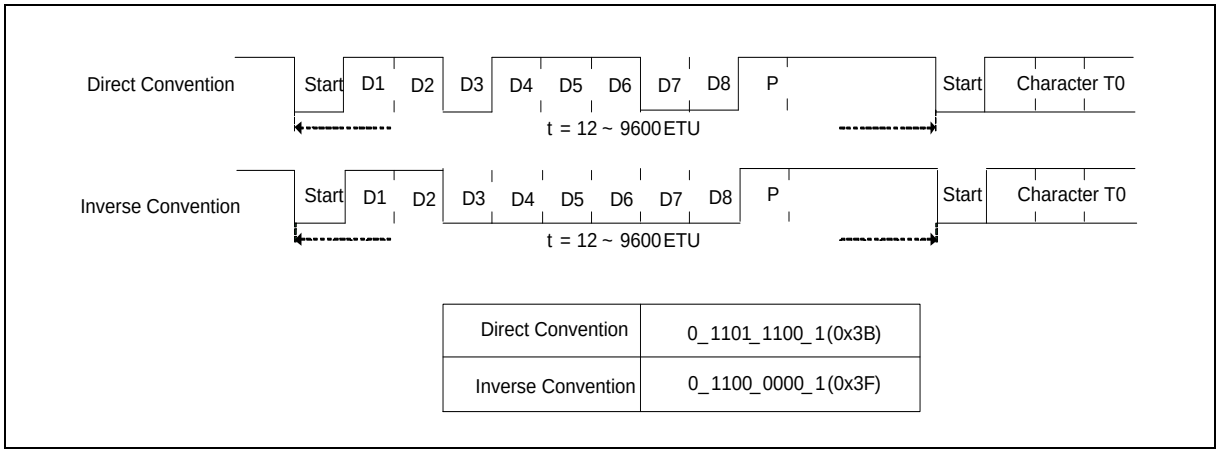


Figure 6.25-8 Initial Character TS

6.25.5.5 Transfer Data Flow and Data Buffer Status

After setting initial sequence, SC can start transferring data which format is corresponding to ISO 7816-3. Set ETURDIV(SCn_ETUCTL[11:0]) to 273 to make ETU(Element Timing Unit) meet ISO 7816-3. Writing data to SCn_DAT, SC will send out an 8-bit data to smart card. Reading data from SCn_DAT, SC will return an 8-bit received data from smart card.

Data buffer status show in SCn_STATUS. TXPOINT (SCn_STATUS[26:24]) and RXPOINT (SCn_STATUS[18:16]) represent how many data in transmit buffer and received buffer. TXEMPTY(SCn_STATUS[9]), TXFULL (SCn_STATUS[10]), TXOV (SCn_STATUS[8]), RXEMPTY (SCn_STATUS[1]), RXFULL (SCn_STATUS[2]), RXOV (SCn_STATUS[0]), represent the transmitted/received buffer status is full, empty, or overflow. SC even can generate interrupt for the transmit buffer empty situation by setting TBEIEN (SCn_INTEN[1]) bit. After interrupt status flag TBEIF (SCn_INTSTS[1]) is generated, user can decide to transfer data or not by polling these flag and it only can be cleared automatically when write data to SCn_DAT again.

TX and RX can be disabled seperately by setting TXOFF (SCn_CTL[2]) and RXOFF (SCn_CTL[1]). TXACT (SCn_STATUS[31]) and RXACT (SCn_STATUS[23]) represent the TX transfer/RX transfer is active or not.

6.25.5.6 Receiver Buffer Time-out

The time-out down counter resets and starts counting whenever the RX buffer received a new data. Once the counter decrease to 1 and no new data is received or CPU does not read data by reading SCn_DAT, a receiver time-out flag RXTOIF (SCn_INTSTS[9]) will be set, and hardware will generate an interrupt to CPU when RXTOIEN (SCn_INTEN[9]) is enabled.

6.25.5.7 Error Signal and Character Repetition

According to ISO 7816-3 T=0 mode description, as shown in Figure 6.25-9, if the receiver receives a wrong parity bit, it will pull the SCn_DAT to low by 1.5 bit period to inform the transmitter parity error. Then the transmitter will retransmit the character. The SC interface controller supports hardware error detection function in receiver and supports hardware re-transmit function in transmitter.

User can enable re-transmit function by setting TXRTYEN (SCn_CTL[23]). User can also define the retry (re-transmit) number limitation in TXRTY (SCn_CTL[22:20]). If the re-transmit number is between 1 and TXRTY, TXRERR (SCn_STATUS[29]) flag will be set by hardware. The re-transmit number is up to TXRTY +1 and if the re-transmit number is equal to TXRTY +1, TXOVERR (SCn_STATUS[30]) flag will be set by hardware and if TERRIEN (SCn_INTEN[2]) is enabled, SC controller will generate a transfer error interrupt to CPU, and TERRIF (SCn_INTSTS[2]) flag will also be set.

User can also enable re-received function by setting RXRTYEN (SCn_CTL[19]).The received retry number limitation is defined in RXRTY (SCn_CTL[18:16]). If the re-received number is between 1 and RXRTY, RXRERR (SCn_STATUS[21]) flag will be set by hardware. The receiver retry number is up to RXRTY +1, if the number of received errors by receiver is equal to RXRTY +1, receiver will receive this error data to buffer and RXOVERR (SCn_STATUS[22]) flag will be set by hardware and if TERRIEN (SCn_INTEN[2]) is enabled, SC controller will generate a transfer error interrupt to CPU, and TERRIF (SC_INTSTS[2]) flag will also be set.

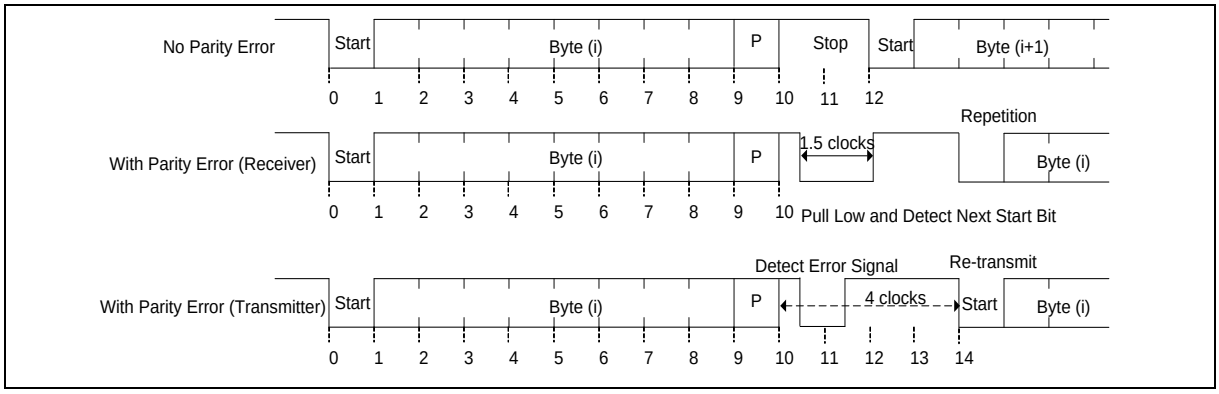


Figure 6.25-9 SC Error Signal

6.25.5.8 Internal Timer Operation Mode

The smart card interface includes a 24-bit time-out counter and two 8-bit time-out counters. These counters help the controller in processing different real-time interval. Each counter can be set to start counting once the trigger enable bit (CNTENx in SCn_ALTCTL[7:5], x = 0, 1, 2) has been written or a START bit has been detected.

The following is the programming flow:

1. Enable counter by setting TMRSEL (SCn_CTL[14:13]) to 11.
2. Select operation mode OPMODE (SCn_TMRCTLx[27:24], x = 0, 1, 2).
3. Give a count value CNT for Timer0, Timer1 and Timer2 by setting CNT0 (SCn_TMRCTL0[23:0]), CNT1 (SCn_TMRCTL1[7:0]) and CNT2 (SCn_TMRCTL2[7:0] register).
4. Set CNTEN0 (SCn_ALTCTL [5]), CNTEN1 (SCn_ALTCTL [6]) or CNTEN2 (SCn_ALTCTL [7]) to enable timer. ACTSTS0 (SCn_ALTCTL[13]), ACTSTS1 (SCn_ALTCTL[14]) and ACTSTS2 (SCn_ALTCTL[15]) represent the status that timer is enable or not.
5. Wait until the counting condition is satisfied, and the timer starts counting.
6. When internal timer counter satisfied interrupt conditions in different modes and TMR0IEN (SCn_INTEN[3]), TMR1IEN (SCn_INTEN[4]), TMR2IEN (SCn_INTEN[5]) are enabled, SC will generate a interrupt to CPU.

The SCn_TMRCTL0, SCn_TMRCTL1 and SCn_TMRCTL2 timer operation mode are listed in Table 6.25-3.

Note1: Only Timer0 (SCn_TMRCTL0 register) supports mode 0011.

Note2: START bit can only be detected when Tx or Rx is idle or finish the last transmission.

OPMODE (SCn_TMRCTLx[27:24]), X = 0, 1, 2)	Operation Mode Description	
0000	The down counter is started when CNTENx (SCn_ALTCTL[7:5]) enabled and ended when counter time-out. The time-out counter value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.	
	Start	Start counting when CNTENx (SCn_ALTCTL[7:5]) enabled.
	End	When the down counter equals 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and clear CNTENx (SCn_ALTCTL[7:5]) automatically.
0001	The down counter is started when the first START bit (reception or transmission) detected and ended when counter time-out. It takes 2 ETU to detect first START bit after writing data to Tx or receiving data from Rx. The time-out counter value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.	
	Start	Start counting when the first START bit (reception or transmission) detected after CNTENx (SCn_ALTCTL[7:5]) set to 1.
	End	When the down counter equals 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and clear CNTENx (SCn_ALTCTL[7:5]) automatically.
0010	The down counter is started when the first START bit (reception) detected and ended when counter time-out. It takes 2 ETU to detect first START bit after receiving data from Rx. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.	
	Start	Start counting when the first START bit (reception) detected bit after CNTENx (SCn_ALTCTL[7:5]) set to 1.
	End	When the down counter equals 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and clear CNTENx (SCn_ALTCTL[7:5]) automatically.
0011	The down counter is only used for hardware activation, warm reset sequence to measure ATR timing. The timing starts when SCn_RST de-assertion and ends when ATR response received or time-out. If the counter decreases to 0 before ATR response received, hardware will set TMR0IF (SCn_INTSTS[3]) and generate an interrupt to CPU if TMR0IEN (SCn_INTEN[3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0]) +1.	
	Start	Start counting when SCn_RST de-assertion after CNTEN0 (SCn_ALTCTL[5]) set to 1. It is only used for hardware activation, warm reset mode.
	End	When the down counter equals 0 before ATR response received, hardware will set TMR0IF and clear CNTEN0 (SCn_ALTCTL[5]) automatically. When ATR received and down counter does not equal to 0, hardware will clear CNTEN0 (SCn_ALTCTL[5]) automatically.
0100	Start	Start down counter counting when CNTENx (SCn_ALTCTL[7:5]) enabled.
	Recount & reload	When ACTSTSx (SCn_ALTCTL[15:13]) is 1, user can change CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL0[7:0], SCn_TMRCTL0[7:0]) value at any time. It will reload the last value which is filled into the CNT(SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) before the counter count to 0. Only when the down counter equals 0, counter reload the CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) value and start to recount.
	Interrupt	If the counter decreases to 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and generate an interrupt to CPU if TMRxIEN (SCn_INTEN[5:3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.
	End	The down counter stopped when use clears CNTENx (SCn_ALTCTL[7:5]) bit.

0101	Start	The down counter is started when the first START bit (reception or transmission) detected after CNTENx (SCn_ALTCTL[7:5]) set to 1. It takes 2 ETU to detect START bit after writing data to Tx or receiving data from Rx.
	Reload	When ACTSTSx (SCn_ALTCTL[15:13]) is 1, user can change CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL0[7:0], SCn_TMRCTL0[7:0]) value at any time. It will reload the last value which is filled into the CNT(SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) before the counter count to 0. Only when the down counter equals 0, counter will reload the CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) value.
	Recount	After down counter reloads the CNT value, timer counter starts to recount only when the next START bit is detected.
	Interrupt	If the counter decreases to 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and generate an interrupt to CPU if TMRxIEN (SCn_INTEN[5:3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.
	End	The down counter stopped when user clears CNTENx (SCn_ALTCTL[7:5]) bit.
0110	Start	The down counter is started when the first START bit (reception) detected after CNTENx (SCn_ALTCTL[7:5]) set to 1. It takes 2 ETU to detect START bit after writing data to Tx or receiving data from Rx.
	Reload	When ACTSTSx (SCn_ALTCTL[15:13]) is 1, user can change CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL0[7:0], SCn_TMRCTL0[7:0]) value at any time. It will reload the last value which is filled into the CNT(SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) before the counter counts to 0. Only when the down counter equals 0, counter reload the CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) value.
	Recount	After the down counter reloads the CNT value, timer counter starts to recount only when the next START bit is detected.
	Interrupt	If the counter decreases to 0, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and generate an interrupt to CPU if TMRxIEN (SCn_INTEN[5:3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0])+1.
	End	The down counter stopped when user clears CNTENx (SCn_ALTCTL[7:5]) bit.
0111	Start	The down counter is started when the first START bit (reception or transmission) detected after CNTENx (SCn_ALTCTL[7:5]) set to 1. It takes 2 ETU to detect START bit after writing data to Tx or receiving data from Rx.
	Reload &recount	Only when the next START bit is detected, counter will reload the new value of CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) and recount.
	Interrupt	If the counter decreases to 0 before the next START bit detected, hardware will set TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]) and generate an interrupt to CPU if TMRxIEN (SCn_INTEN[5:3]) enabled. The time-out value will be CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) +1.
	End	The down counter stopped when user clears CNTENx (SCn_ALTCTL[7:5]) bit.
1111	Start	The down counter starts counting when user sets CNTENx (SCn_ALTCTL[7:5]) bit and it will count to time-out.
	Reload &recount	Only when the next START bit is detected, counter will reload the new value of CNT (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0], SCn_TMRCTL2[7:0]) and recount.
	Interrupt	If the counter decreases to 0 before the next START bit detected, hardware will generate time-out interrupt flag TMR0IF, TMR1IF, TMR2IF (SCn_INTSTS[5:3]). The time-out value will be CNTx (SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0],

		SCn_TMRCTL2[7:0]) +1.
	End	The down counter stopped when user clears CNTENx (SCn_ALTCTL[7:5]) bit.

Table 6.25-3 Timer0/Timer1/Timer2 Operation Mode

6.25.5.9 Block Guard Time and Extra Guard Time

Block guard time means the minimum interval between the leading edges of two consecutive characters between different transfer directions. This field indicates the counter for the bit length of block guard time. According to ISO7816-3, in T = 0 mode, user must fill 15 (real block guard time = 16.5) to this field; in T = 1 mode, user must fill 21 (real block guard time = 22.5) to it.

In transmit direction, the smart card sends data to smart card host controller, first. After the period is greater than BGT (SCn_CTL[12:8]), the smart card host controller begin to send the data.

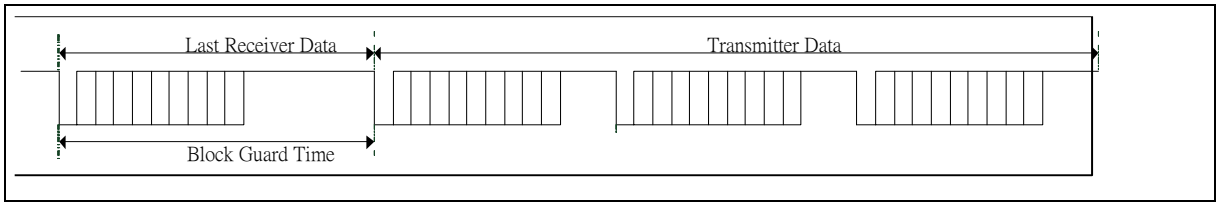


Figure 6.25-10 Transmit Direction Block Guard Time Operation

In receive direction, the smart card host controller sends data to smart card, first. If the smart card responses data to smart card host controller at the time which is less than BGT (SCn_CTL[12:8]), the block guard time interrupt BGTIF (SC_INTSTS[6]) is generated when RXBGTEN (SCn_ALTCTL[12]) and BGTIEN (SCn_INTEN[6]) is enabled.

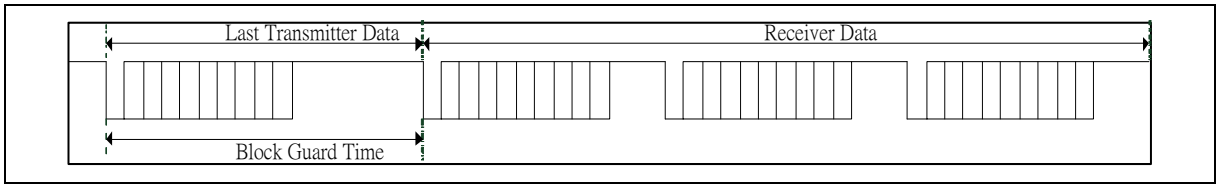


Figure 6.25-11 Receive Direction Block Guard Time Operation

Extra Guard Time is EGT (SCn_EGT[7:0]), it only affects the data transmitted by smart card interface, the format is shown as Figure 6.25-12Figure 6.25-11.

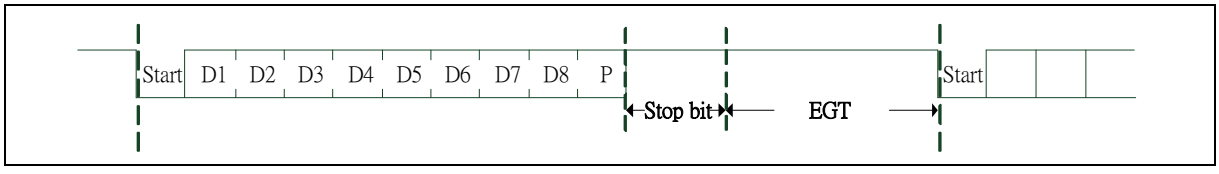


Figure 6.25-12 Extra Guard Time Operation

6.25.5.10 UART Mode

When the UARTEN (SCn_UARTCTL[0]) bit is set, the Smart Card Interface controller can also be used as basic UART function. The following is the program example for UART mode.

Programming example:

1. Set UARTEN (SCn_UARTCTL[0]) bit to enter UART mode.
2. Do user reset by setting RXRST (SCn_ALTCTL[1]) and TXRST(SCn_ALTCTL[0]) bit to ensure that all state machine return idle state.
3. Fill “0” to CONSEL (SCn_CTL[5:4]) and AUTOCEN (SCn_CTL[3]) field. In UART mode, those fields must be “0”.
4. Select the UART baud rate by setting ETURDIV (SCn_ETUCTL[11:0]) fields. For example, if smartcard module clock is 12 MHz and target baud rate is 115200 bps, ETURDIV should fill with ((12000000 / 115200) - 1).
5. Select the data format include data length (by setting WLS (SCn_UARTCTL[5:4]), parity format (by setting OPE (SCn_UARTCTL[7]) and PBOFF (SCn_UARTCTL[6])) and stop bit length (by setting NSB (SCn_CTL[15] or EGT (SCn_EGT[7:0])).
6. Select the receiver buffer number trigger level by setting RXTRGLV (SCn_CTL[7:6]) field and select the receiver buffer time-out interval by setting RFTM (SCn_RXTOUT[8:0]) field.
7. Write the SCn_DAT (SCn_DAT[7:0]) (TX) register or read the SCn_DAT (SCn_DAT[7:0]) (RX) register can perform UART function.

6.25.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SC Base Address: **SCn_BA = 0x4009_0000 + (0x1000 * n)** **n=0,1,2**				
SC_DAT	SCn_BA+0x00	R/W	SC Receive/Transmit Holding Buffer Register	0xXXXX_XXXX
SC_CTL	SCn_BA+0x04	R/W	SC Control Register	0x0000_0000
SC_ALTCTL	SCn_BA+0x08	R/W	SC Alternate Control Register	0x0000_0000
SC_EGT	SCn_BA+0x0C	R/W	SC Extra Guard Time Register	0x0000_0000
SC_RXTOUT	SCn_BA+0x10	R/W	SC Receive Buffer Time-out Counter Register	0x0000_0000
SC_ETUCTL	SCn_BA+0x14	R/W	SC Element Time Unit Control Register	0x0000_0173
SC_INTEN	SCn_BA+0x18	R/W	SC Interrupt Enable Control Register	0x0000_0000
SC_INTSTS	SCn_BA+0x1C	R/W	SC Interrupt Status Register	0x0000_0002
SC_STATUS	SCn_BA+0x20	R/W	SC Transfer Status Register	0x0000_X202
SC_PINCTL	SCn_BA+0x24	R/W	SC Pin Control State Register	0x0000_0000
SC_TMRCTL0	SCn_BA+0x28	R/W	SC Internal Timer0 Control Register	0x0000_0000
SC_TMRCTL1	SCn_BA+0x2C	R/W	SC Internal Timer1 Control Register	0x0000_0000
SC_TMRCTL2	SCn_BA+0x30	R/W	SC Internal Timer2 Control Register	0x0000_0000
SC_UARTCTL	SCn_BA+0x34	R/W	SC UART Mode Control Register	0x0000_0000
SC_ACTCTL	SCn_BA+0x4C	R/W	SC Activation Control Register	0x0000_0000

6.25.7 Register Description

SC Receive/Transmit Holding Buffer Register (SC_DAT)

Register	Offset	R/W	Description	Reset Value
SC_DAT	SCn_BA+0x00	R/W	SC Receive/Transmit Holding Buffer Register	0xXXXX_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DAT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DAT**	**Receive/Transmit Holding Buffer** Write Operation: By writing data to DAT, the SC will send out an 8-bit data. Read Operation: By reading DAT, the SC will return an 8-bit received data. **Note:** If SCEN (SCn_CTL[0]) is not enabled, DAT cannot be programmed.

SC Control Register (SC_CTL)

Register	Offset	R/W	Description	Reset Value
SC_CTL	SCn_BA+0x04	R/W	SC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	SYNC	Reserved			CDLV	CDDBSEL	
23	22	21	20	19	18	17	16
TXRTYEN	TXRTY			RXRTYEN	RXRTY		
15	14	13	12	11	10	9	8
NSB	TMRSEL		BGT				
7	6	5	4	3	2	1	0
RXTRGLV		CONSEL		AUTOCEN	TXOFF	RXOFF	SCEN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit before writing a new value to RXRTY and TXRTY fields. 0 = Synchronizing is completion, user can write new data to RXRTY and TXRTY. 1 = Last value is synchronizing.
[29:27]	**Reserved**	Reserved.
[26]	**CDLV**	**Card Detect Level Selection** 0 = When hardware detects the card detect pin (SCn_CD) from high to low, it indicates a card is detected. 1 = When hardware detects the card detect pin (SCn_CD) from low to high, it indicates a card is detected. **Note:** User must select card detect level before Smart Card controller enabled.
[25:24]	**CDDBSEL**	**Card Detect De-bounce Selection** This field indicates the card detect de-bounce selection. 00 = De-bounce sample card insert once per 384 (128 * 3) SC module clocks and de-bounce sample card removal once per 128 SC module clocks. Other configurations are reserved.
[23]	**TXRTYEN**	**TX Error Retry Enable Bit** This bit enables transmitter retry function when parity error has occurred. 0 = TX error retry function Disabled. 1 = TX error retry function Enabled.
[22:20]	**TXRTY**	**TX Error Retry Count Number** This field indicates the maximum number of transmitter retries that are allowed when parity error has occurred. **Note 1:** The real retry number is TXRTY + 1, so 8 is the maximum retry number. **Note 2:** This field cannot be changed when TXRTYEN enabled. The change flow is to disable TXRTYEN first and then fill in new retry value.
[19]	**RXRTYEN**	**RX Error Retry Enable Bit** This bit enables receiver retry function when parity error has occurred.

		0 = RX error retry function Disabled. 1 = RX error retry function Enabled. **Note:** User must fill in the RXRTY value before enabling this bit.
[18:16]	**RXRTY**	**RX Error Retry Count Number** This field indicates the maximum number of receiver retries that are allowed when parity error has occurred **Note 1:** The real retry number is RXRTY + 1, so 8 is the maximum retry number. **Note 2:** This field cannot be changed when RXRTYEN enabled. The change flow is to disable RXRTYEN first and then fill in new retry value.
[15]	**NSB**	**Stop Bit Length** This field indicates the length of stop bit. 0 = The stop bit length is 2 ETU. 1= The stop bit length is 1 ETU. **Note 1:** The default stop bit length is 2. SC and UART adopts NSB to program the stop bit length. **Note 2:** In UART mode, RX can receive the data sequence in 1 stop bit or 2 stop bits with NSB is set to 0.
[14:13]	**TMRSEL**	**Timer Channel Selection** 00 = All internal timer function Disabled. 11 = Internal 24-bit timer and two 8-bit timers Enabled. User can configure them by setting SCn_TMRCTL0[23:0], SCn_TMRCTL1[7:0] and SCn_TMRCTL2[7:0]. Other configurations are reserved.
[12:8]	**BGT**	**Block Guard Time** Block guard time means the minimum interval between the leading edges of two consecutive characters between different transfer directions. This field indicates the counter for the bit length of block guard time. According to ISO 7816-3, in T = 0 mode, user must fill 15 (real block guard time = 16.5) to this field; in T = 1 mode, user must fill 21 (real block guard time = 22.5) to it. **Note:** The real block guard time is BGT + 1.
[7:6]	**RXTRGLV**	**Rx Buffer Trigger Level** When the number of bytes in the receiving buffer equals the RXTRGLV, the RDAIF will be set. If RDAIEN (SCn_INTEN[0]) is enabled, an interrupt will be generated to CPU. 00 = Rx Buffer Trigger Level with 1 bytes. 01 = Rx Buffer Trigger Level with 2 bytes. 10 = Rx Buffer Trigger Level with 3 bytes. 11 = Reserved.
[5:4]	**CONSEL**	**Convention Selection** 00 = Direct convention. 01 = Reserved. 10 = Reserved. 11 = Inverse convention. **Note:** If AUTOCEN (SCn_CTL[3]) is enabled, this field is ignored.
[3]	**AUTOCEN**	**Auto Convention Enable Bit** This bit is used for enable auto convention function. 0 = Auto-convention Disabled. 1 = Auto-convention Enabled. **Note 1:** If user enables auto convention function, the setting step must be done before Answer to Reset (ATR) state and the first data must be 0x3B or 0x3F. After hardware received first data and stored it at buffer, hardware will decided the convention and change the CONSEL (SCn_CTL[5:4]) bits automatically when received first data is 0x3B or 0x3F. If received first byte is 0x3B, TS is direct convention, CONSEL (SCn_CTL[5:4]) will be set to 00 automatically, otherwise the TS is inverse convention, and CONSEL (SCn_CTL[5:4]) will be set to 11. **Note 2:** If the first data is not 0x3B or 0x3F, hardware will set ACERRIF (SCn_INTSTS[10]) and generate

		an interrupt to CPU when ACERRIEN (SCn_INTEN[10]) is enabled.
[2]	**TXOFF**	**TX Transition Disable Control Bit** This bit is used for disable Tx transition function. 0 = The transceiver Enabled. 1 = The transceiver Disabled.
[1]	**RXOFF**	**RX Transition Disable Control Bit** This bit is used for disable Rx transition function. 0 = The receiver Enabled. 1 = The receiver Disabled. **Note:** If AUTOCEN (SCn_CTL[3]) is enabled, this field is ignored.
[0]	**SCEN**	**SC Controller Enable Bit** Set this bit to 1 to enable SC operation. If this bit is cleared, 0 = SC will force all transition to IDLE state. 1 = SC controller is enabled and all function can work correctly. **Note:** SCEN must be set to 1 before filling in other SC registers, or smart card will not work properly.

SC Alternate Control Register (SC_ALTCTL)

Register	Offset	R/W	Description	Reset Value
SC_ALTCTL	SCn_BA+0x08	R/W	SC Alternate Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SYNC	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ACTSTS2	ACTSTS1	ACTSTS0	RXBGTEN	ADACEN	Reserved	INITSEL	
7	6	5	4	3	2	1	0
CNTEN2	CNTEN1	CNTEN0	WARSTEN	ACTEN	DACTEN	RXRST	TXRST

Bits	Description	
[31]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit when writing a new value to SCn_ALTCTL register. 0 = Synchronizing is completion, user can write new data to SCn_ALTCTL register. 1 = Last value is synchronizing.
[30:16]	**Reserved**	Reserved.
[15]	**ACTSTS2**	**Internal Timer2 Active Status (Read Only)** This bit indicates the timer counter status of Timer2. 0 = Timer2 is not active. 1 = Timer2 is active. **Note:** Timer2 is active does not always mean Timer2 is counting the CNT (SCn_TMRCTL2[7:0]).
[14]	**ACTSTS1**	**Internal Timer1 Active Status (Read Only)** This bit indicates the timer counter status of Timer1. 0 = Timer1 is not active. 1 = Timer1 is active. **Note:** Timer1 is active does not always mean Timer1 is counting the CNT (SCn_TMRCTL1[7:0]).
[13]	**ACTSTS0**	**Internal Timer0 Active Status (Read Only)** This bit indicates the timer counter status of Timer0. 0 = Timer0 is not active. 1 = Timer0 is active. **Note:** Timer0 is active does not always mean Timer0 is counting the CNT (SCn_TMRCTL0[23:0]).
[12]	**RXBGTEN**	**Receiver Block Guard Time Function Enable Bit** This bit enables the receiver block guard time function. 0 = Receiver block guard time function Disabled. 1 = Receiver block guard time function Enabled.
[11]	**ADACEN**	**Auto Deactivation When Card Removal** This bit is used for enable hardware auto deactivation when smart card is removed.

		0 = Auto deactivation Disabled. 1 = Auto deactivation Enabled. **Note:** When the card is removed, hardware will stop any process and then do deactivation sequence if this bit is set. If auto deactivation process completes, hardware will set INITIF (SCn_INTSTS[8]) also.
[10]	**Reserved**	Reserved.
[9:8]	**INITSEL**	**Initial Timing Selection** This fields indicates the initial timing of hardware activation, warm-reset or deactivation. The unit of initial timing is SC module clock. **Activation:** refer to SC Activation Sequence in Figure 6.25-4. **Warm-reset:** refer to Warm-Reset Sequence in Figure 6.25-5. **Deactivation:** refer to Deactivation Sequence in Figure 6.25-6. **Note:** When set activation and warm reset in Timer0 operation mode 0011, it may have deviation at most 128 SC module clock cycles.
[7]	**CNTEN2**	**Internal Timer2 Start Enable Bit** This bit enables Timer 2 to start counting. User can fill 0 to stop it and set 1 to reload and count. The counter unit is ETU base. 0 = Stops counting. 1 = Start counting. **Note 1:** This field is used for internal 8-bit timer when TMRSEL (SCn_CTL[14:13]) is 11 only. Do not fill in CNTEN2 when TMRSEL (SCn_CTL[14:13]) is not equal to 11. **Note 2:** If the operation mode is not in auto-reload mode (SCn_TMRCTL2[26] = 0), this bit will be auto-cleared by hardware. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed.
[6]	**CNTEN1**	**Internal Timer1 Start Enable Bit** This bit enables Timer 1 to start counting. User can fill 0 to stop it and set 1 to reload and count. The counter unit is ETU base. 0 = Stops counting. 1 = Start counting. **Note 1:** This field is used for internal 8-bit timer when TMRSEL(SCn_CTL[14:13]) is 11 only. Do not fill CNTEN1 when TMRSEL (SCn_CTL[14:13]) is not equal to 11. **Note 2:** If the operation mode is not in auto-reload mode (SCn_TMRCTL1[26] = 0), this bit will be auto-cleared by hardware. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed.
[5]	**CNTEN0**	**Internal Timer0 Start Enable Bit** This bit enables Timer 0 to start counting. User can fill 0 to stop it and set 1 to reload and count. The counter unit is ETU base. 0 = Stops counting. 1 = Start counting. **Note 1:** This field is used for internal 24-bit timer when TMRSEL (SCn_CTL[14:13]) is 11 only. **Note 2:** If the operation mode is not in auto-reload mode (SCn_TMRCTL0[26] = 0), this bit will be auto-cleared by hardware. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed.
[4]	**WARSTEN**	**Warm Reset Sequence Generator Enable Bit** This bit enables SC controller to initiate the card by warm reset sequence. 0 = No effect. 1 = Warm reset sequence generator Enabled. **Note 1:** When the warm reset sequence completed, this bit will be cleared automatically and the INITIF (SCn_INTSTS[8]) will be set to 1. **Note 2:** This field will be cleared by TXRST (SCn_ALTCTL[0]) and RXRST (SCn_ALTCTL[1]). Thus, do not fill in this bit WARSTEN, TXRST and RXRST at the same time.

		Note 3: If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed. **Note 4:** During the warm reset sequence, RX is disabled automatically and cannot receive data. After the warm reset sequence completion, RXOFF (SCn_CTL[1]) keeps the state before perform warm reset sequence.
[3]	**ACTEN**	**Activation Sequence Generator Enable Bit** This bit enables SC controller to initiate the card by activation sequence. 0 = No effect. 1 = Activation sequence generator Enabled. **Note 1:** When the activation sequence completed, this bit will be cleared automatically and the INITIF (SCn_INTSTS[8]) will be set to 1. **Note 2:** This field will be cleared by TXRST (SCn_ALTCTL[0]) and RXRST (SCn_ALTCTL[1]). Thus, do not fill in this bit ACTEN, TXRST and RXRST at the same time. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed. **Note 4:** During the activation sequence, RX is disabled automatically and cannot receive data. After the activation sequence completion, RXOFF (SCn_CTL[1]) keeps the state before hardware activation.
[2]	**DACTEN**	**Deactivation Sequence Generator Enable Bit** This bit enables SC controller to initiate the card by deactivation sequence. 0 = No effect. 1 = Deactivation sequence generator Enabled. **Note 1:** When the deactivation sequence completed, this bit will be cleared automatically and the INITIF (SCn_INTSTS[8]) will be set to 1. **Note 2:** This field will be cleared by TXRST (SCn_ALTCTL[0]) and RXRST (SCn_ALTCTL[1]). Thus, do not fill in this bit DACTEN, TXRST and RXRST at the same time. **Note 3:** If SCEN (SCn_CTL[0]) is not enabled, this filed cannot be programmed.
[1]	**RXRST**	**Rx Software Reset** When RXRST is set, all the bytes in the receive buffer and Rx internal state machine will be cleared. 0 = No effect. 1 = Reset the Rx internal state machine and pointers. **Note:** This bit will be auto cleared after reset is complete.
[0]	**TXRST**	**TX Software Reset** When TXRST is set, all the bytes in the transmit buffer and TX internal state machine will be cleared. 0 = No effect. 1 = Reset the TX internal state machine and pointers. **Note:** This bit will be auto cleared after reset is complete.

SC Extra Guard Time Register (SC_EGT)

Register	Offset	R/W	Description	Reset Value
SC_EGT	SCn_BA+0x0C	R/W	SC Extra Guard Time Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
EGT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**EGT**	**Extra Guard Time** This field indicates the extra guard time value. **Note:** The extra guard time unit is ETU base.

SC Receiver Buffer Time-out Register (SC_RXTOUT)

Register	Offset	R/W	Description	Reset Value
SC_RXTOUT	SCn_BA+0x10	R/W	SC Receive Buffer Time-out Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							RFTM
7	6	5	4	3	2	1	0
RFTM							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**RFTM**	**SC Receiver FIFO Time-out Counter** The time-out down counter resets and starts counting whenever the RX buffer received a new data. Once the counter decrease to 1 and no new data is received or CPU does not read data by reading SCn_DAT, a receiver time-out flag RXTOIF (SCn_INTSTS[9]) will be set, and hardware will generate an interrupt to CPU when RXTOIEN (SCn_INTEN[9]) is enabled. **Note 1:** The counter unit is ETU based and the interval of time-out is RFTM + 0.5. **Note 2:** Filling in all 0 to this field indicates to disable this function.

SC Element Time Unit Control Register (SC_ETUCTL)

Register	Offset	R/W	Description	Reset Value
SC_ETUCTL	SCn_BA+0x14	R/W	SC Element Time Unit Control Register	0x0000_0173

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				ETURDIV			
7	6	5	4	3	2	1	0
ETURDIV							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**ETURDIV**	**ETU Rate Divider** The field is used for ETU clock rate divider. The real ETU is ETURDIV + 1. **Note:** User can configure this field, but this field must be greater than 0x04.

SC Interrupt Enable Control Register (SC_INTEN)

Register	Offset	R/W	Description	Reset Value
SC_INTEN	SCn_BA+0x18	R/W	SC Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					ACERRIEN	RXTOIEN	INITIEN
7	6	5	4	3	2	1	0
CDIEN	BGTIEN	TMR2IEN	TMR1IEN	TMR0IEN	TERRIEN	TBEIEN	RDAIEN

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**ACERRIEN**	**Auto Convention Error Interrupt Enable Bit** This field is used to enable auto-convention error interrupt. 0 = Auto-convention error interrupt Disabled. 1 = Auto-convention error interrupt Enabled.
[9]	**RXTOIEN**	**Receiver Buffer Time-out Interrupt Enable Bit** This field is used to enable receiver buffer time-out interrupt. 0 = Receiver buffer time-out interrupt Disabled. 1 = Receiver buffer time-out interrupt Enabled.
[8]	**INITIEN**	**Initial End Interrupt Enable Bit** This field is used to enable activation (ACTEN (SCn_ALTCTL[3] = 1)), deactivation (DACTEN (SCn_ALTCTL[2] = 1)) and warm reset (WARSTEN (SCn_ALTCTL [4])) sequence complete interrupt. 0 = Initial end interrupt Disabled. 1 = Initial end interrupt Enabled.
[7]	**CDIEN**	**Card Detect Interrupt Enable Bit** This field is used to enable card detect interrupt. The card detect status is CDPINSTS (SCn_STATUS[13]). 0 = Card detect interrupt Disabled. 1 = Card detect interrupt Enabled.
[6]	**BGTIEN**	**Block Guard Time Interrupt Enable Bit** This field is used to enable block guard time interrupt in receive direction. 0 = Block guard time interrupt Disabled. 1 = Block guard time interrupt Enabled. **Note:** This bit is valid only for receive direction block guard time.
[5]	**TMR2IEN**	**Timer2 Interrupt Enable Bit** This field is used to enable Timer2 interrupt function. 0 = Timer2 interrupt Disabled. 1 = Timer2 interrupt Enabled.

[4]	**TMR1IEN**	**Timer1 Interrupt Enable Bit** This field is used to enable the Timer1 interrupt function. 0 = Timer1 interrupt Disabled. 1 = Timer1 interrupt Enabled.
[3]	**TMR0IEN**	**Timer0 Interrupt Enable Bit** This field is used to enable Timer0 interrupt function. 0 = Timer0 interrupt Disabled. 1 = Timer0 interrupt Enabled.
[2]	**TERRIEN**	**Transfer Error Interrupt Enable Bit** This field is used to enable transfer error interrupt. The transfer error states is at SCn_STATUS register which includes receiver break error BEF (SCn_STATUS[6]), frame error FEF (SCn_STATUS[5]), parity error PEF (SCn_STATUS[4]), receive buffer overflow error RXOV (SCn_STATUS[0]), transmit buffer overflow error TXOV (SCn_STATUS[8]), receiver retry over limit error RXOVERR (SCn_STATUS[22]) and transmitter retry over limit error TXOVERR (SCn_STATUS[30]). 0 = Transfer error interrupt Disabled. 1 = Transfer error interrupt Enabled.
[1]	**TBEIEN**	**Transmit Buffer Empty Interrupt Enable Bit** This field is used to enable transmit buffer empty interrupt. 0 = Transmit buffer empty interrupt Disabled. 1 = Transmit buffer empty interrupt Enabled.
[0]	**RDAIEN**	**Receive Data Reach Interrupt Enable Bit** This field is used to enable received data reaching trigger level RXTRGLV (SCn_CTL[7:6]) interrupt. 0 = Receive data reach trigger level interrupt Disabled. 1 = Receive data reach trigger level interrupt Enabled.

SC Interrupt Status Register (SC_INTSTS)

Register	Offset	R/W	Description	Reset Value
SC_INTSTS	SCn_BA+0x1C	R/W	SC Interrupt Status Register	0x0000_0002

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					ACERRIF	RXTOIF	INITIF
7	6	5	4	3	2	1	0
CDIF	BGTIF	TMR2IF	TMR1IF	TMR0IF	TERRIF	TBEIF	RDAIF

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**ACERRIF**	**Auto Convention Error Interrupt Status Flag** This field indicates auto convention sequence error. 0 = Received TS at ATR state is 0x3B or 0x3F. 1 = Received TS at ATR state is neither 0x3B nor 0x3F. **Note:** This bit can be cleared by writing 1 to it.
[9]	**RXTOIF**	**Receive Buffer Time-out Interrupt Status Flag (Read Only)** This field is used for indicate receive buffer time-out interrupt status flag. 0 = Receive buffer time-out interrupt did not occur. 1 = Receive buffer time-out interrupt occurred. **Note:** This bit is read only, user must read all receive buffer remaining data by reading SCn_DAT register to clear it.
[8]	**INITIF**	**Initial End Interrupt Status Flag** This field is used for activation (ACTEN (SCn_ALTCTL[3])), deactivation (DACTEN (SCn_ALTCTL[2])) and warm reset (WARSTEN (SCn_ALTCTL[4])) sequence interrupt status flag. 0 = Initial sequence is not complete. 1 = Initial sequence is completed. **Note:** This bit can be cleared by writing 1 to it.
[7]	**CDIF**	**Card Detect Interrupt Status Flag (Read Only)** This field is used for card detect interrupt status flag. The card detect status is CINSERT (SCn_STATUS[12]) and CREMOVE (SCn_STATUS[11]). 0 = Card detect event did not occur. 1 = Card detect event occurred. **Note:** This bit is read only, user must to clear CINSERT or CREMOVE status to clear it.
[6]	**BGTIF**	**Block Guard Time Interrupt Status Flag** This field is used for indicate block guard time interrupt status flag in receive direction. 0 = Block guard time interrupt did not occur. 1 = Block guard time interrupt occurred.

		Note 1: This bit is valid only when RXBGTEN (SCn_ALTCTL[12]) is enabled. **Note 2:** This bit can be cleared by writing 1 to it.
[5]	**TMR2IF**	**Timer2 Interrupt Status Flag** This field is used for Timer2 interrupt status flag. 0 = Timer2 interrupt did not occur. 1 = Timer2 interrupt occurred. **Note:** This bit can be cleared by writing 1 to it.
[4]	**TMR1IF**	**Timer1 Interrupt Status Flag** This field is used for Timer1 interrupt status flag. 0 = Timer1 interrupt did not occur. 1 = Timer1 interrupt occurred. **Note:** This bit can be cleared by writing 1 to it.
[3]	**TMR0IF**	**Timer0 Interrupt Status Flag** This field is used for Timer0 interrupt status flag. 0 = Timer0 interrupt did not occur. 1 = Timer0 interrupt occurred. **Note:** This bit can be cleared by writing 1 to it.
[2]	**TERRIF**	**Transfer Error Interrupt Status Flag** This field is used for transfer error interrupt status flag. The transfer error states is at SCn_STATUS register which includes receiver break error BEF (SCn_STATUS[6]), frame error FEF (SCn_STATUS[5], parity error PEF (SCn_STATUS[4] and receive buffer overflow error RXOV (SCn_STATUS[0]), transmit buffer overflow error TXOV (SCn_STATUS[8]), receiver retry over limit error RXOVERR (SCn_STATUS[22] or transmitter retry over limit error TXOVERR (SCn_STATUS[30]). 0 = Transfer error interrupt did not occur. 1 = Transfer error interrupt occurred. **Note 1:** This field is the status flag of BEF, FEF, PEF, RXOV, TXOV, RXOVERR or TXOVERR. **Note 2:** This bit can be cleared by writing 1 to it.
[1]	**TBEIF**	**Transmit Buffer Empty Interrupt Status Flag (Read Only)** This field is used for transmit buffer empty interrupt status flag. 0 = Transmit buffer is not empty. 1 = Transmit buffer is empty. **Note:** This bit is read only. If user wants to clear this bit, user must write data to DAT (SCn_DAT[7:0]) and then this bit will be cleared automatically.
[0]	**RDAIF**	**Receive Data Reach Interrupt Status Flag (Read Only)** This field is used for received data reaching trigger level RXTRGLV (SCn_CTL[7:6]) interrupt status flag. 0 = Number of receive buffer is less than RXTRGLV setting. 1 = Number of receive buffer data equals the RXTRGLV setting. **Note:** This bit is read only. If user reads data from SCn_DAT and receiver buffer data byte number is less than RXTRGLV, this bit will be cleared automatically.

SC Transfer Status Register (SC_STATUS)

Register	Offset	R/W	Description	Reset Value
SC_STATUS	SCn_BA+0x20	R/W	SC Transfer Status Register	0x0000_X202

31	30	29	28	27	26	25	24
TXACT	TXOVERR	TXRERR	Reserved		TXPOINT		
23	22	21	20	19	18	17	16
RXACT	RXOVERR	RXRERR	Reserved		RXPOINT		
15	14	13	12	11	10	9	8
Reserved		CDPINSTS	CINSERT	CREMOVE	TXFULL	TXEMPTY	TXOV
7	6	5	4	3	2	1	0
Reserved	BEF	FEF	PEF	Reserved	RXFULL	RXEMPTY	RXOV

Bits	Description	
[31]	**TXACT**	**Transmit in Active Status Flag (Read Only)** This bit indicates Tx transmit status. 0 = This bit is cleared automatically when Tx transfer is finished or the last byte transmission has completed. 1 = Transmit is active and this bit is set by hardware when Tx transfer is in active and the STOP bit of the last byte has not been transmitted.
[30]	**TXOVERR**	**Transmitter over Retry Error** This bit is used for transmitter retry counts over than retry number limitation. 0 = Transmitter retries counts is less than TXRTY (SCn_CTL[22:20]) + 1. 1 = Transmitter retries counts is equal or over to TXRTY (SCn_CTL[22:20]) + 1. **Note:** This bit can be cleared by writing 1 to it.
[29]	**TXRERR**	**Transmitter Retry Error** This bit is used for indicate transmitter error retry and set by hardware. 0 = No Tx retry transfer. 1 = Tx has any error and retries transfer. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is a flag and cannot generate any interrupt to CPU.
[28:27]	**Reserved**	Reserved.
[26:24]	**TXPOINT**	**Transmit Buffer Pointer Status (Read Only)** This field indicates the Tx buffer pointer status. When CPU writes data into SCn_DAT, TXPOINT increases one. When one byte of Tx buffer is transferred to transmitter shift register, TXPOINT decreases one.
[23]	**RXACT**	**Receiver in Active Status Flag (Read Only)** This bit indicates Rx transfer status. 0 = This bit is cleared automatically when Rx transfer is finished. 1 = This bit is set by hardware when Rx transfer is in active.
[22]	**RXOVERR**	**Receiver over Retry Error** This bit is used for receiver retry counts over than retry number limitation.

		0 = Receiver retries counts is less than RXRTY (SCn_CTL[18:16]) + 1. 1 = Receiver retries counts is equal or over than RXRTY (SCn_CTL[18:16]) + 1. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** If CPU enables receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[21]	**RXRERR**	**Receiver Retry Error** This bit is used for receiver error retry and set by hardware. 0 = No Rx retry transfer. 1 = Rx has any error and retries transfer. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** This bit is a flag and cannot generate any interrupt to CPU. **Note 3:** If CPU enables receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[20:19]	**Reserved**	Reserved.
[18:16]	**RXPOINT**	**Receive Buffer Pointer Status (Read Only)** This field indicates the Rx buffer pointer status. When SC controller receives one byte from external device, RXPOINT increases one. When one byte of Rx buffer is read by CPU, RXPOINT decreases one.
[15:14]	**Reserved**	Reserved.
[13]	**CDPINSTS**	**Card Detect Pin Status (Read Only)** This bit is the pin status of SCn_CD. 0 = The SCn_CD pin state at low. 1 = The SCn_CD pin state at high.
[12]	**CINSERT**	**Card Insert Status of SCn_CD Pin** This bit is set whenever card has been inserted. 0 = No effect. 1 = Card insert. **Note 1:** This bit can be cleared by writing “1” to it. **Note 2:** The card detect function will start after SCEN (SCn_CTL[0]) set.
[11]	**CREMOVE**	**Card Removal Status of SCn_CD Pin** This bit is set whenever card has been removal. 0 = No effect. 1 = Card removed. **Note 1:** This bit can be cleared by writing “1” to it. **Note 2:** Card detect function will start after SCEN (SCn_CTL[0]) set.
[10]	**TXFULL**	**Transmit Buffer Full Status Flag (Read Only)** This bit indicates Tx buffer full or not. 0 = Tx buffer count is less than 4. 1 = Tx buffer count equals 4.
[9]	**TXEMPTY**	**Transmit Buffer Empty Status Flag (Read Only)** This bit indicates TX buffer empty or not. 0 = Tx buffer is not empty. 1 = Tx buffer is empty, which means the last byte of Tx buffer has been transferred to Transmitter Shift Register. **Note:** This bit will be cleared when writing data into DAT (SCn_DAT[7:0]).
[8]	**TXOV**	**Transmit Overflow Error Interrupt Status Flag** This bit is set when Tx buffer overflow.

		0 = Tx buffer is not overflow. 1 = Tx buffer is overflow when Tx buffer is full and an additional write operation to DAT (SCn_DAT[7:0]). **Note:** This bit can be cleared by writing 1 to it.
[7]	**Reserved**	Reserved.
[6]	**BEF**	**Receiver Break Error Status Flag** This bit is set to logic 1 whenever the received data input (Rx) held in the "spacing state" (logic 0) is longer than a full word transmission time (that is, the total time of “start bit” + “data bits” + “parity bit” + “stop bits”). 0 = Receiver break error flag did not occur. 1 = Receiver break error flag occurred. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** If CPU sets receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[5]	**FEF**	**Receiver Frame Error Status Flag** This bit is set to logic 1 whenever the received character does not have a valid "stop bit" (that is, the stop bit following the last data bit or parity bit is detected as logic 0). 0 = Receiver frame error flag did not occur. 1 = Receiver frame error flag occurred. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** If CPU sets receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[4]	**PEF**	**Receiver Parity Error Status Flag** This bit is set to logic 1 whenever the received character does not have a valid "parity bit". 0 = Receiver parity error flag did not occur. 1 = Receiver parity error flag occurred. **Note 1:** This bit can be cleared by writing 1 to it. **Note 2:** If CPU sets receiver retries function by setting RXRTYEN (SCn_CTL[19]), hardware will not set this flag.
[3]	**Reserved**	Reserved.
[2]	**RXFULL**	**Receive Buffer Full Status Flag (Read Only)** This bit indicates Rx buffer full or not. 0 = Rx buffer count is less than 4. 1 = Rx buffer count equals 4.
[1]	**RXEMPTY**	**Receive Buffer Empty Status Flag (Read Only)** This bit indicates Rx buffer empty or not. 0 = Rx buffer is not empty. 1 = Rx buffer is empty, which means the last byte of Rx buffer has read from DAT (SCn_DAT[7:0]) by CPU.
[0]	**RXOV**	**Receive Overflow Error Status Flag** This bit is set when Rx buffer overflow. 0 = Rx buffer is not overflow. 1 = Rx buffer is overflow when the number of received bytes is greater than Rx buffer size (4 bytes). **Note:** This bit can be cleared by writing 1 to it.

SC Pin Control State Register (SC_PINCTL)

Register	Offset	R/W	Description	Reset Value
SC_PINCTL	SCn_BA+0x24	R/W	SC Pin Control State Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	SYNC	Reserved					
23	22	21	20	19	18	17	16
Reserved					RSTSTS	PWRSTS	DATASTS
15	14	13	12	11	10	9	8
Reserved				PWRINV	Reserved	SCDATA	Reserved
7	6	5	4	3	2	1	0
Reserved	CLKKEEP	Reserved				RSTEN	PWREN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit when writing a new value to SCn_PINCTL register. 0 = Synchronizing is completion, user can write new data to SCn_PINCTL register. 1 = Last value is synchronizing.
[29:19]	**Reserved**	Reserved.
[18]	**RSTSTS**	**SCn_RST Pin Status (Read Only)** This bit is the pin status of SCn_RST. 0 = SCn_RST pin is low. 1 = SCn_RST pin is high.
[17]	**PWRSTS**	**SCn_PWR Pin Status (Read Only)** This bit is the pin status of SCn_PWR. 0 = SCn_PWR pin to low. 1 = SCn_PWR pin to high.
[16]	**DATASTS**	**SCn_DATA Pin Status (Read Only)** This bit is the pin status of SCn_DATA. 0 = The SCn_DATA pin status is low. 1 = The SCn_DATA pin status is high.
[15:12]	**Reserved**	Reserved.
[11]	**PWRINV**	**SCn_PWR Pin Inverse** This bit is used for inverse the SCn_PWR pin. There are four kinds of combination for SCn_PWR pin setting by PWRINV (SCn_PINCTL[11]) and PWREN (SCn_PINCTL[0]). PWRINV (SCn_PINCTL[11]) is bit 1 and PWREN (SCn_PINCTL[0]) is bit 0 and all conditions as below list, 00 = SCn_PWR pin is 0. 01 = SCn_PWR pin is 1.

		10 = SCn_PWR pin is 1. 11 = SCn_PWR pin is 0. **Note:** User must select PWRINV (SCn_PINCTL[11]) before smart card is enabled by SCEN (SCn_CTL[0]).
[10]	**Reserved**	Reserved.
[9]	**SCDATA**	**SCn_DATA Pin Signal** This bit is the signal status of SCn_DATA but user can drive SCn_DATA pin to high or low by setting this bit. 0 = Drive SCn_DATA pin to low. 1 = Drive SCn_DATA pin to high. Read this field to get SCn_DATA signal status. 0 = SCn_DATA signal status is low. 1 = SCn_DATA signal status is high. **Note:** When SC is at activation, warm reset or deactivation mode, this bit will be changed automatically. Thus, do not fill in this field when SC is in these modes.
[8:7]	**Reserved**	Reserved.
[6]	**CLKKEEP**	**SC Clock Enable Bit** 0 = SC clock generation Disabled. 1 = SC clock always keeps free running. **Note:** When operating in activation, warm reset or deactivation mode, this bit will be changed automatically. Thus, do not fill in this field when operating in these modes.
[5:2]	**Reserved**	Reserved.
[1]	**RSTEN**	**SCn_RST Pin Signal** User can set RSTEN (SCn_PINCTL[1]) to decide SCn_RST pin is in high or low level. Write this field to drive SCn_RST pin. 0 = Drive SCn_RST pin to low. 1 = Drive SCn_RST pin to high. Read this field to get SCn_RST signal status. 0 = SCn_RST signal status is low. 1 = SCn_RST signal status is high. **Note:** When operating at activation, warm reset or deactivation mode, this bit will be changed automatically. Thus, do not fill in this field when operating in these modes.
[0]	**PWREN**	**SCn_PWR Pin Signal** User can set PWRINV (SCn_PINCTL[11]) and PWREN (SCn_PINCTL[0]) to decide SCn_PWR pin is in high or low level. Write this field to drive SCn_PWR pin. Refer PWRINV (SCn_PINCTL[11]) description for programming SCn_PWR pin voltage level. Read this field to get SCn_PWR signal status. 0 = SCn_PWR signal status is low. 1 = SCn_PWR signal status is high. **Note:** When operating at activation, warm reset or deactivation mode, this bit will be changed automatically. Thus, do not fill in this field when operating in these modes.

SC Timer0 Control Register (SC_TMRCTL0)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL0	SCn_BA+0x28	R/W	SC Internal Timer0 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SYNC	Reserved			OPMODE			
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit when writing a new value to the SCn_TMRCTL0 register. 0 = Synchronizing is completion, user can write new data to SCn_TMRCTL0 register. 1 = Last value is synchronizing.
[30:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer0 Operation Mode Selection** This field indicates the internal 24-bit Timer0 operation selection. Refer to Table 6.25-3 for programming Timer0.
[23:0]	**CNT**	**Timer0 Counter Value** This field indicates the internal Timer0 counter values. **Note:** Unit of Timer0 counter is ETU base.

SC Timer1 Control Register (SC_TMRCTL1)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL1	SCn_BA+0x2C	R/W	SC Internal Timer1 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SYNC	Reserved			OPMODE			
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, software should check this bit when writing a new value to SCn_TMRCTL1 register. 0 = Synchronizing is completion, user can write new data to SCn_TMRCTL1 register. 1 = Last value is synchronizing.
[30:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer 1 Operation Mode Selection** This field indicates the internal 8-bit Timer1 operation selection. Refer to Table 6.25-3 for programming Timer1.
[23:8]	**Reserved**	Reserved.
[7:0]	**CNT**	**Timer 1 Counter Value** This field indicates the internal Timer1 counter values. **Note:** Unit of Timer1 counter is ETU base.

SC Timer2 Control Register (SC_TMRCTL2)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL2	SCn_BA+0x30	R/W	SC Internal Timer2 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SYNC	Reserved			OPMODE			
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31]	**SYNC**	**SYNC Flag Indicator (Read Only)** Due to synchronization, user should check this bit when writing a new value to SCn_TMRCTL2 register. 0 = Synchronizing is completion, user can write new data to SCn_TMRCTL2 register. 1 = Last value is synchronizing.
[30:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer 2 Operation Mode Selection** This field indicates the internal 8-bit Timer2 operation selection Refer to Table 6.25-3 for programming Timer2.
[23:8]	**Reserved**	Reserved.
[7:0]	**CNT**	**Timer 2 Counter Value** This field indicates the internal Timer2 counter values. **Note:** Unit of Timer2 counter is ETU base.

SC UART Mode Control Register (SC_UARTCTL)

Register	Offset	R/W	Description	Reset Value
SC_UARTCTL	SCn_BA+0x34	R/W	SC UART Mode Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
OPE	PBOFF	WLS		Reserved			UARTEN

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**OPE**	**Odd Parity Enable Bit** This is used for odd/even parity selection. 0 = Even number of logic 1 are transmitted or check the data word and parity bits in receiving mode. 1 = Odd number of logic 1 are transmitted or check the data word and parity bits in receiving mode. **Note:** This bit has effect only when PBOFF bit is 0.
[6]	**PBOFF**	**Parity Bit Disable Bit** This bit is used for disable parity check function. 0 = Parity bit is generated or checked between the “last data word bit” and “stop bit” of the serial data. 1 = Parity bit is not generated (transmitting data) or checked (receiving data) during transfer. **Note:** In smart card mode, this field must be 0 (default setting is with parity bit).
[5:4]	**WLS**	**Word Length Selection** This field is used for select UART data length. 00 = Word length is 8 bits. 01 = Word length is 7 bits. 10 = Word length is 6 bits. 11 = Word length is 5 bits. **Note:** In smart card mode, this WLS must be 00.
[3:1]	**Reserved**	Reserved.
[0]	**UARTEN**	**UART Mode Enable Bit** Set this bit to enable UART mode function. 0 = Smart Card mode. 1 = UART mode. **Note 1:** When operating in UART mode, user must set CONSEL (SCn_CTL[5:4]) = 00 and AUTOCEN (SCn_CTL[3]) = 0. **Note 2:** When operating in Smart Card mode, user must set UARTEN (SCn_UARTCTL[0]) = 0. **Note 3:** When UART mode is enabled, hardware will generate a reset to reset FIFO and internal state

Bits	Description	
		machine.

SC Activation Control Register (SC_ACTCTL)

Register	Offset	R/W	Description	Reset Value
SC_ACTCTL	SCn_BA+0x4C	R/W	SC Activation Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			T1EXT				

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**T1EXT**	**T1 Extend Time of Hardware Activation** This field provide the configurable cycles to extend the activation time T1 period. The cycle scaling factor is 2048. Extend cycles = (filled value * 2048) cycles. Refer to SC activation sequence in Figure 6.25-4. For example, SCLK = 4 MHz, each cycle = 0.25 us. Filled 20 to this field Extend time = 20 * 2048 * 0.25us = 10.24 ms. **Note:** Setting 0 to this field conforms to the protocol ISO/IEC 7816-3

6.26 I²S Controller (I²S)

6.26.1 Overview

The I²S controller consists of I²S protocol to interface with external audio CODEC. Two 16-level depth FIFO for reading path and writing path respectively are capable of handling 8/16/24/32 bits audio data sizes. A PDMA controller handles the data movement between FIFO and memory.

6.26.2 Features

- Supports Master mode and Slave mode
- Capable of handling 8, 16, 24 and 32 bits data sizes in each audio channel
- Supports monaural and stereo audio data
- Supports I²S protocols: Philips standard, MSB-justified, and LSB-justified data format
- Supports PCM protocols: PCM standard, MSB-justified, and LSB-justified data format
- PCM protocol supports TDM multi-channel transmission in one audio sample, and the number of data channel can be set as 2, 4, 6, or 8
- Provides two 16-level FIFO data buffers, one for transmitting and the other for receiving
- Generates interrupt requests when buffer levels cross a programmable boundary
- Supports two PDMA requests, one for transmitting and the other for receiving

6.26.3 Block Diagram

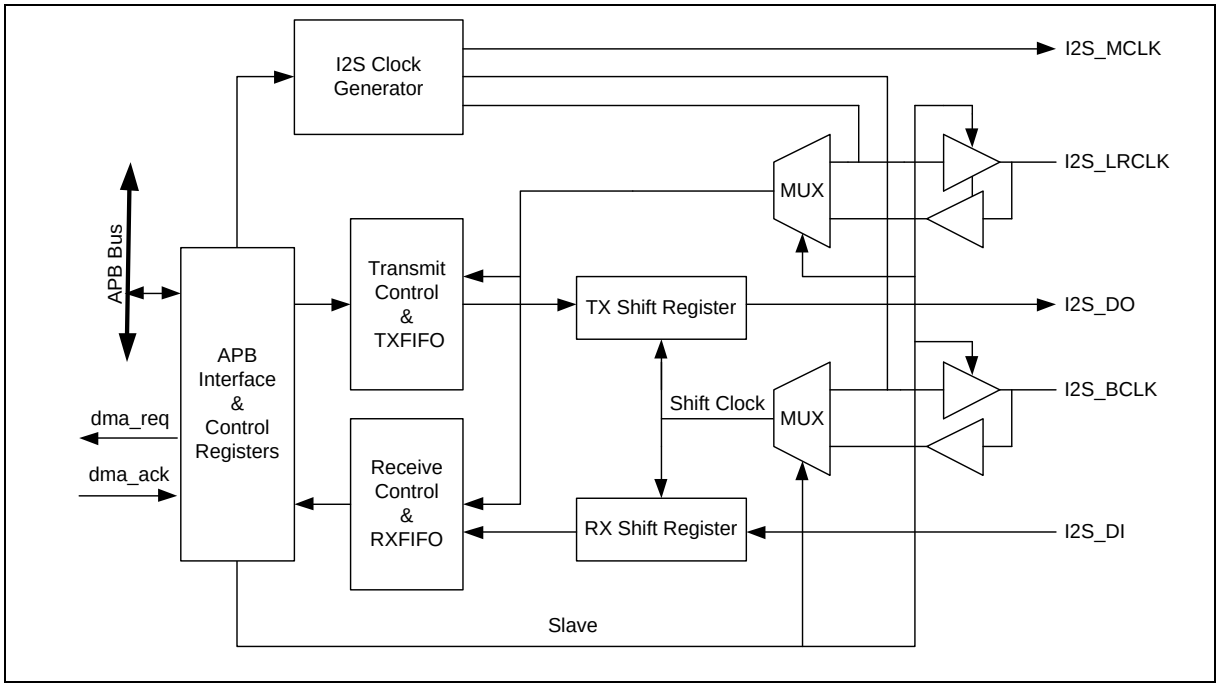


Figure 6.26-1 I²S Controller Block Diagram

6.26.4 Basic Configuration

6.26.4.1 I²S Basic Configuration

- Clock source Configuration

- Select the source of I²S peripheral clock on I2S0SEL (CLK_CLKSEL3[18:16]) and I2S1SEL (CLK_CLKSEL2[18:16]).
- Enable I²S peripheral clock in I2S0CKEN (CLK_APBCLK0[29]) and I2S1CKEN (CLK_APBCLK1[29]).

- Reset Configuration
  - Reset I²S controller in I2S0RST (SYS_IPRST1[29]) and I2S1RST (SYS_IPRST2[29]).

6.26.5 Functional Description

6.26.5.1 I²S Clock

The I²S controller has four clock sources selected by I2S0SEL (CLK_CLKSEL3[18:16]) and I2S1SEL (CLK_CLKSEL2[18:16]). The I²S clock rate must be slower than or equal to system clock rate.

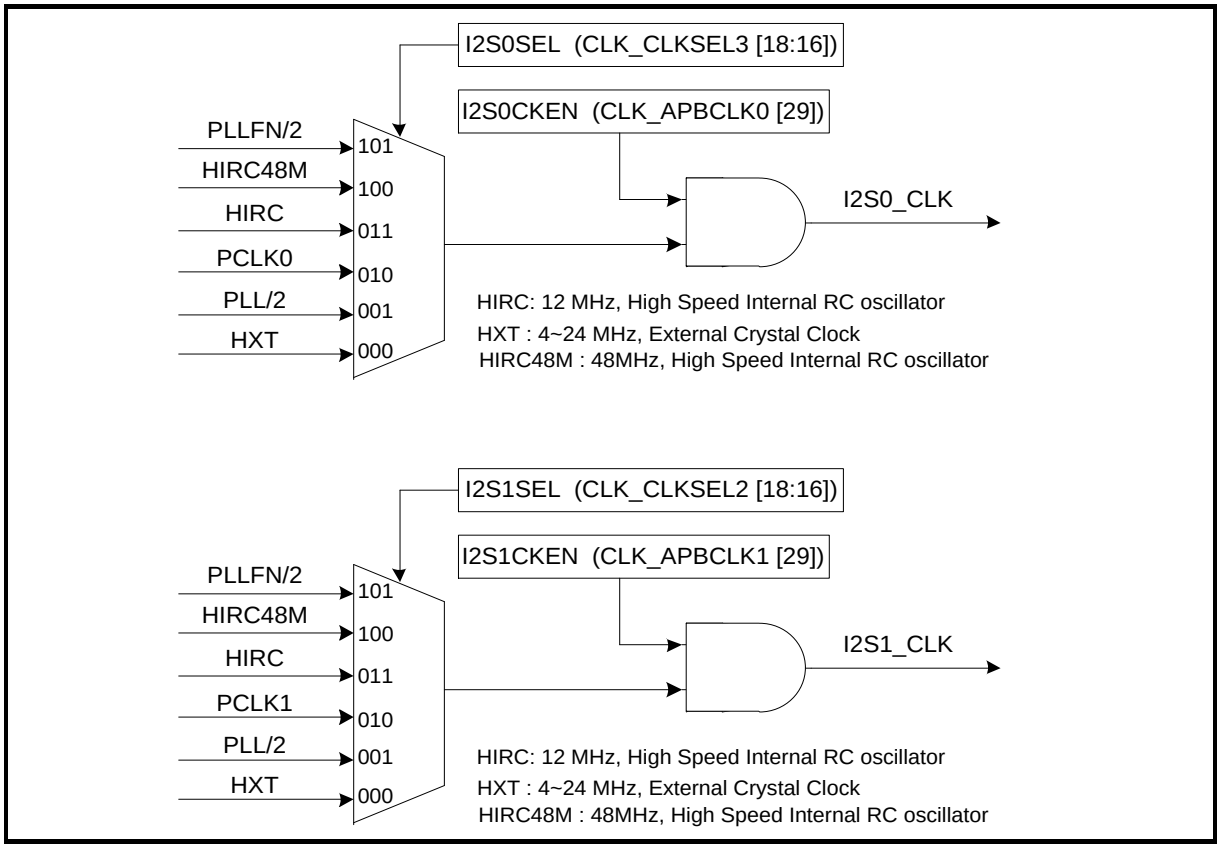


Figure 6.26-2 I²S Clock Control Diagram

6.26.5.2 Master/Slave Interface

The I²S function can operate as master or slave mode by setting SLAVE (I2S_CTL0[8]) to communicate with other I²S slave or master. The serial bus clock I2S_BCLK is permanently generated by the master even though there is no transferring data bit at the moment. The word select signal I2S_LRCLK is also generated by the master and it indicates the beginning of a new data word and the targeted audio channel. Both the I2S_LRCLK and the transmitting data change synchronously to the falling edges of I2S_BCLK.

In some applications, especially for Audio-ADC or Audio-DAC, a master clock signal, I2S_MCLK, is required with a fixed phase relation to the I2S_BCLK. The I2S_MCLK is enabled by MCLKEN (I2S_CTL0[15]). In Master mode, the I2S_MCLK, I2S_BCLK, I2S_LRCLK is output to device slave. And

if in slave mode, the I2S_MCLK is output to master, and I2S_BCLK or I2S_LRCLK is input from master.

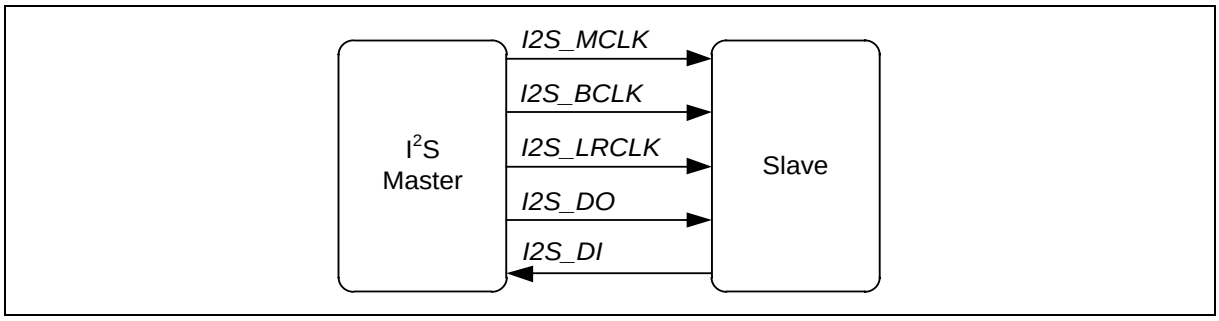


Figure 6.26-3 Master Mode Interface Block Diagram

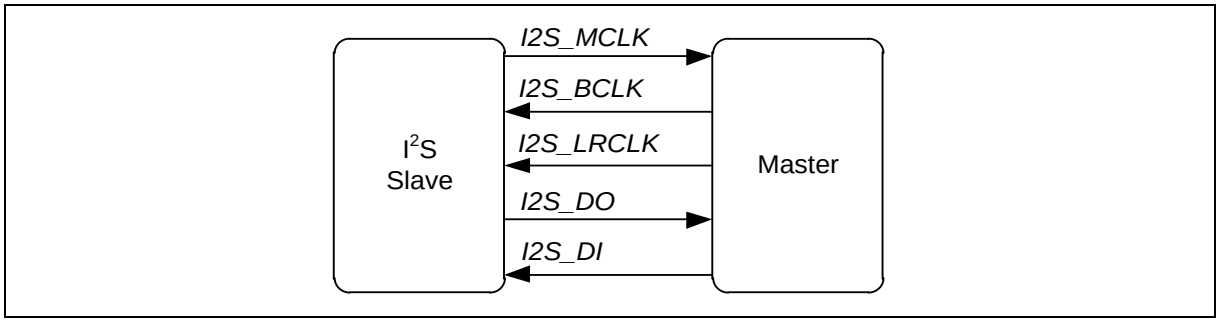


Figure 6.26-4 Slave Mode Interface Block Diagram

6.26.5.3 I²S Operation

The I²S controller supports MSB-justified, LSB-justified, and I²S Philips standard data format. The I2S_LRCLK signal indicates which audio channel is in transferring. The bit count of an audio channel is defined by CHWIDTH (I2S_CTL0[29:28]), and the bit-width of data word in an audio channel is determined by DATWIDTH (I2S_CTL0[5:4]). If CHWIDTH (I2S_CTL0[29:28]) is less than DATWIDTH (I2S_CTL0[5:4]), the hardware will set the channel bit-width to be same as data bit-width. However, there will be redundant zero bits in each audio channel if CHWIDTH (I2S_CTL0[29:28]) is greater than DATWIDTH (I2S_CTL0[5:4]).

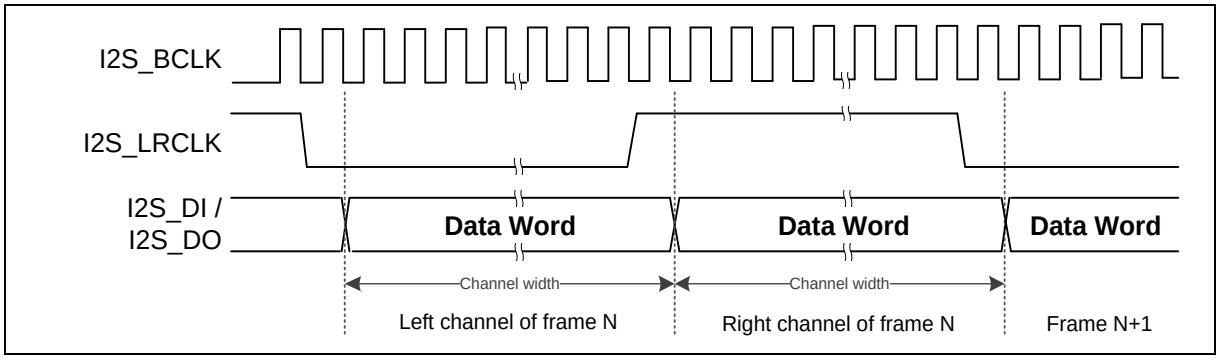


Figure 6.26-5 I²S Channel Width and Data Width (CHWIDTH≦DATWIDTH)

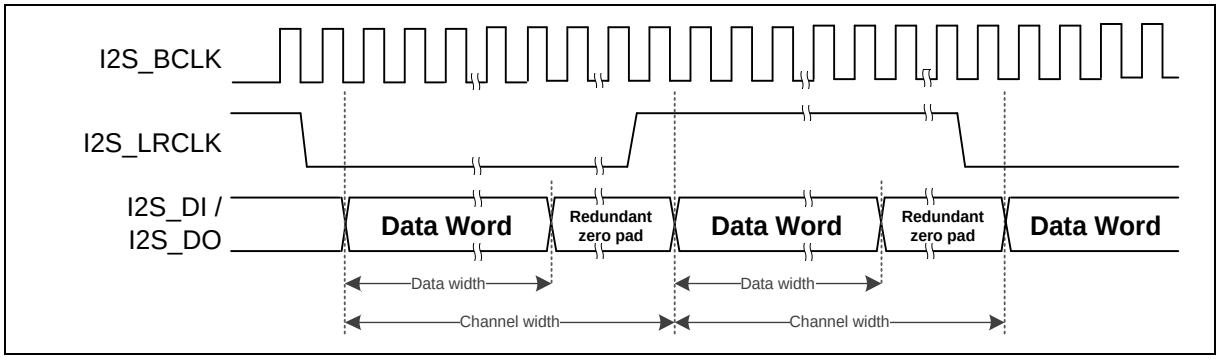


Figure 6.26-6 I²S Channel Width and Data Width (CHWIDTH > DATWIDTH)

The transferring data sequence is always started from the MSB (most significant bit) to the LSB (least significant bit). As shown in Figure 6.26-7, transmitting data are read at rising edge of I2S_BCLK and sent out at falling edge of I2S_BCLK in I²S protocol. In I²S data format, the MSB is sent and latched at the next falling edge of I2S_BCLK cycle after the transition of I2S_LRCLK. In MSB justified data format, the I2S_LRCLK changes the polarity at the transmitting of the first data bit (MSB) in each audio channel. In LSB justified data format, the LSB is sent and latched at the last I2S_BCLK cycle of an audio channel. The MSB justified and LSB justified data format of I²S protocol can be selected by FORMAT (I2S_CTL0[26:24]).

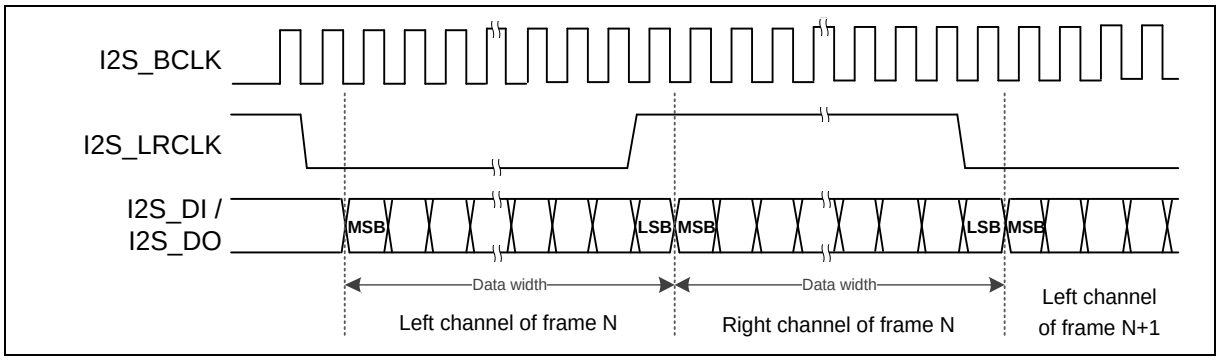


Figure 6.26-7 I²S Data Format Timing Diagram (FORMAT = 0x0 ; CHWIDTH≦DATWIDTH)

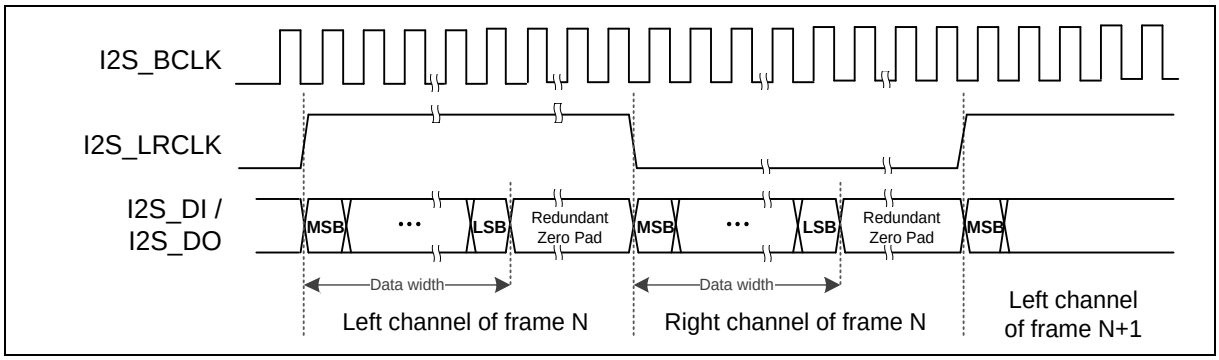


Figure 6.26-8 MSB Justified Data Format (FORMAT = 0x1 ; CHWIDTH > DATWIDTH)

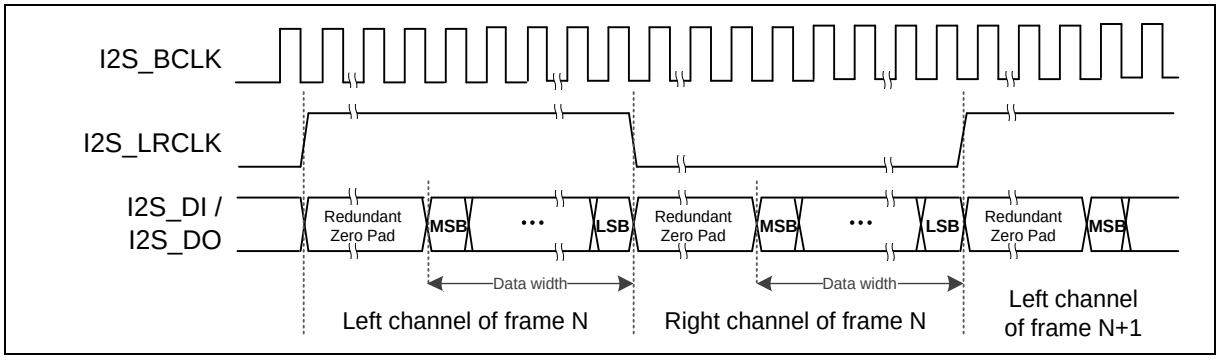


Figure 6.26-9 LSB Justified Data Format (FORMAT = 0x2 ; CHWIDTH > DATWIDTH)

The I²S controller also supports PCM audio transmission which can be selected by FORMAT (I2S_CTL0[26:24]). In PCM protocol, the function of I2S_LRCLK is simply to identify the beginning of an audio sample (or audio frame) and it is always indicated by the rising edge of the pulse. Therefore, the I2S_LRCLK in PCM protocol may be also called “frame start” or “frame sync” signal. In master device, there are two common representations for the width of the frame start pulse which can use PCMSYNC (I2S_CTL0[27]) to choose: One is equivalent to the period of a channel width and the other is equivalent to a single period of the I2S_BCLK.

Same as I²S protocol, the DATWIDTH (I2S_CTL0[5:4]) and CHWIDTH (I2S_CTL0[29:28]) can be used to configure the data bit-width and channel bit-width in PCM protocol. Besides, FORMAT (I2S_CTL0[26:24]) can also be used to select the different data formats of PCM standard mode, PCM with MSB justified, and PCM with LSB justified data format.

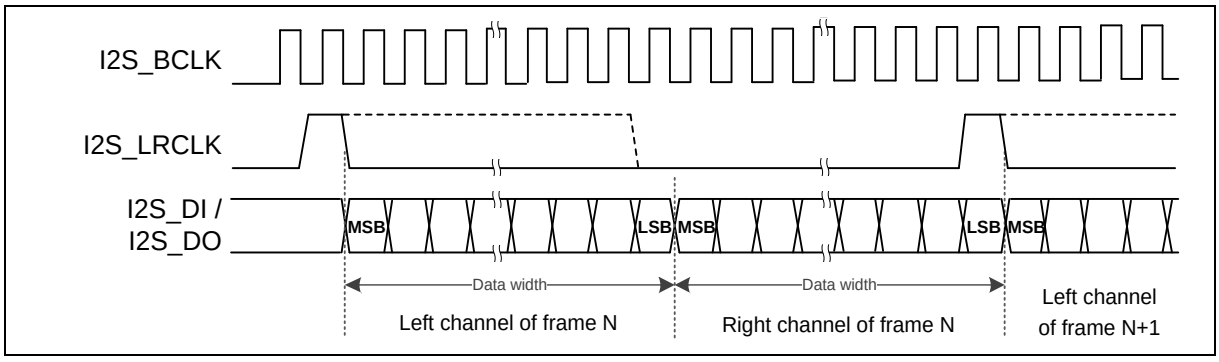


Figure 6.26-10 Standard PCM Audio Timing Diagram (FORMAT = 0x4 ; CHWIDTH≦DATWIDTH)

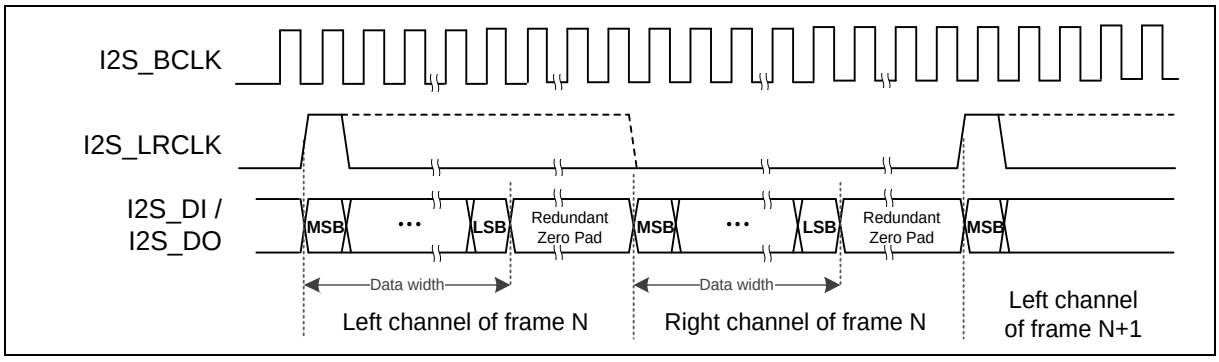


Figure 6.26-11 PCM with MSB Justified Data Format (FORMAT = 0x5 ; CHWIDTH > DATWIDTH)

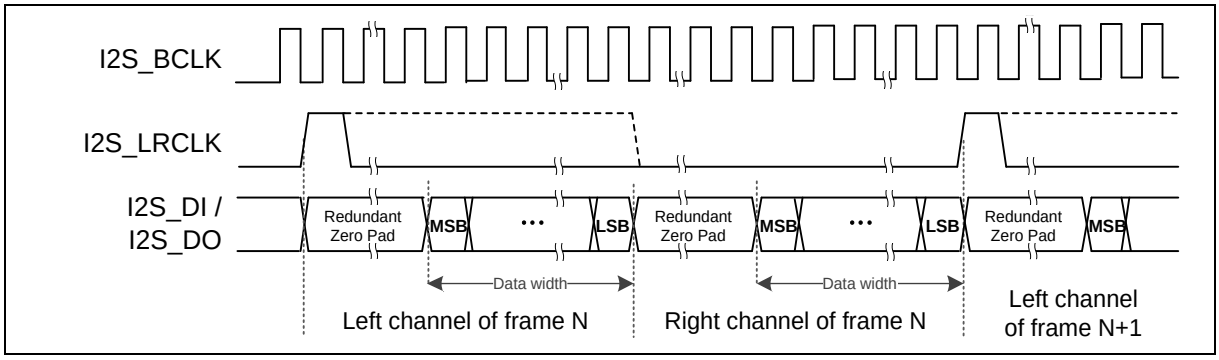


Figure 6.26-12 PCM with LSB Justified Data Format (FORMAT = 0x6 ; CHWIDTH > DATWIDTH)

6.26.5.4 TDM Multi-channel Transmission

The PCM mode in this I²S controller also supports TDM transmission. The Time Division Multiplexed (TDM) method allows multiple channels of audio data to be transmitted on a single data line. The TDM interface is similar to the 2-channel PCM audio interface with the exception that more audio channels are transmitted within a sample frame which is defined by a period of the I2S_LRCLK. The channel number of TDM interface is typically 4, 6, or 8 and it is selected by TDMCHNUM (I2S_CTL0[31:30]).

Same as previous I²S and PCM descriptions, each channel block is comprised of the audio data word followed by a sufficient number of zero data bits to complete one channel block. The bit-width of data word and channel block are defined by DATWIDTH (I2S_CTL0[5:4]) and CHWIDTH (I2S_CTL0[29:28]) respectively. Note that the TDM PCM mode supports 16-bit, 24-bit, 32-bit audio data word (excluding 8-bit data), and the hardware will set the bit-width of transmitting data as 16-bit if DATWIDTH (I2S_CTL0[5:4]) is 0x0. The pulse width of frame start signal is also selected by PCMSYNC (I2S_CTL0[27]).

The examples of 6-channel TDM transmission with 24-bit audio data in 32-bit channel block are shown in Figure 6.26-13. In 2-channel audio interface, the first and second audio channels are called as left-channel and right-channel (or channel0 and channel1). In TDM multi-channel application, the first and second audio channels are called as channel0 and channel1.

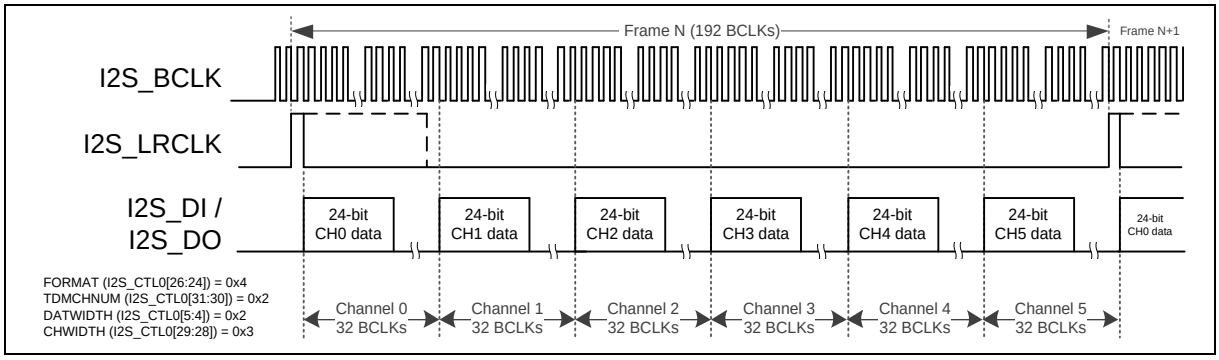


Figure 6.26-13 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM Standard Data Format; FORMAT=0x4)

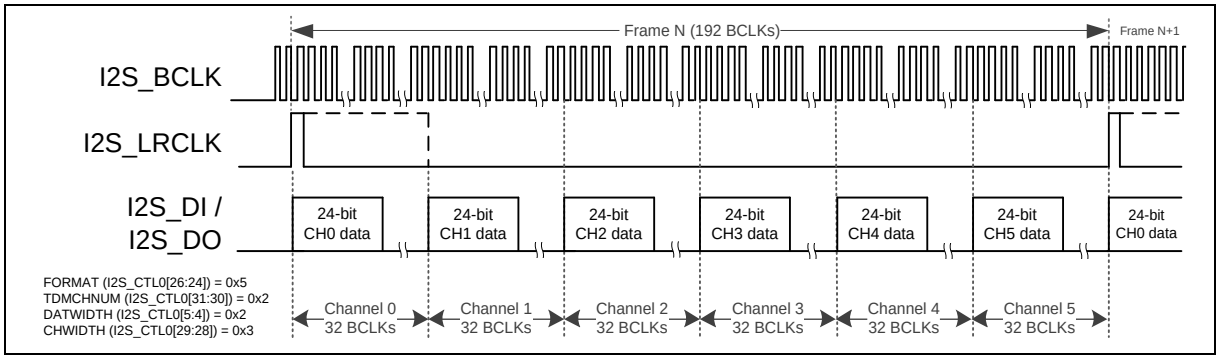


Figure 6.26-14 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM with MSB Justified; FORMAT=0x5)

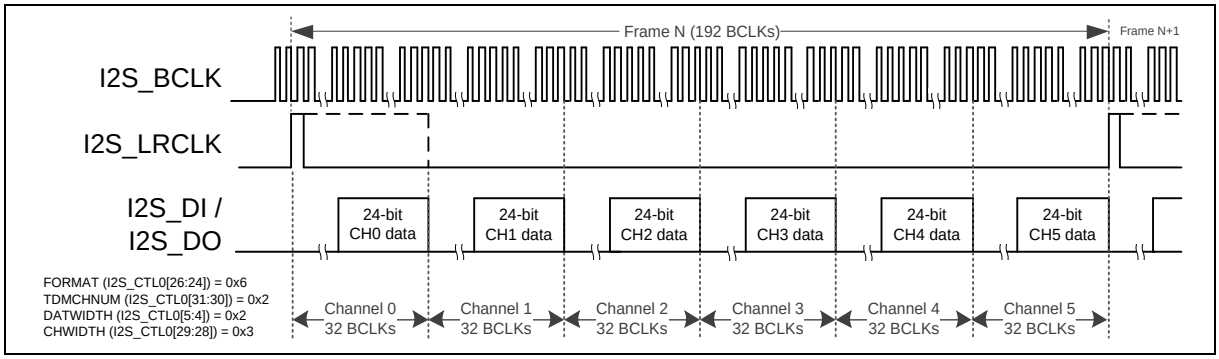


Figure 6.26-15 TDM 6-channel Audio Format with 24-bit Data in 32-bit Channel Block (PCM with LSB Justified; FORMAT=0x6)

6.26.5.5 Zero Crossing

When playing the audio by I²S controller, the output transmitting data comes from the memory by PDMA or by CPU. However, there may be some pop noise which induces the uncomfortable hearing if the playing sound volume is changed greatly by user. The zero-crossing event of audio data means the playing sound is relatively silent at the moment. Therefore, the zero-cross interrupt can be used for the indication of gain level adjustment in order to prevent the huge variance of sound volume.

If zero-cross detection of individual audio channel is enabled by the corresponding control bit from CH0ZCEN to CH7ZCEN (I2S_CTL1[0] to I2S_CTL1[7]), the hardware will detect the next transferring data word of the corresponding audio channel whether it is 0 or its MSB has been changed. If zero value or MSB (sign bit) changing of the transmitting audio data has been detected while zero-cross detection is enabled, the hardware will set the corresponding status bit from CH0ZCIF to CH7ZCIF (I2S_STATUS1[0] to I2S_STATUS1[7]) for the audio channel and then keep the output audio data silent (all data bit zero) automatically until the corresponding event status bit is cleared by software.

Therefore, if user wants to modify the audio playing gain, users can enable the zero crossing interrupt function, CH0ZCIEN to CH7ZCIEN (I2S_IEN[16:23]), to indicate the zero crossing time and to change the audio gain. This will reduce the pop noise.

6.26.5.6 PDMA Mode

The I²S function can use PDMA function for transmitting or receiving data access. If the PDMA function of transmitting data is enabled by TXPDMAEN (I2S_CTL0[20]), the I²S controller will generate the request signal and then get transmitting audio data from memory by PDMA IP automatically while TX FIFO is not full. If the PDMA function of receiving data is enabled by RXPDMAEN (I2S_CTL0[21]), the I²S controller will generate the request signal and then the receiving data will be moved into memory by PDMA hardware automatically while the RX FIFO is not empty. Therefore, using PDMA function will save the CPU loading to service other functions.

6.26.5.7 I²S Interrupt Sources

The I²S controller supports zero-cross interrupt of individual audio channel, transmit FIFO threshold level interrupt, transmit FIFO overflow interrupt and transmit FIFO underflow interrupt in transmit operation. In receive operation, it supports receive FIFO threshold level interrupt, receive FIFO overflow interrupt and receive FIFO underflow interrupt. When I²S interrupt occurs, user can check I2STXINT (I2S_STATUS0[2]) and I2SRXINT (I2S_STATUS0[1]) flags to recognize the interrupt sources.

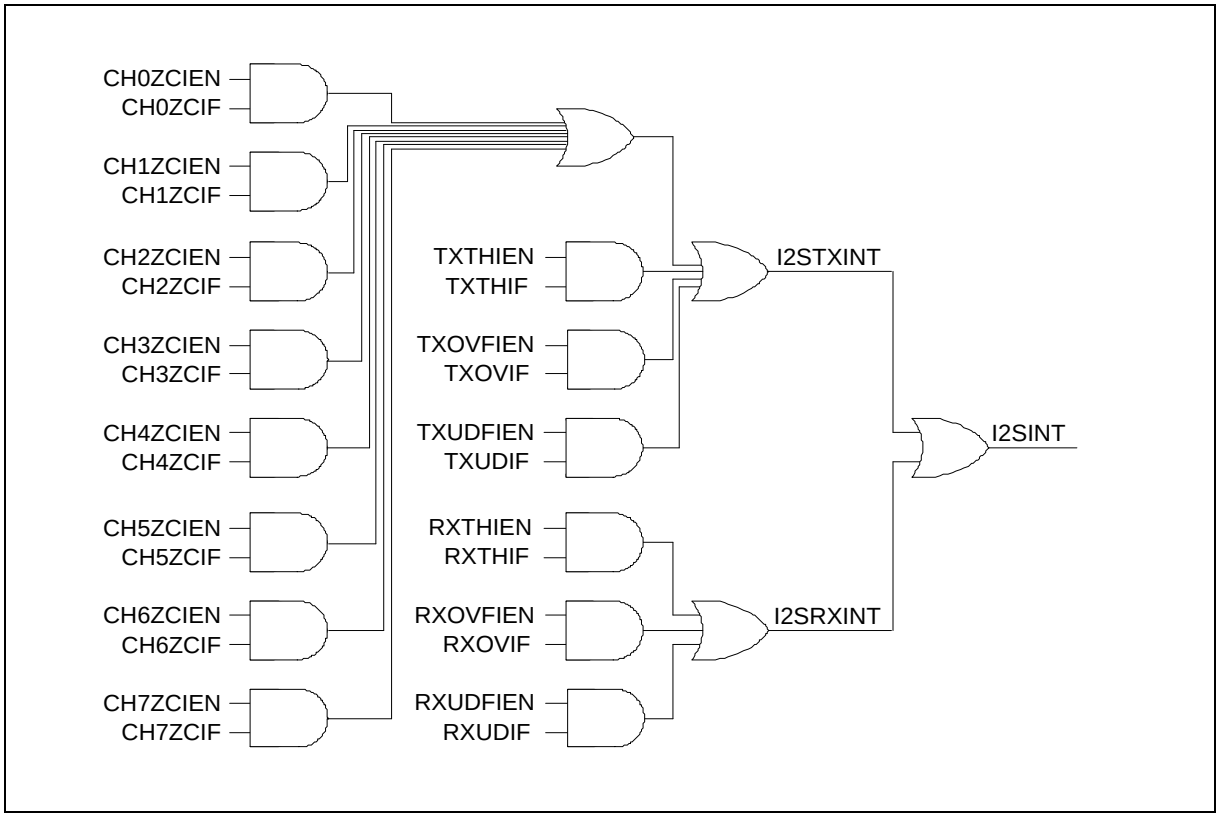


Figure 6.26-16 I²S Interrupts

6.26.5.8 FIFO Operation

In 2-channel I²S or PCM protocol, the bit-width of audio data in a channel block can be 8, 16, 24, or 32 bits. The memory arrangements of audio data for various settings are shown in Figure 6.26-17.

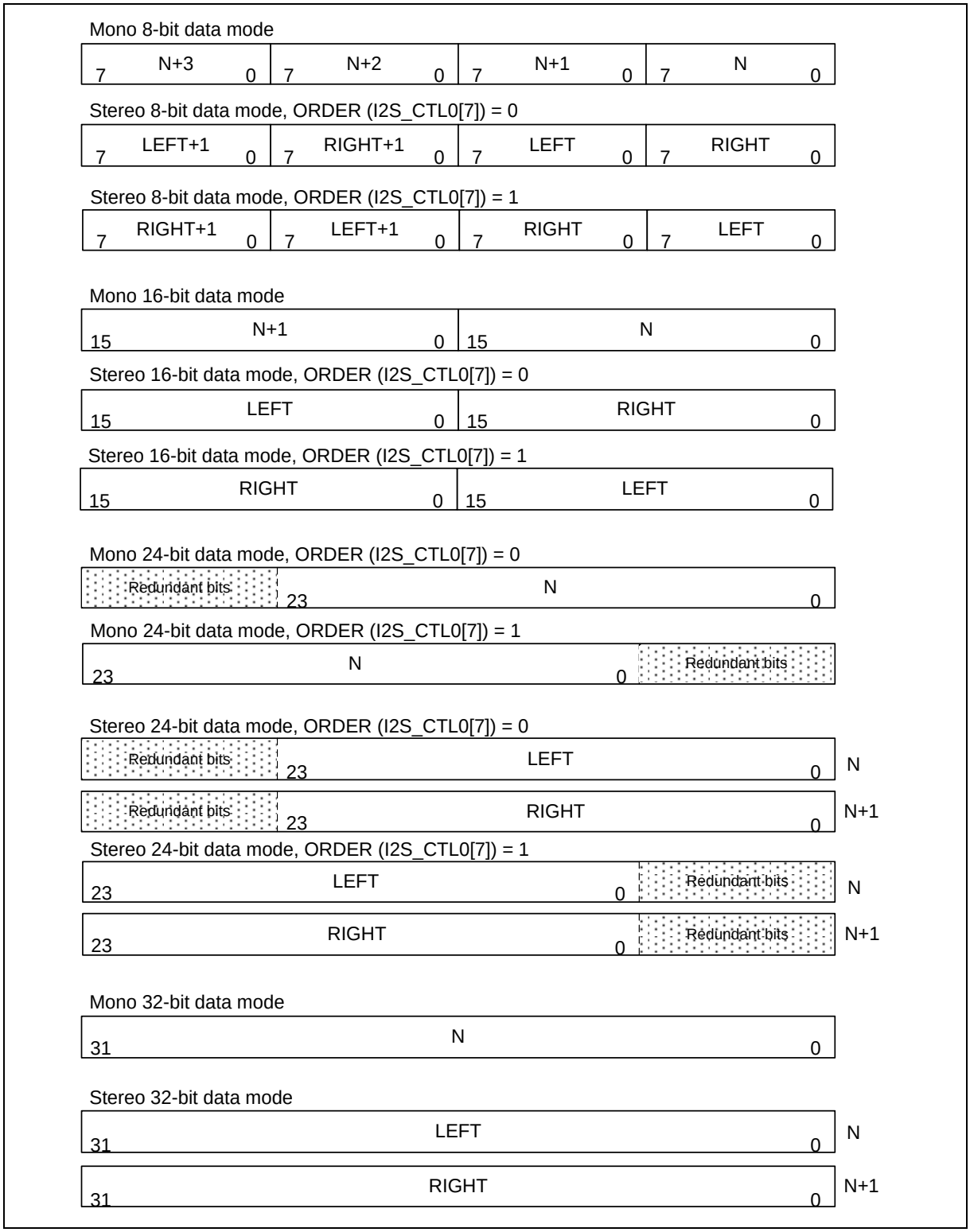


Figure 6.26-17 FIFO Contents for Various 2-channel Audio Modes

In 4-channel TDM PCM data format, the bit-width of audio data in a channel block can be 16, 24, or 32 bits. The memory arrangements of audio data for various settings are shown in Figure 6.26-18.

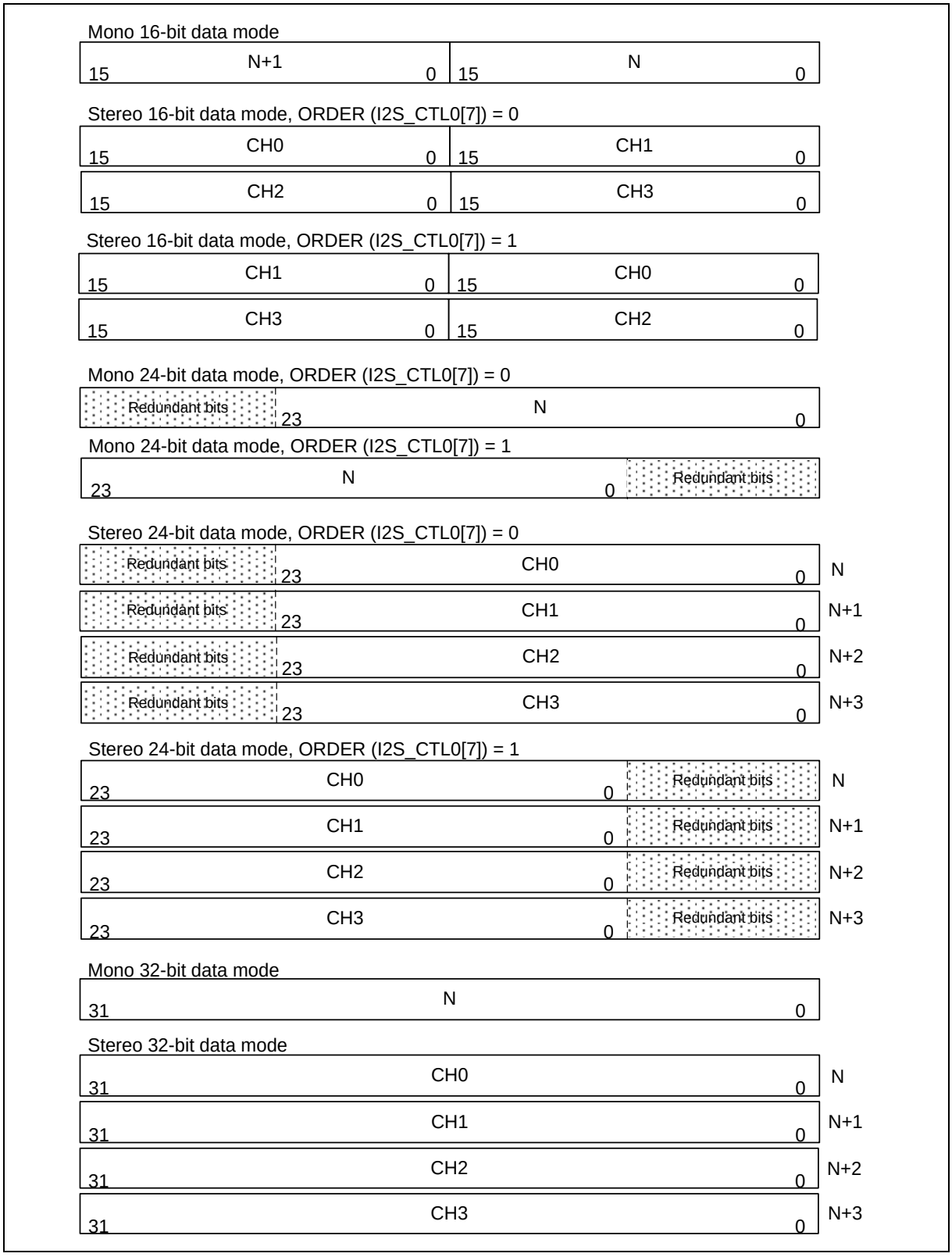


Figure 6.26-18 FIFO Contents for Various 4-channel Audio Modes

In 6-channel TDM PCM data format, the bit-width of audio data in a channel block can be 16, 24, or 32 bits. The memory arrangements of audio data for various settings are shown Figure 6.26-19. In 16-bit

audio data transmission, ORDER (I2S_CTL0[7]) can be used to swap the audio data of even and odd channels which are stored in transmitting and receiving FIFO. In 24-bit audio data transmission, ORDER (I2S_CTL0[7]) can be also used to select the left-alignment or right-alignment formula of audio data which is stored in 32-bit FIFO entries.

The FIFO content of 8-channel TDM PCM data format is similar to 6-channel and it can be analogized easily.

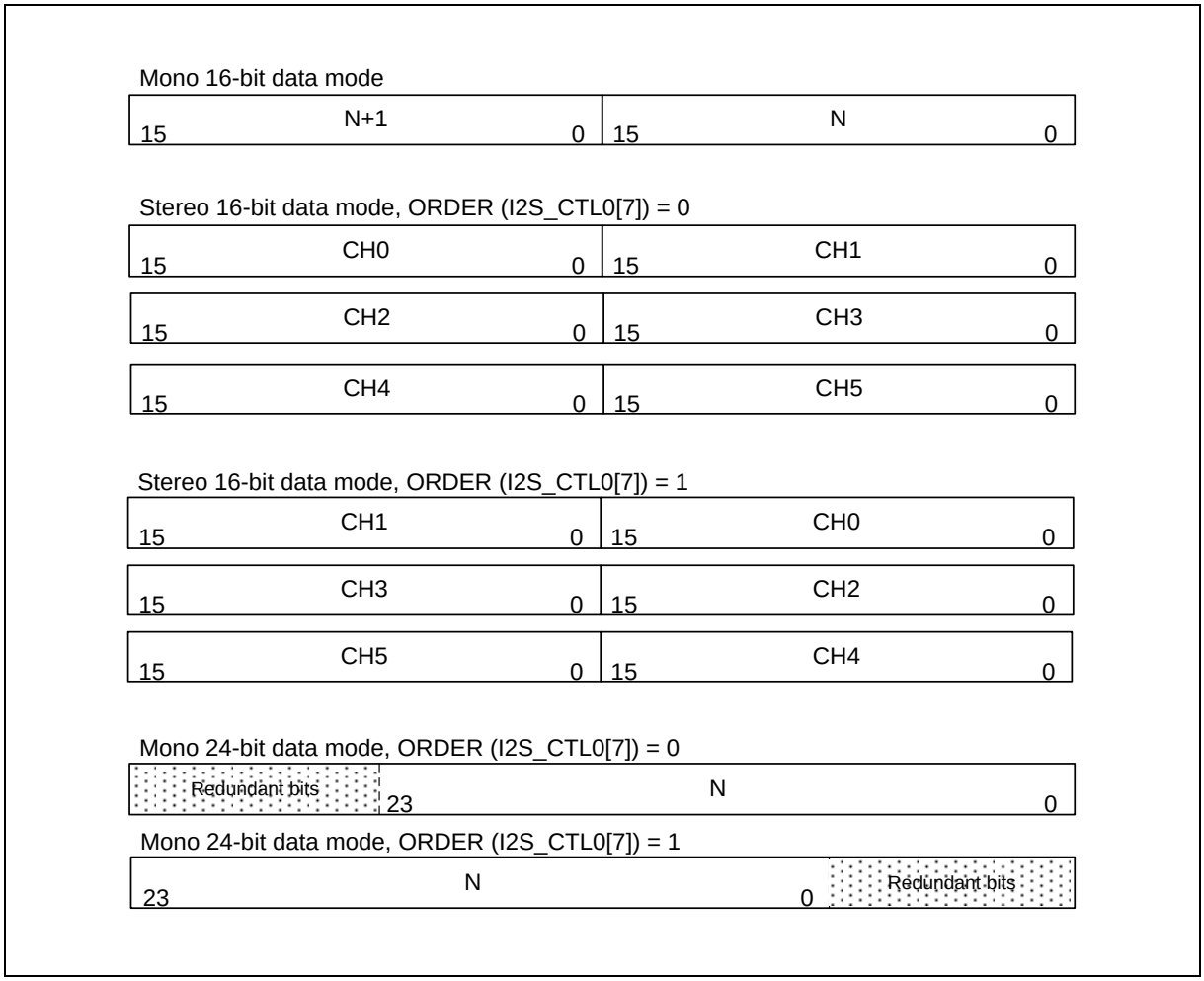


Figure 6.26-19 FIFO Contents for Various 6-channel Audio Modes (Part-1)

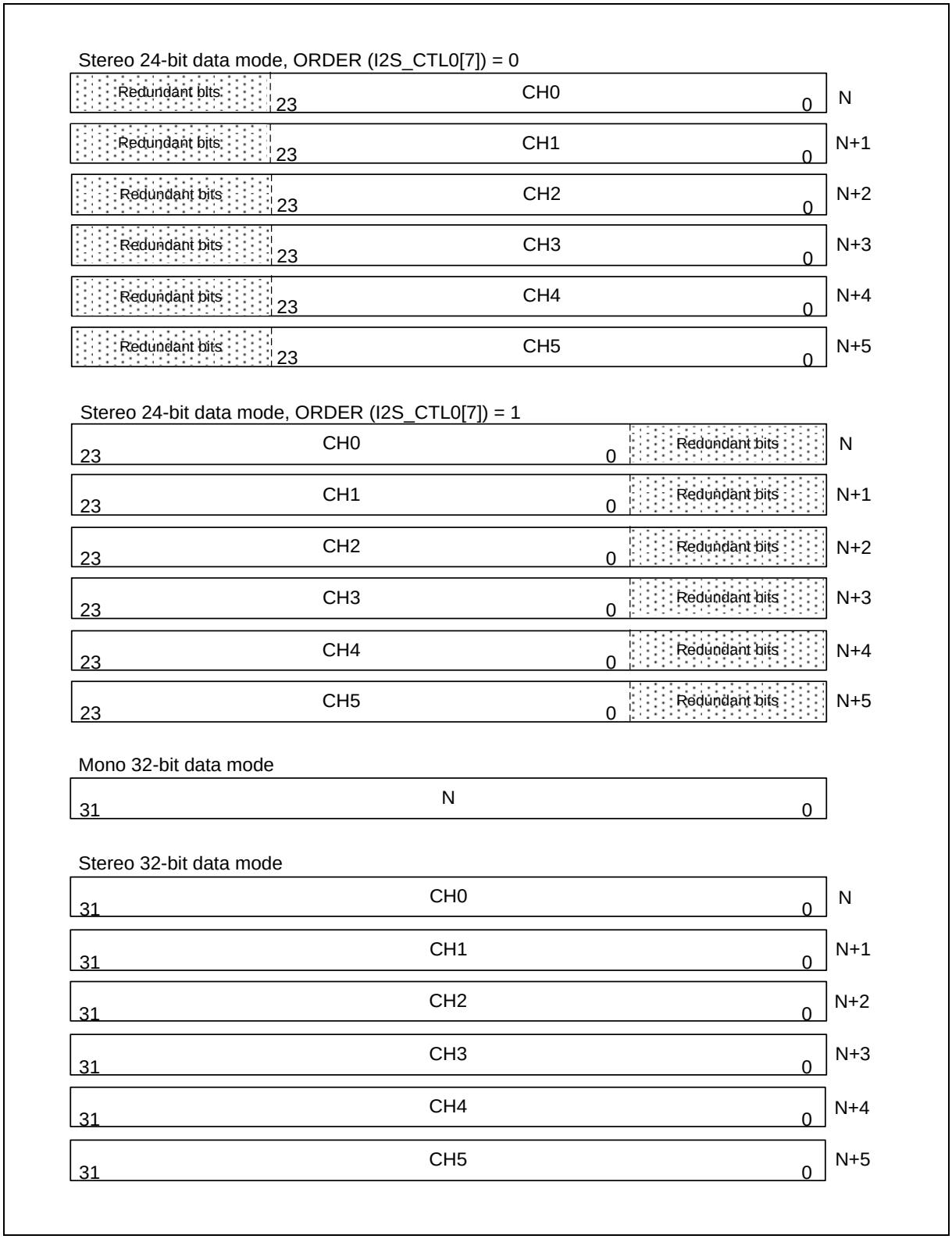


Figure 6.26-20 FIFO Contents for Various 6-channel Audio Modes (Part-2)

6.26.6 Register Map

R: *Read only,* ***W:*** *Write only,* ***R/W:*** *Both read and write*

Register	Offset	R/W	Description	Reset Value
I²S Base Address **I2Sx_BA = 0x4004_8000 + (0x0000_1000 * x)** **x=0, 1**				
I2Sx_CTL0	I2Sx_BA+0x00	R/W	I²S Control Register 0	0x0000_0000
I2Sx_CTL1	I2Sx_BA+0x20	R/W	I²S Control Register 1	0x0000_0000
I2Sx_CLKDIV	I2Sx_BA+0x04	R/W	I²S Clock Divider Register	0x0000_0000
I2Sx_IEN	I2Sx_BA+0x08	R/W	I²S Interrupt Enable Register	0x0000_0000
I2Sx_STATUS0	I2Sx_BA+0x0C	R/W	I²S Status Register 0	0x0014_1038
I2Sx_STATUS1	I2Sx_BA+0x24	R/W	I²S Status Register 1	0x0000_0000
I2Sx_TXFIFO	I2Sx_BA+0x10	W	I²S Transmit FIFO Register	0x0000_0000
I2Sx_RXFIFO	I2Sx_BA+0x14	R	I²S Receive FIFO Register	0x0000_0000

6.26.7 Register Description

I²S Control Register 0 (I2Sx_CTL0)

Register	Offset	R/W	Description	Reset Value
I2Sx_CTL0	I2Sx_BA+0x00	R/W	I²S Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
TDMCHNUM		CHWIDTH		PCMSYNC	FORMAT		
23	22	21	20	19	18	17	16
RXLCH	Reserved	RXPDMAEN	TXPDMAEN	RXFBCLR	TXFBCLR	Reserved	
15	14	13	12	11	10	9	8
MCLKEN	Reserved						SLAVE
7	6	5	4	3	2	1	0
ORDER	MONO	DATWIDTH		MUTE	RXEN	TXEN	I2SEN

Bits	Description	
[31:30]	**TDMCHNUM**	**TDM Channel Number** This bit fields are used to define the TDM channel number in one audio frame while PCM mode (FORMAT[2] = 1). 00 = 2 channels in audio frame. 01 = 4 channels in audio frame. 10 = 6 channels in audio frame. 11 = 8 channels in audio frame.
[29:28]	**CHWIDTH**	**Channel Width** This bit fields are used to define the length of audio channel. If CHWIDTH < DATWIDTH, the hardware will set the real channel length as the bit-width of audio data which is defined by DATWIDTH. 00 = The bit-width of each audio channel is 8-bit. 01 = The bit-width of each audio channel is 16-bit. 10 = The bit-width of each audio channel is 24-bit. 11 = The bit-width of each audio channel is 32-bit.
[27]	**PCMSYNC**	**PCM Synchronization Pulse Length Selection** This bit field is used to select the high pulse length of frame synchronization signal in PCM protocol 0 = One BCLK period. 1 = One channel period. **Note:** This bit is only available in master mode

[26:24]	**FORMAT**	**Data Format Selection** 000 = I²S standard data format. 001 = I²S with MSB justified. 010 = I²S with LSB justified. 011 = Reserved. 100 = PCM standard data format. 101 = PCM with MSB justified. 110 = PCM with LSB justified. 111 = Reserved.
[23]	**RXLCH**	**Receive Left Channel Enable Bit** When monaural format is selected (MONO = 1), I²S will receive channel1 data if RXLCH is set to 0, and receive channel0 data if RXLCH is set to 1. 0 = Receive channel1 data in MONO mode. 1 = Receive channel0 data in MONO mode.
[22]	**Reserved**	Reserved.
[21]	**RXPDMAEN**	**Receive PDMA Enable Bit** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[20]	**TXPDMAEN**	**Transmit PDMA Enable Bit** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled.
[19]	**RXFBCLR**	**Receive FIFO Buffer Clear** 0 = No Effect. 1 = Clear RX FIFO. **Note 1:** Write 1 to clear receive FIFO, internal pointer is reset to FIFO start point, and RXCNT (I2S_STATUS1[20:16]) returns 0 and receive FIFO becomes empty. **Note 2:** This bit is cleared by hardware automatically, read it return 0.
[18]	**TXFBCLR**	**Transmit FIFO Buffer Clear** 0 = No Effect. 1 = Clear TX FIFO. **Note 1:** Write 1 to clear transmit FIFO, internal pointer is reset to FIFO start point, and TXCNT (I2S_STATUS1[12:8]) returns 0 and transmit FIFO becomes empty but data in transmit FIFO is not changed. **Note 2:** This bit is clear by hardware automatically, read it return 0.
[17:16]	**Reserved**	Reserved.
[15]	**MCLKEN**	**Master Clock Enable Bit** If MCLKEN is set to 1, I²S controller will generate master clock on I2S_MCLK pin for external audio devices. 0 = Master clock Disabled. 1 = Master clock Enabled.
[14:9]	**Reserved**	Reserved.

[8]	**SLAVE**	**Slave Mode Enable Bit** 0 = Master mode. 1 = Slave mode. **Note:** I²S can operate as master or slave. For Master mode, I2S_BCLK and I2S_LRCLK pins are output mode and send out bit clock to Audio CODEC chip. In Slave mode, I2S_BCLK and I2S_LRCLK pins are input mode and I2S_BCLK and I2S_LRCLK signals are received from outer Audio CODEC chip.
[7]	**ORDER**	**Stereo Data Order in FIFO** In 8-bit/16-bit data width, this bit is used to select whether the even or odd channel data is stored in higher byte. In 24-bit data width, this is used to select the left/right alignment method of audio data which is stored in data memory consisted of 32-bit FIFO entries. 0 = Even channel data at high byte in 8-bit/16-bit data width. LSB of 24-bit audio data in each channel is aligned to right side in 32-bit FIFO entries. 1 = Even channel data at low byte in 8-bit/16-bit data width. MSB of 24-bit audio data in each channel is aligned to left side in 32-bit FIFO entries.
[6]	**MONO**	**Monaural Data Control** 0 = Data is stereo format. 1 = Data is monaural format. **Note:** When chip records data, RXLCH (I2S_CTL0[23]) indicates which channel data will be saved if monaural format is selected.
[5:4]	**DATWIDTH**	**Data Width** This bit field is used to define the bit-width of data word in each audio channel 00 = The bit-width of data word is 8-bit. 01 = The bit-width of data word is 16-bit. 10 = The bit-width of data word is 24-bit. 11 = The bit-width of data word is 32-bit.
[3]	**MUTE**	**Transmit Mute Enable Bit** 0 = Transmit data is shifted from buffer. 1 = Send zero on transmit channel.
[2]	**RXEN**	**Receive Enable Bit** 0 = Data receiving Disabled. 1 = Data receiving Enabled.
[1]	**TXEN**	**Transmit Enable Bit** 0 = Data transmission Disabled. 1 = Data transmission Enabled.
[0]	**I2SEN**	**I²S Controller Enable Bit** 0 = I²S controller Disabled. 1 = I²S controller Enabled.

I²S Control Register 1 (I2Sx_CTL1)

Register	Offset	R/W	Description	Reset Value
I2Sx_CTL1	I2Sx_BA+0x20	R/W	I²S Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						PB16ORD	PBWIDTH
23	22	21	20	19	18	17	16
Reserved				RXTH			
15	14	13	12	11	10	9	8
Reserved				TXTH			
7	6	5	4	3	2	1	0
CH7ZCEN	CH6ZCEN	CH5ZCEN	CH4ZCEN	CH3ZCEN	CH2ZCEN	CH1ZCEN	CH0ZCEN

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**PB16ORD**	**FIFO Read/Write Order in 16-bit Width of Peripheral Bus** When PBWIDTH = 1, the data FIFO will be increased or decreased by two peripheral bus access. This bit is used to select the order of FIFO access operations to meet the 32-bit transmitting/receiving FIFO entries. 0 = Low 16-bit read/write access first. 1 = High 16-bit read/write access first. **Note:** This bit is available while PBWIDTH = 1.
[24]	**PBWIDTH**	**Peripheral Bus Data Width Selection** This bit is used to choice the available data width of APB bus. It must be set to 1 while PDMA function is enable and it is set to 16-bit transmission mode 0 = 32 bits data width. 1 = 16 bits data width. **Note 1:** If PBWIDTH=1, the low 16 bits of 32-bit data bus are available. **Note 2:** If PBWIDTH=1, the transmitting FIFO level will be increased after two FIFO write operations. **Note 3:** If PBWIDTH=1, the receiving FIFO level will be decreased after two FIFO read operations.
[23:20]	**Reserved**	Reserved.
[19:16]	**RXTH**	**Receive FIFO Threshold Level** 0000 = 1 data word in receive FIFO. 0001 = 2 data words in receive FIFO. 0010 = 3 data words in receive FIFO. …. 1110 = 15 data words in receive FIFO. 1111 = 16 data words in receive FIFO. **Note:** When received data word number in receive buffer is larger than threshold level then RXTHIF (I2S_STATUS0[10]) flag is set.

[15:12]	**Reserved**	Reserved.
[11:8]	**TXTH**	**Transmit FIFO Threshold Level** 0000 = 0 data word in transmit FIFO. 0001 = 1 data word in transmit FIFO. 0010 = 2 data words in transmit FIFO. .... 1110 = 14 data words in transmit FIFO. 1111 = 15 data words in transmit FIFO. **Note:** If remain data word number in transmit FIFO is less than or equal to threshold level then TXTHIF (I2S_STATUS0[18]) flag is set.
[7]	**CH7ZCEN**	**Channel7 Zero-cross Detect Enable Bit** 0 = channel7 zero-cross detect Disabled. 1 = channel7 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3. **Note 2:** If this bit is set to 1, when channel7 data sign bit change or next shift data bits are all 0 then CH7ZCIF (I2S_STATUS1[7]) flag is set to 1. **Note 3:** If CH7ZCIF flag is set to 1, the channel7 will be mute.
[6]	**CH6ZCEN**	**Channel6 Zero-cross Detect Enable Bit** 0 = channel6 zero-cross detect Disabled. 1 = channel6 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3. **Note 2:** If this bit is set to 1, when channel6 data sign bit change or next shift data bits are all 0 then CH6ZCIF(I2S_STATUS1[6]) flag is set to 1. **Note 3:** If CH6ZCIF flag is set to 1, the channel6 will be mute.
[5]	**CH5ZCEN**	**Channel5 Zero-cross Detect Enable Bit** 0 = channel5 zero-cross detect Disabled. 1 = channel5 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3. **Note 2:** If this bit is set to 1, when channel5 data sign bit change or next shift data bits are all 0 then CH5ZCIF(I2S_STATUS1[5]) flag is set to 1. **Note 3:** If CH5ZCIF flag is set to 1, the channel5 will be mute.
[4]	**CH4ZCEN**	**Channel4 Zero-cross Detect Enable Bit** 0 = channel4 zero-cross detect Disabled. 1 = channel4 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3. **Note 2:** If this bit is set to 1, when channel4 data sign bit change or next shift data bits are all 0 then CH4ZCIF(I2S_STATUS1[4]) flag is set to 1. **Note 3:** If CH4ZCIF flag is set to 1, the channel4 will be mute.
[3]	**CH3ZCEN**	**Channel3 Zero-cross Detect Enable Bit** 0 = channel3 zero-cross detect Disabled. 1 = channel3 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3. **Note 2:** If this bit is set to 1, when channel3 data sign bit change or next shift data bits are all 0 then CH3ZCIF(I2S_STATUS1[3]) flag is set to 1. **Note 3:** If CH3ZCIF flag is set to 1, the channel3 will be mute.

[2]	**CH2ZCEN**	**Channel2 Zero-cross Detect Enable Bit** 0 = channel2 zero-cross detect Disabled. 1 = channel2 zero-cross detect Enabled. **Note 1:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3. **Note 2:** If this bit is set to 1, when channel2 data sign bit change or next shift data bits are all 0 then CH2ZCIF(I2S_STATUS1[2]) flag is set to 1. **Note 3:** If CH2ZCIF flag is set to 1, the channel2 will be mute.
[1]	**CH1ZCEN**	**Channel1 Zero-cross Detect Enable Bit** 0 = channel1 zero-cross detect Disabled. 1 = channel1 zero-cross detect Enabled. **Note 1:** Channel1 also means right audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode. **Note 2:** If this bit is set to 1, when channel1 data sign bit change or next shift data bits are all 0 then CH1ZCIF(I2S_STATUS1[1]) flag is set to 1. **Note 3:** If CH1ZCIF flag is set to 1, the channel1 will be mute.
[0]	**CH0ZCEN**	**Channel0 Zero-cross Detection Enable Bit** 0 = channel0 zero-cross detect Disabled. 1 = channel0 zero-cross detect Enabled. **Note 1:** Channel0 also means left audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode. **Note 2:** If this bit is set to 1, when channel0 data sign bit change or next shift data bits are all 0 then CH0ZCIF(I2S_STATUS1[0]) flag is set to 1. **Note 3:** If CH0ZCIF flag is set to 1, the channel0 will be mute.

I²S Clock Divider (I2Sx_CLKDIV)

Register	Offset	R/W	Description	Reset Value
I2Sx_CLKDIV	I2Sx_BA+0x04	R/W	I²S Clock Divider Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**BCLKDIV**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BCLKDIV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**MCLKDIV**						

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:8]	**BCLKDIV**	**Bit Clock Divider** The I²S controller will generate bit clock in Master mode. Software can program these bit fields to generate sampling rate clock frequency. F_BCLK= F_I2SCLK / (2*(BCLKDIV + 1)). **Note:** F_BCLK is the frequency of BCLK and F_I2SCLK is the frequency of I2S_CLK
[7]	**Reserved**	Reserved.
[6:0]	**MCLKDIV**	**Master Clock Divider** If chip external crystal frequency is (2xMCLKDIV)*256fs then software can program these bits to generate 256fs clock frequency to audio codec chip. If MCLKDIV is set to 0, MCLK is the same as external clock input. For example, sampling rate is 24 kHz and chip external crystal clock is 12.288 MHz, set MCLKDIV = 1. F_MCLK = F_I2SCLK/(2x(MCLKDIV)) (When MCLKDIV is >= 1). F_MCLK = F_I2SCLK (When MCLKDIV is set to 0). **Note:** F_MCLK is the frequency of MCLK, and F_I2SCLK is the frequency of the I2S_CLK

I²S Interrupt Enable Register (I2Sx_IEN)

Register	Offset	R/W	Description	Reset Value
I2Sx_IEN	I2Sx_BA+0x08	R/W	I²S Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CH7ZCIEN	CH6ZCIEN	CH5ZCIEN	CH4ZCIEN	CH3ZCIEN	CH2ZCIEN	CH1ZCIEN	CH0ZCIEN
15	14	13	12	11	10	9	8
Reserved					TXTHIEN	TXOVFIEN	TXUDFIEN
7	6	5	4	3	2	1	0
Reserved					RXTHIEN	RXOVFIEN	RXUDFIEN

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**CH7ZCIEN**	**Channel7 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel7 zero-cross **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3.
[22]	**CH6ZCIEN**	**Channel6 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel6 zero-cross **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3.
[21]	**CH5ZCIEN**	**Channel5 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel5 zero-cross **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3.
[20]	**CH4ZCIEN**	**Channel4 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel4 zero-cross **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3.

[19]	**CH3ZCIEN**	**Channel3 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel3 zero-cross **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3.
[18]	**CH2ZCIEN**	**Channel2 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel2 zero-cross **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3.
[17]	**CH1ZCIEN**	**Channel1 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel1 zero-cross **Note 2:** Channel1 also means right audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode.
[16]	**CH0ZCIEN**	**Channel0 Zero-cross Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note 1:** Interrupt occurs if this bit is set to 1 and channel0 zero-cross **Note 2:** Channel0 also means left audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode.
[15:11]	**Reserved**	Reserved.
[10]	**TXTHIEN**	**Transmit FIFO Threshold Level Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and data words in transmit FIFO is less than or equal to TXTH (I2S_CTL1[11:8]).
[9]	**TXOVFIEN**	**Transmit FIFO Overflow Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and TXOVIF (I2S_STATUS0[17]) flag is set to 1
[8]	**TXUDFIEN**	**Transmit FIFO Underflow Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occur if this bit is set to 1 and TXUDIF (I2S_STATUS0[16]) flag is set to 1.
[7:3]	**Reserved**	Reserved.
[2]	**RXTHIEN**	**Receive FIFO Threshold Level Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and data words in receive FIFO is larger than RXTH (I2S_CTL1[19:16]).

[1]	**RXOVFIEN**	**Receive FIFO Overflow Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Interrupt occurs if this bit is set to 1 and RXOVIF (I2S_STATUS0[9]) flag is set to 1
[0]	**RXUDFIEN**	**Receive FIFO Underflow Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** If software reads receive FIFO when it is empty then RXUDIF (I2S_STATUS0[8]) flag is set to 1. If RXUDFIEN bit is enabled, interrupt occurs.

I²S Status Register 0 (I2Sx_STATUS0)

Register	Offset	R/W	Description	Reset Value
I2Sx_STATUS0	I2Sx_BA+0x0C	R/W	I²S Status Register 0	0x0014_1038

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		TXBUSY	TXEMPTY	TXFULL	TXTHIF	TXOVIF	TXUDIF
15	14	13	12	11	10	9	8
Reserved			RXEMPTY	RXFULL	RXTHIF	RXOVIF	RXUDIF
7	6	5	4	3	2	1	0
Reserved		DATACH			I2STXINT	I2SRXINT	I2SINT

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**TXBUSY**	**Transmit Busy (Read Only)** 0 = Transmit shift buffer is empty. 1 = Transmit shift buffer is busy. **Note:** This bit is cleared to 0 when all data in transmit FIFO and shift buffer is shifted out. And set to 1 when 1st data is load to shift buffer.
[20]	**TXEMPTY**	**Transmit FIFO Empty (Read Only)** This bit reflect data word number in transmit FIFO is 0 0 = Not empty. 1 = Empty.
[19]	**TXFULL**	**Transmit FIFO Full (Read Only)** This bit reflect data word number in transmit FIFO is 16 0 = Not full. 1 = Full.
[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = Data word(s) in FIFO is larger than threshold level. 1 = Data word(s) in FIFO is less than or equal to threshold level. **Note:** When data word(s) in transmit FIFO is less than or equal to threshold value set in TXTH (I2S_CTL1[11:8]) the TXTHIF bit becomes to 1. It keeps at 1 till TXCNT (I2S_STATUS1[12:8]) is larger than TXTH (I2S_CTL1[11:8]) after software write TXFIFO register.
[17]	**TXOVIF**	**Transmit FIFO Overflow Interrupt Flag** 0 = No overflow. 1 = Overflow. **Note 1:** Write data to transmit FIFO when it is full and this bit set to 1 **Note 2:** Write 1 to clear this bit to 0.

[16]	**TXUDIF**	**Transmit FIFO Underflow Interrupt Flag** 0 = No underflow. 1 = Underflow. **Note 1:** This bit will be set to 1 when shift logic hardware read data from transmitting FIFO and the filling data level in transmitting FIFO is not enough for one audio frame. **Note 2:** Write 1 to clear this bit to 0.
[15:13]	**Reserved**	Reserved.
[12]	**RXEMPTY**	**Receive FIFO Empty (Read Only)** 0 = Not empty. 1 = Empty. **Note:** This bit reflects data words number in receive FIFO is 0
[11]	**RXFULL**	**Receive FIFO Full (Read Only)** 0 = Not full. 1 = Full. **Note:** This bit reflects data words number in receive FIFO is 16.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = Data word(s) in FIFO is less than or equal to threshold level. 1 = Data word(s) in FIFO is larger than threshold level. **Note:** When data word(s) in receive FIFO is larger than threshold value set in RXTH (I2S_CTL1[19:16]) the RXTHIF bit becomes to 1. It keeps at 1 till RXCNT (I2S_STATUS1[20:16]) is less than or equal to RXTH (I2S_CTL1[19:16]) after software read RXFIFO register.
[9]	**RXOVIF**	**Receive FIFO Overflow Interrupt Flag** 0 = No overflow occur. 1 = Overflow occur. **Note 1:** When receive FIFO is full and receive hardware attempt to write data into receive FIFO then this bit is set to 1, data in 1st buffer is overwritten. **Note 2:** Write 1 to clear this bit to 0.
[8]	**RXUDIF**	**Receive FIFO Underflow Interrupt Flag** 0 = No underflow occur. 1 = Underflow occur. **Note 1:** When receive FIFO is empty, and software reads the receive FIFO again. This bit will be set to 1, and it indicates underflow situation occurs. **Note 2:** Write 1 to clear this bit to 0
[7:6]	**Reserved**	Reserved.
[5:3]	**DATACH**	**Transmission Data Channel (Read Only)** This bit fields are used to indicate which audio channel is current transmit data belong. 000 = channel0 (means left channel while 2-channel I2S/PCM mode). 001 = channel1 (means right channel while 2-channel I2S/PCM mode). 010 = channel2 (available while 4-channel TDM PCM mode). 011 = channel3 (available while 4-channel TDM PCM mode). 100 = channel4 (available while 6-channel TDM PCM mode). 101 = channel5 (available while 6-channel TDM PCM mode). 110 = channel6 (available while 8-channel TDM PCM mode). 111 = channel7 (available while 8-channel TDM PCM mode).

[2]	**I2STXINT**	**I²S Transmit Interrupt (Read Only)** 0 = No transmit interrupt. 1 = Transmit interrupt.
[1]	**I2SRXINT**	**I²S Receive Interrupt (Read Only)** 0 = No receive interrupt. 1 = Receive interrupt.
[0]	**I2SINT**	**I²S Interrupt Flag (Read Only)** 0 = No I²S interrupt. 1 = I²S interrupt. **Note:** It is wire-OR of I2STXINT and I2SRXINT bits.

I²S Status Register 1 (I2Sx_STATUS1)

Register	Offset	R/W	Description	Reset Value
I2Sx_STATUS1	I2Sx_BA+0x24	R/W	I²S Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			RXCNT				
15	14	13	12	11	10	9	8
Reserved			TXCNT				
7	6	5	4	3	2	1	0
CH7ZCIF	CH6ZCIF	CH5ZCIF	CH4ZCIF	CH3ZCIF	CH2ZCIF	CH1ZCIF	CH0ZCIF

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:16]	**RXCNT**	**Receive FIFO Level (Read Only)** These bits indicate the number of available entries in receive FIFO 00000 = No data. 00001 = 1 word in receive FIFO. 00010 = 2 words in receive FIFO. …. 01110 = 14 words in receive FIFO. 01111 = 15 words in receive FIFO. 10000 = 16 words in receive FIFO. Others are reserved.
[15:13]	**Reserved**	Reserved.
[12:8]	**TXCNT**	**Transmit FIFO Level (Read Only)** These bits indicate the number of available entries in transmit FIFO 00000 = No data. 00001 = 1 word in transmit FIFO. 00010 = 2 words in transmit FIFO. …. 01110 = 14 words in transmit FIFO. 01111 = 15 words in transmit FIFO. 10000 = 16 words in transmit FIFO. Others are reserved.

[7]	**CH7ZCIF**	**Channel7 Zero-cross Interrupt Flag** It indicates channel7 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel7. 1 = Channel7 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3.
[6]	**CH6ZCIF**	**Channel6 Zero-cross Interrupt Flag** It indicates channel6 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel6. 1 = Channel6 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x3.
[5]	**CH5ZCIF**	**Channel5 Zero-cross Interrupt Flag** It indicates channel5 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel5. 1 = Channel5 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3.
[4]	**CH4ZCIF**	**Channel4 Zero-cross Interrupt Flag** It indicates channel4 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel4. 1 = Channel4 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x2, 0x3.
[3]	**CH3ZCIF**	**Channel3 Zero-cross Interrupt Flag** It indicates channel3 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel3. 1 = Channel3 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3.
[2]	**CH2ZCIF**	**Channel2 Zero-cross Interrupt Flag** It indicates channel2 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel2. 1 = Channel2 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** This bit is available while multi-channel PCM mode and TDMCHNUM (I2S_CTL0[31:30]) = 0x1, 0x2, 0x3.

[1]	**CH1ZCIF**	**Channel1 Zero-cross Interrupt Flag** It indicates channel1 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel1. 1 = Channel1 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** Channel1 also means right audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode.
[0]	**CH0ZCIF**	**Channel0 Zero-cross Interrupt Flag** It indicates channel0 next sample data sign bit is changed or all data bits are 0. 0 = No zero-cross in channel0. 1 = Channel0 zero-cross is detected. **Note 1:** Write 1 to clear this bit to 0. **Note 2:** Channel0 also means left audio channel while I^2S (FORMAT[2]=0) or 2-channel PCM mode.

I²S Transmit FIFO (I2Sx_TXFIFO)

Register	Offset	R/W	Description	Reset Value
I2Sx_TXFIFO	I2Sx_BA+0x10	W	I²S Transmit FIFO Register	0x0000_0000

31	30	29	28	27	26	25	24
TXFIFO							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TXFIFO							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXFIFO							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXFIFO							

Bits	Description	
[31:0]	**TXFIFO**	**Transmit FIFO Bits** I²S contains 16 words (16x32 bits) data buffer for data transmit. Write data to this register to prepare data for transmit. The remaining word number is indicated by TXCNT (I2S_STATUS1[12:8]).

I²S Receive FIFO (I2Sx_RXFIFO)

Register	Offset	R/W	Description	Reset Value
I2Sx_RXFIFO	I2Sx_BA+0x14	R	I²S Receive FIFO Register	0x0000_0000

31	30	29	28	27	26	25	24
RXFIFO							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RXFIFO							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXFIFO							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXFIFO							

Bits	Description	
[31:0]	**RXFIFO**	**Receive FIFO Bits** I²S contains 16 words (16x32 bits) data buffer for data receive. Read this register to get data in FIFO. The remaining data word number is indicated by RXCNT (I2S_STATUS1[20:16]).

6.27 Serial Peripheral Interface (SPI)

6.27.1 Overview

The Serial Peripheral Interface (SPI) applies to synchronous serial data communication and allows full duplex transfer. Devices communicate in Master/Slave mode with the 4-wire bi-direction interface. The chip contains up to four sets of SPI controllers performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device. Each SPI controller can be configured as a master or a slave device and supports the PDMA function to access the data buffer. Each SPI controller also supports I^2S mode to connect external audio CODEC.

6.27.2 Features

- SPI Mode
  - Up to four sets of SPI controllers
  - Supports Master or Slave mode operation
  - Master mode up to 100 MHz (when chip works at V_{DD} = 2.7~3.6V)
  - Slave mode up to 50 MHz (when chip works at V_{DD} = 2.7~3.6V)
  - Configurable bit length of a transaction word from 4 to 32-bit
  - Provides separate 4-level depth transmit and receive FIFO buffers
  - Supports MSB first or LSB first transfer sequence
  - Supports Byte Reorder function
  - Supports Byte or Word Suspend mode
  - Supports PDMA transfer
  - Supports 3-Wire, no slave selection signal, bi-direction interface
  - Supports one data channel half-duplex transfer
  - Supports receive-only mode
- I^2S Mode
  - Supports Master or Slave
  - Capable of handling 8-, 16-, 24- and 32-bit word sizes
  - Each provides two 4-level FIFO data buffers, one for transmitting and the other for receiving
  - Supports monaural and stereo audio data
  - Supports PCM mode A, PCM mode B, I^2S and MSB justified data format
  - Supports two PDMA requests, one for transmitting and the other for receiving

6.27.3 Block Diagram

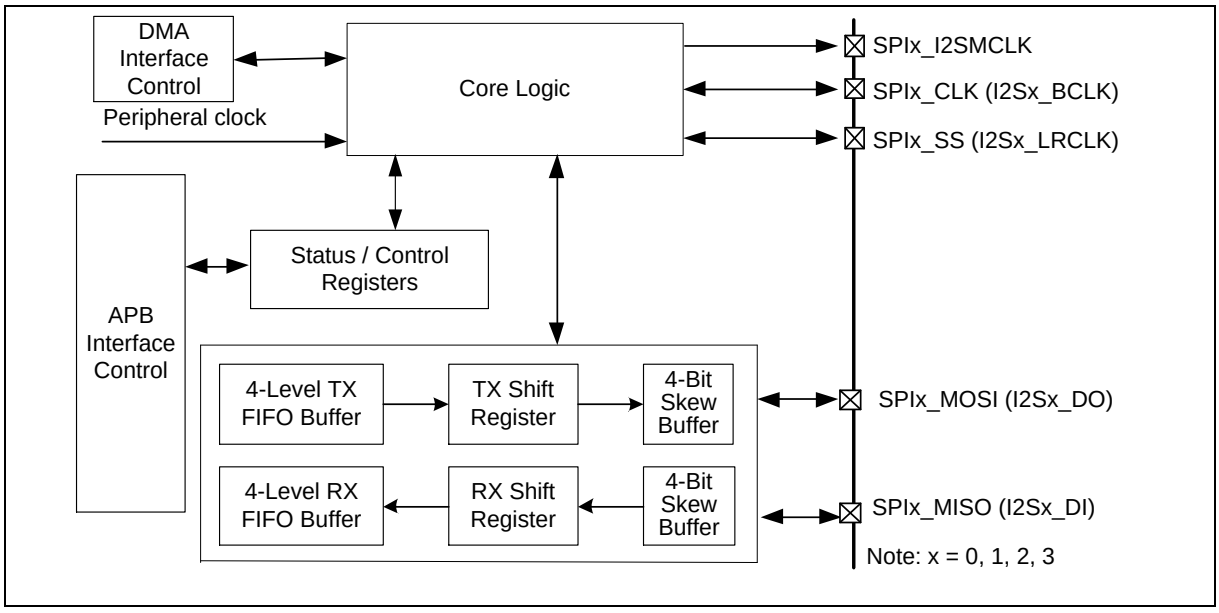


Figure 6.27-1 SPI Block Diagram

TX FIFO Buffer:

The transmit FIFO buffer is a 4-level depth, 32-bit wide, first-in, first-out register buffer. The data can be written to the transmit FIFO buffer in advance through software by writing the SPIx_TX register. In SPI mode, the transmit FIFO will be configured as 8-level while data length is set as 4~16 bits.

RX FIFO Buffer:

The receive FIFO buffer is also a 4-level depth, 32-bit wide, first-in, first-out register buffer. The receive control logic will store the receive data to this buffer. The FIFO buffer data can be read from SPIx_RX register by software. In SPI mode, the receive FIFO will be configured as 8-level while data length is set as 4~16 bits.

TX Shift Register:

The transmit shift register is a 32-bit wide register buffer. The transmit data is loaded from the TX FIFO buffer and shifted out bit-by-bit to the skew buffer.

RX Shift Register:

The receive shift register is also a 32-bit wide register buffer. The receive data is shift in bit-by-bit from the skew buffer and is loaded into RX FIFO buffer when a transaction done.

Skew Buffer:

The skew buffer is a 4-level depth, 1-bit wide buffer. There are two skew buffers in transmitting and received side. In received side, it is used to shift bits into RX shift register from SPI bus. In transmitting side, it is used to shift bits into SPI bus from TX shift register.

6.27.4 Basic Configuration

6.27.4.1 SPI0 Basic Configuration

- Clock source Configuration
  - Select the source of SPI0 peripheral clock on SPI0SEL (CLK_CLKSEL2[6:4]).

- Enable SPI0 peripheral clock in SPI0CKEN (CLK_APBCLK0[13]).

- Reset Configuration
  - Reset SPI0 controller in SPI0RST (SYS_IPRST1[13]).

6.27.4.2 SPI1 Basic Configuration

- Clock source Configuration
  - Select the source of SPI1 peripheral clock on SPI1SEL (CLK_CLKSEL2[14:12]).
  - Enable SPI1 peripheral clock in SPI1CKEN (CLK_APBCLK0[14]).
- Reset Configuration
  - Reset SPI1 controller in SPI1RST (SYS_IPRST1[14]).

6.27.4.3 SPI2 Basic Configuration

- Clock source Configuration
  - Select the source of SPI2 peripheral clock on SPI2SEL (CLK_CLKSEL3[11:9]).
  - Enable SPI2 peripheral clock in SPI2CKEN (CLK_APBCLK0[15]).
- Reset Configuration
  - Reset SPI2 controller in SPI2RST (SYS_IPRST1[15]).

6.27.4.4 SPI3 Basic Configuration

- Clock source Configuration
  - Select the source of SPI3 peripheral clock on SPI3SEL (CLK_CLKSEL3[14:12]).
  - Enable SPI3 peripheral clock in SPI3CKEN (CLK_APBCLK1[6]).
- Reset Configuration
  - Reset SPI3 controller in SPI3RST (SYS_IPRST2[6]).

SPI/I^{2}S (SPI0~SPI3) Interface Controller Pin description is shown as follows:

Pin	SPI Mode	I^{2}S Mode
SPIx_SS	SPI slave selection pin	I^{2}S left/right channel synchronization clock pin (I2Sx_LRCLK)
SPIx_CLK	SPI clock pin	I^{2}S bit clock pin (I2Sx_BCLK)
SPIx_MISO	SPI master input or slave output pin	I^{2}S data input pin (I2Sx_DI)
SPIx_MOSI	SPI master output or slave input pin	I^{2}S data output pin (I2Sx_DO)
SPIx_I2SMCLK	Not available	I^{2}S Master clock output pin

Table 6.27-1 SPI/I^{2}S Interface Controller Pin Description (SPI0~SPI3)

6.27.5 Functional Description

6.27.5.1 Terminology

SPI Peripheral Clock and SPI Bus Clock

The SPI controller needs the peripheral clock to drive the SPI logic unit to perform the data transfer. The peripheral clock rate is determined by the settings of clock divisor (SPIx_CLKDIV) and the clock source which can be HXT, PLL, PCLK, HIRC or PLLFN. SPIxSEL of CLK_CLKSEL2 or CLK_CLKSEL3 registers determine the clock source of the peripheral clock. The DIVIDER (SPIx_CLKDIV[8:0]) setting determines the divisor of the clock rate calculation.

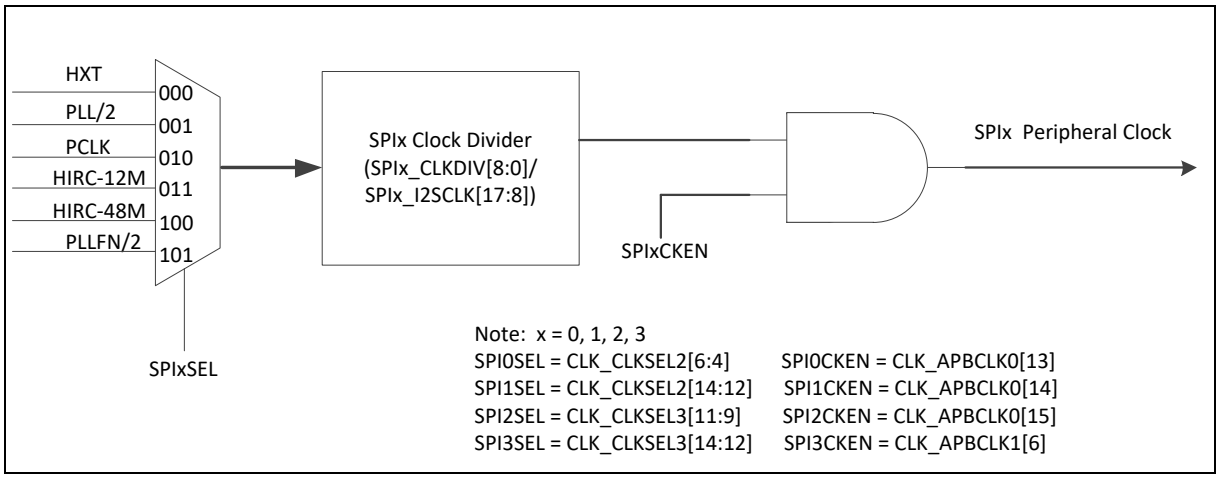


Figure 6.27-2 SPI Peripheral Clock

In Master mode, the frequency of the SPI bus clock is equal to the peripheral clock rate. In general, the SPI bus clock is denoted as SPI clock. In Slave mode, the SPI bus clock is provided by a master device. The frequency of SPI peripheral clock cannot be faster than the system clock rate regardless of Master or Slave mode. If the clock source of peripheral clock is not system clock, the frequency of SPI peripheral clock shall be slower than the system clock frequency regardless of Master or Slave mode.

In I^2S mode, the peripheral clock rate is equal to I^2S bit clock rate determined by SPIx_I2SCLK register.

Master/Slave mode

The SPI controllers can be set as Master or Slave mode by setting the SLAVE (SPIx_CTL[18]) to communicate with the off-chip SPI slave or master device. The HALFDPX (SPIx_CTL[14]) can be used to select the full-duplex or half-duplex in SPI transmission. The application block diagrams in Master and Slave mode are shown below.

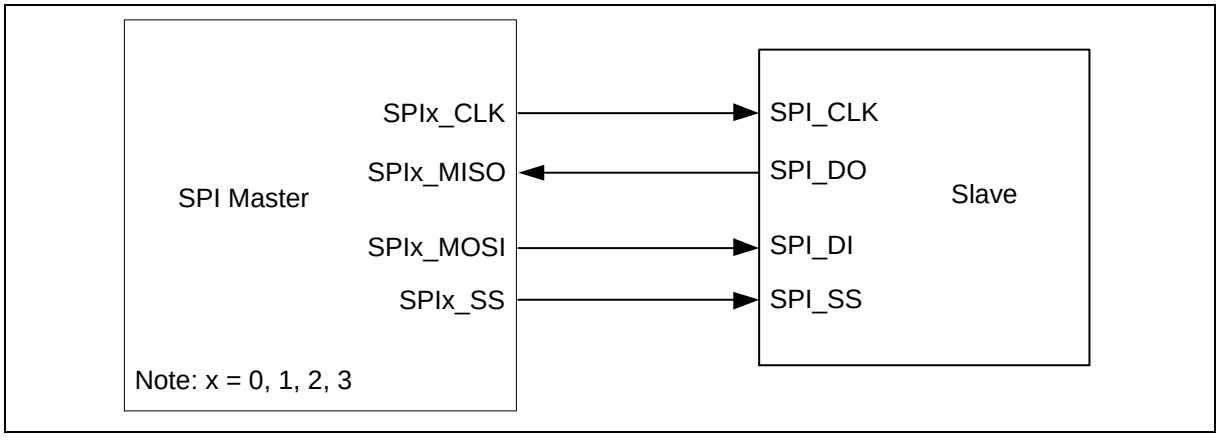


Figure 6.27-3 SPI Full-Duplex Master Mode Application Block Diagram

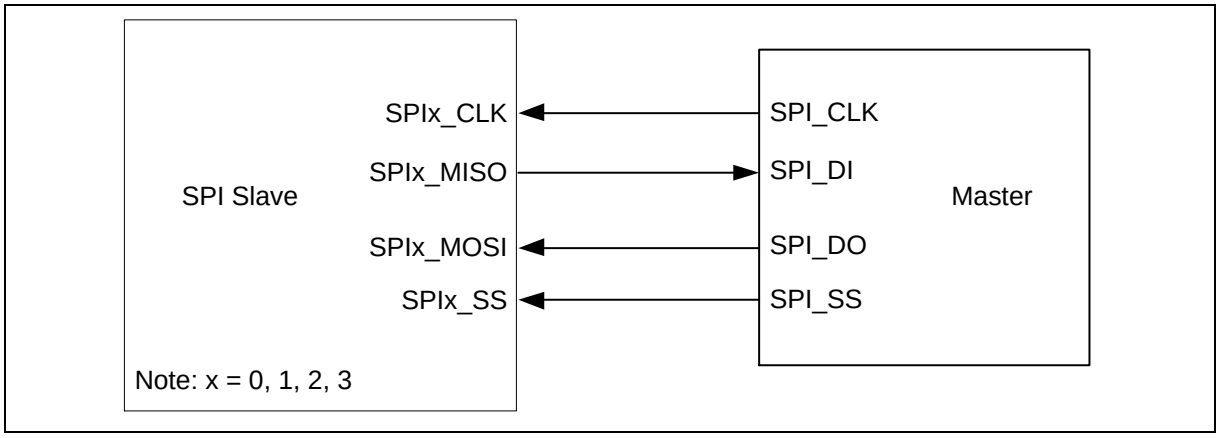


Figure 6.27-4 SPI Full-Duplex Slave Mode Application Block Diagram

Slave Selection

In Master mode, the SPI controller can drive off-chip slave device through the slave select output pin SPIx_SS. In Slave mode, the off-chip master device drives the slave selection signal from the SPIx_SS input port to this SPI controller. The duration between the slave select active edge and the first SPI clock input shall over 3 SPI peripheral clock cycles of slave.

In Master/Slave mode, the active state of slave selection signal can be programmed to low or high active in SSACTPOL (SPIx_SSCTL[2]). The selection of slave select conditions depends on what type of device is connected. In Slave mode, to recognize the inactive state of the slave selection signal, the inactive period of the slave selection signal must be larger than or equal to 3 peripheral clock cycles between two successive transactions.

In Slave mode, the SPIx_MISO signal type is controlled by the SPIx_SS pin. If the SPIx_SS pin is inactive state, the SPIx_MISO is set to tri-state to avoid interference with other Slave devices. If the SPIx_SS pin is active state, the SPIx_MISO is set to output mode for data transfer.

Timing Condition

The CLKPOL (SPIx_CTL[3]) defines the SPI clock idle state. If CLKPOL = 1, the output SPI clock is idle at high state; if CLKPOL = 0, it is idle at low state.

TXNEG (SPIx_CTL[2]) defines the data transmitted out either on negative edge or on positive edge of SPI clock. RXNEG (SPIx_CTL[1]) defines the data received either on negative edge or on positive edge of SPI clock.

Note: The settings of TXNEG and RXNEG are mutually exclusive. In other words, do not transmit and receive data at the same clock edge.

Transmit/Receive Bit Length

The bit length of a transaction word is defined in DWIDTH (SPIx_CTL[12:8]) and can be configured up to 32-bit length in a transaction word for transmitting and receiving.

When SPI controller finishes a transaction, i.e. receives or transmits a specific count of bits defined in DWIDTH (SPIx_CTL[12:8]), the unit transfer interrupt flag UNITIF (SPIx_STATUS[1]) will be set to 1.

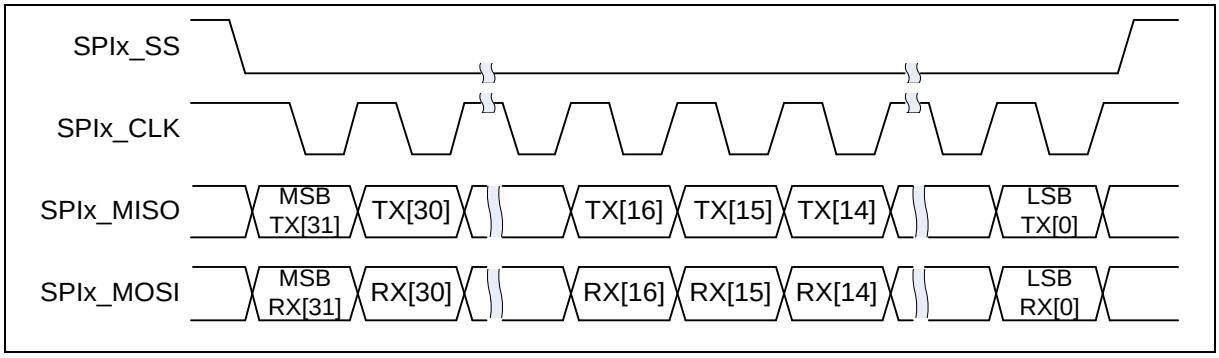


Figure 6.27-5 32-bit in One Transaction

LSB/MSB First

LSB (SPIx_CTL[13]) defines the bit transfer sequence in a transaction. If the LSB (SPIx_CTL[13]) is set to 1, the transfer sequence is LSB first. The bit 0 will be transferred firstly. If the LSB (SPIx_CTL[13]) is cleared to 0, the transfer sequence is MSB first.

Suspend Interval

SUSPITV (SPIx_CTL[7:4]) provides a configurable suspend interval, 0.5 ~ 15.5 SPI clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SUSPITV is 0x3 (3.5 SPI clock cycles).

6.27.5.2 Automatic Slave Selection

In Master mode, if AUTOSS (SPIx_SSCTL[3]) is set, the slave selection signal will be generated automatically and output to the SPIx_SS pin according to whether SS (SPIx_SSCTL[0]) is enabled or not. The slave selection signal will be set to active state by the SPI controller when the SPI data transfer is started by writing to FIFO. It will be set to inactive state when SPI bus is idle. If SPI bus is not idle, i.e. TX FIFO, TX shift register or TX skew buffer is not empty, the slave selection signal will be set to inactive state between transactions if the value of SUSPITV (SPIx_CTL[7:4]) is greater than or equal to 3.

In Master mode, if the value of SUSPITV is less than 3 and the AUTOSS is set as 1, the slave selection signal will be kept at active state between two successive transactions.

If the AUTOSS bit is cleared, the slave selection output signal will be determined by the SS setting. The active state of the slave selection output signal is specified in SSACTPOL (SPIx_SSCTL[2]).

The duration between the slave selection signal active edge and the first SPI bus clock edge is 1 SPI bus clock cycle and the duration between the last SPI bus clock and the slave selection signal inactive edge is 1.5 SPI bus clock cycle.

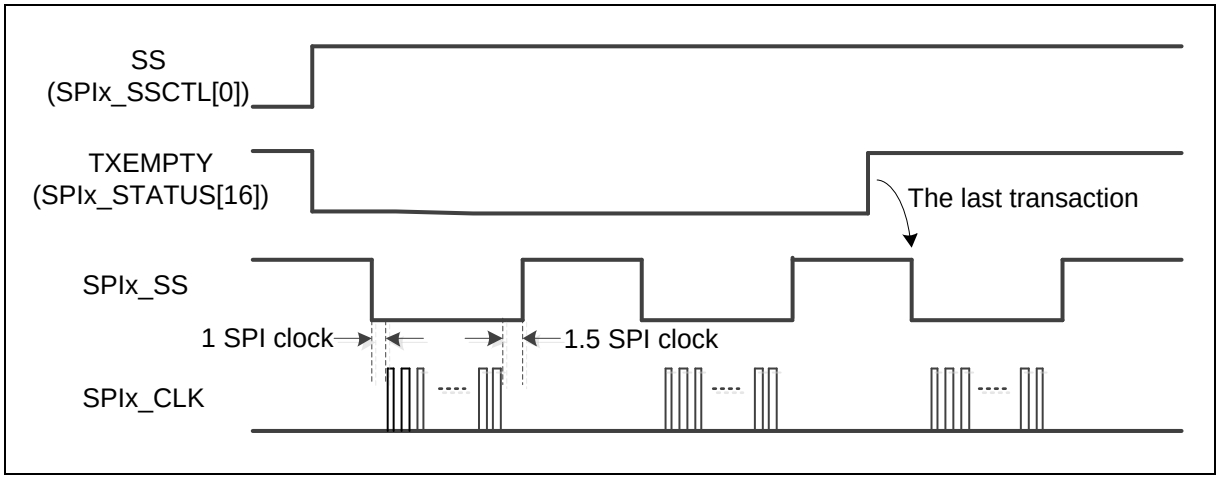


Figure 6.27-6 Automatic Slave Selection (SSACTPOL = 0, SUSPITV > 0x2)

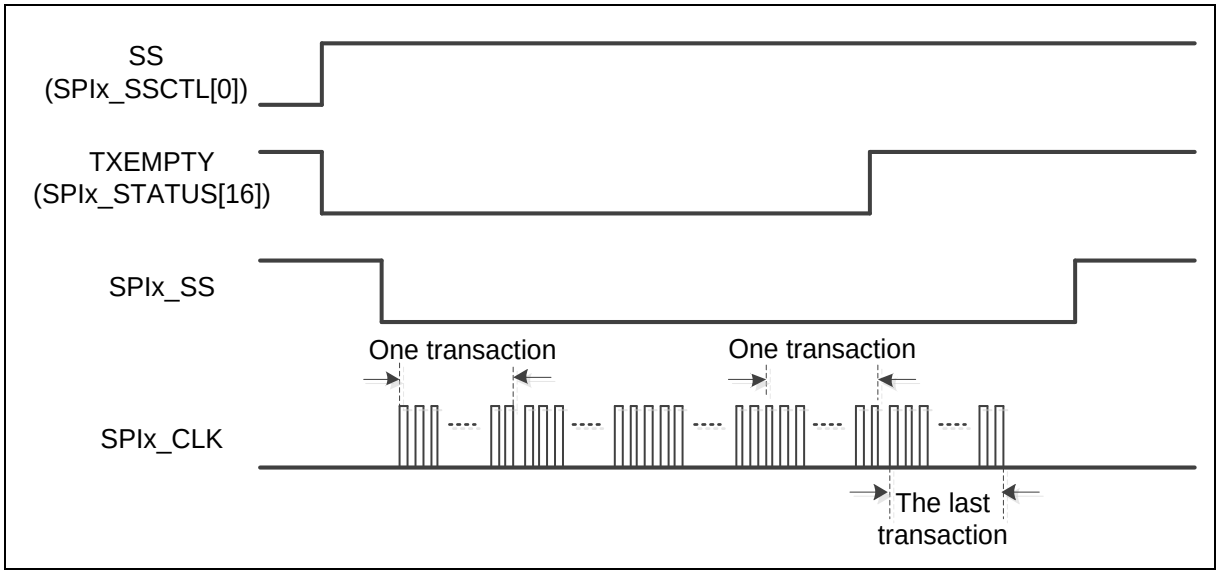


Figure 6.27-7 Automatic Slave Selection (SSACTPOL = 0, SUSPITV < 0x3)

6.27.5.3 Byte Reorder and Suspend Function

When the transfer is set as MSB first (LSB = 0) and the REORDER (SPIx_CTL[19]) is set to 1, the data stored in the TX buffer and RX buffer will be rearranged in the order as [Byte0, Byte1, Byte2, Byte3] in 32-bit transfer (DWIDTH = 0). The sequence of transmitted/received data will be Byte0, Byte1, Byte2, and then Byte3. If the DWIDTH is set as 24-bit transfer mode, the data in TX buffer and RX buffer will be rearranged as [unknown byte, Byte0, Byte1, Byte2]. The SPI controller will transmit/receive data with the sequence of Byte0, Byte1 and then Byte2. Each byte will be transmitted/received with MSB first. The rule of 16-bit mode is the same as above. Byte Reorder function is only available when DWIDTH is configured as 16, 24, and 32 bits.

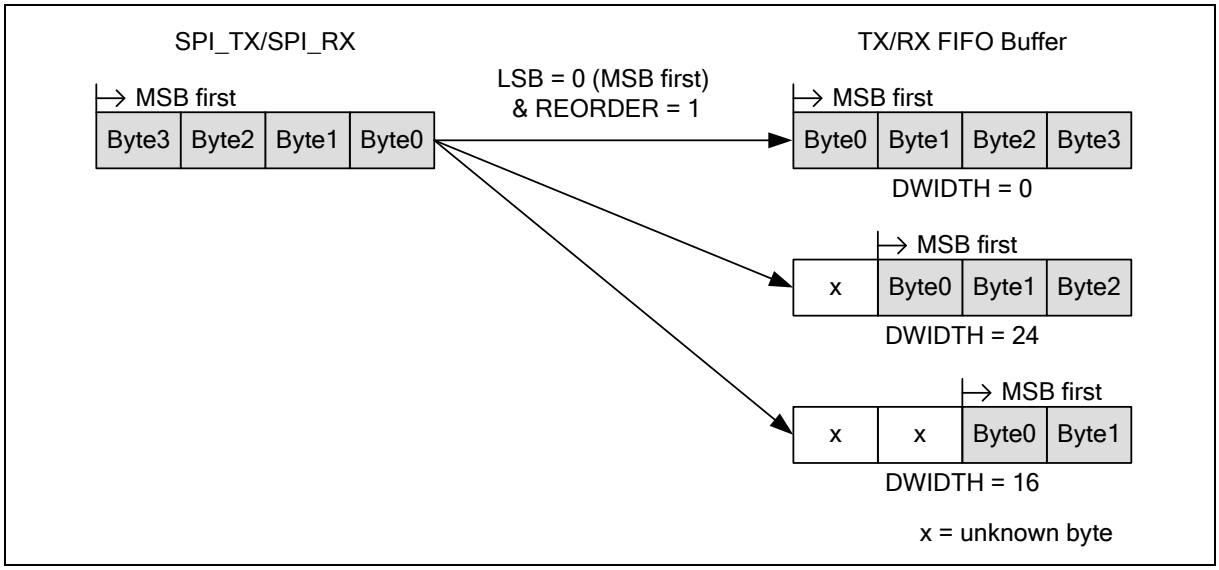


Figure 6.27-8 Byte Reorder Function

In Master mode, if REORDER (SPIx_CTL[19]) is set to 1, a suspend interval of 0.5 ~ 15.5 SPI clock periods will be inserted by hardware between two successive bytes in a transaction word. The suspend interval is configured in SUSPITV (SPIx_CTL[7:4]).

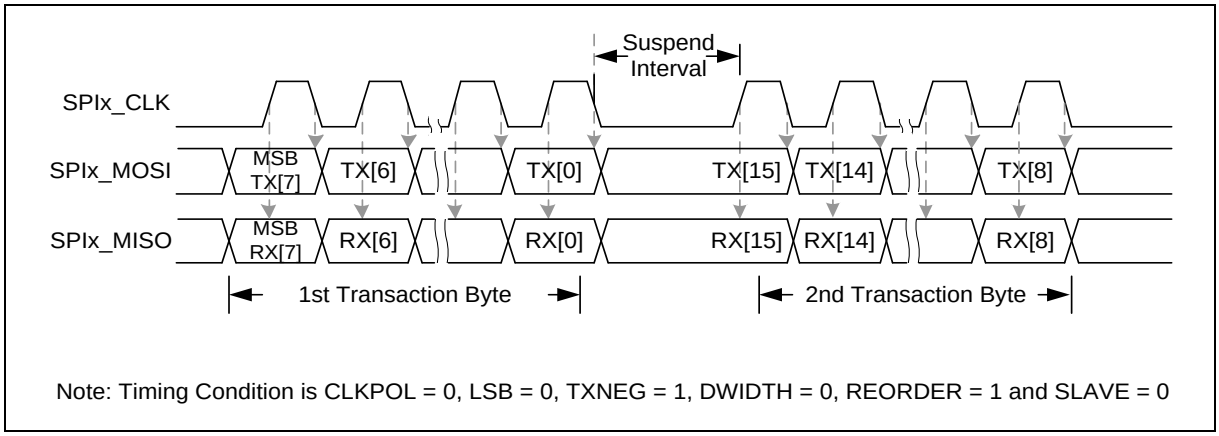


Figure 6.27-9 Timing Waveform for Byte Suspend

6.27.5.4 Half-Duplex Communication

The SPI controller can communicate in half-duplex mode by setting HALFDPX (SPIx_CTL[14]) bit. In half-duplex mode, there is only one data line for receiving or transmitting data direction which is defined by DATDIR (SPIx_CTL[20]). In half-duplex configuration, the SPIx_MISO pin is free for other applications and it can be configured as GPIO. Enabling or disabling the control bit HALFDPX (SPIx_CTL[14]) will produce TXFBCLR (SPIx_FIFOCTL[9]) and RXFBCLR (SPIx_FIFOCTL[8]) at the same time automatically.

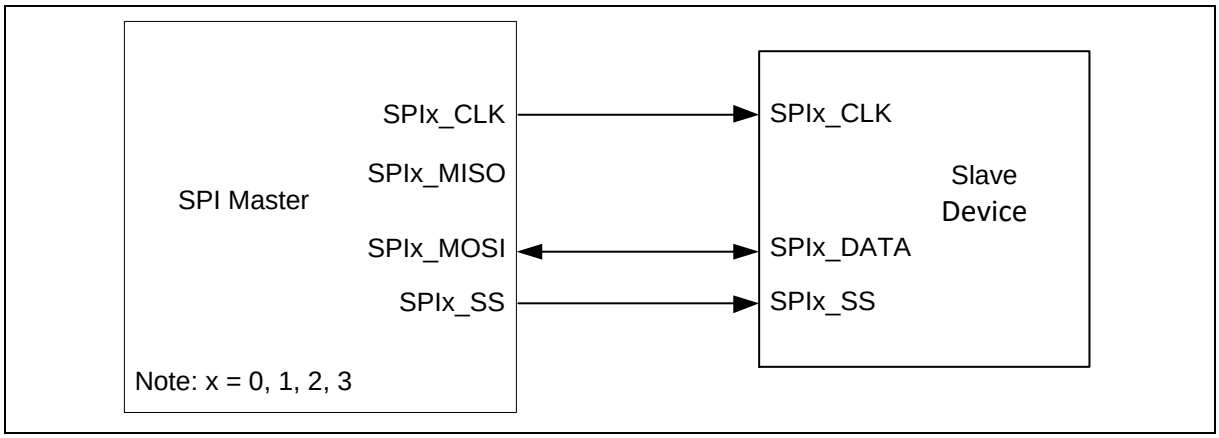


Figure 6.27-10 SPI Half-Duplex Master Mode Application Block Diagram

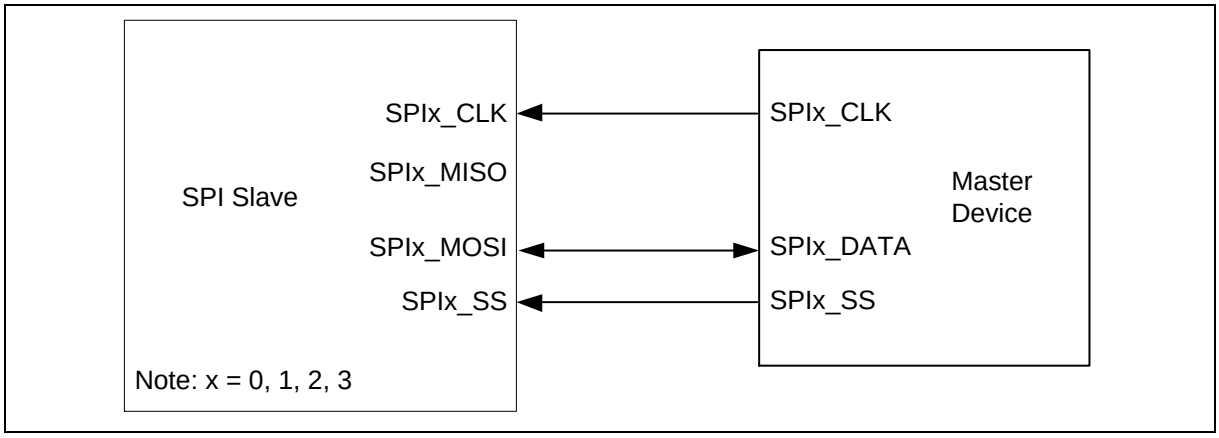


Figure 6.27-11 SPI Half-Duplex Slave Mode Application Block Diagram

6.27.5.5 Receive-Only Mode

In SPI Master device, it can communicate in receive-only mode by setting RXONLY (SPIx_CTL[15]). In this configuration, the SPI Master device will generate SPI bus clock continuously as long as the receive-only mode is enabled for receiving data bit from SPI slave device. If AUTOSS (SPIx_SSCTL[3]) is enabled in receive-only mode, SPI Master will keep activating the slave select signal.

The remaining SPIx_MOSI pin of SPI Master device is not used for communication and can be configured as GPIO. The status BUSY (SPIx_STATUS[0]) will be asserted in receive-only mode due to the generation of SPI bus clock. Entering this mode will produce the TXFBCLR (SPIx_FIFOCTL[9]) and RXFBCLR (SPIx_FIFOCTL[8]) at the same time automatically. When user enables this mode, the output SPI bus clock will be sent out after 6 peripheral clock cycles. In this mode, the data which has been written into transmit FIFO will be loaded into transmit shift register and sent out.

When user sets RXONLY (SPIx_CTL[15]) to enable, SPI RX data with data bit width of DWIDTH (SPIx_CTL[12:8]) will be received into RX FIFO and SPI clock will be sent to SPI slave device until RX FIFO is full.

For data bit width of 4~16 bits, the SPI master will send SPI output clock to SPI slave and receive RX data when RX FIFO counter RXCNT (SPIx_STATUS[27:24]) is less than or equal to 6.

For data bit width of 17~32 bits, the SPI master will send SPI output clock to SPI slave and receive RX data when RX FIFO counter RXCNT (SPIx_STATUS[27:24]) is less than or equal to 2.

6.27.5.6 Slave 3-Wire Mode

When SLV3WIRE (SPIx_SSCTL[4]) is set by software to enable the Slave 3-Wire mode, the SPI controller can work with no slave selection signal in Slave mode. The SLV3WIRE (SPIx_SSCTL[4]) only takes effect in Slave mode. Only three pins, SPIx_CLK, SPIx_MISO, and SPIx_MOSI, are required to communicate with a SPI master. The SPIx_SS pin can be configured as a GPIO. When the SLV3WIRE (SPIx_SSCTL[4]) is set to 1, the SPI slave will be ready to transmit/receive data after the SPIEN (SPIx_CTL[0]) is set to 1.

6.27.5.7 PDMA Transfer Function

SPI controller supports PDMA transfer function.

When TXPDMAEN (SPIx_PDMACTL[0]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

When RXPDMAEN (SPIx_PDMACTL[1]) is set to 1, the controller will start the PDMA reception process. SPI controller will issue request to PDMA controller automatically when there is data in the RX FIFO buffer.

Note: SPI supports single request PDMA (Read/Write) only; burst request PDMA is not supported.

6.27.5.8 FIFO Buffer Operation

The SPI controllers are equipped with four 32-bit wide transmit and receive FIFO buffers. The data stored in the transmit FIFO buffer will be read and sent out by the transmission control logic. If the transmit FIFO buffer is full, the TXFULL (SPIx_STATUS[17]) will be set to 1. When the SPI transmission logic unit draws out the last datum of the transmit FIFO buffer, so that the transmit FIFO buffer is empty, the TXEMPTY (SPIx_STATUS[16]) will be set to 1. Note that the TXEMPTY (SPIx_STATUS[16]) flag is set to 1 while the last transaction is still in progress. In Master mode, the BUSY (SPIx_STATUS[0]) is set to 1 when the FIFO buffer is written any data or there is any transaction on the SPI bus. (e.g. the slave selection signal is active and the SPI controller is receiving data in Slave mode). It will set to 0 when the transmit FIFO is empty and the current transaction has done. Thus, the status of BUSY (SPIx_STATUS[0]) should be checked by software to make sure whether the SPI is in idle or not.

The receive control logic will store the SPI input data into the receive FIFO buffer. There are FIFO related status bits, like RXEMPTY (SPIx_STATUS[8]) and RXFULL (SPIx_STATUS[9]), to indicate the current status of RX FIFO buffer.

The transmitting and receiving threshold can be configured by setting TXTH (SPIx_FIFOCTL[30:28])

and RXTH (SPIx_FIFOCTL[26:24]). When the count of valid data stored in transmit FIFO buffer is less than or equal to TXTH (SPIx_FIFOCTL[30:28]) setting, TXTHIF (SPIx_STATUS[18]) will be set to 1. When the count of valid data stored in receive FIFO buffer is larger than RXTH (SPIx_FIFOCTL[26:24]) setting, RXTHIF (SPIx_STATUS[10]) will be set to 1.

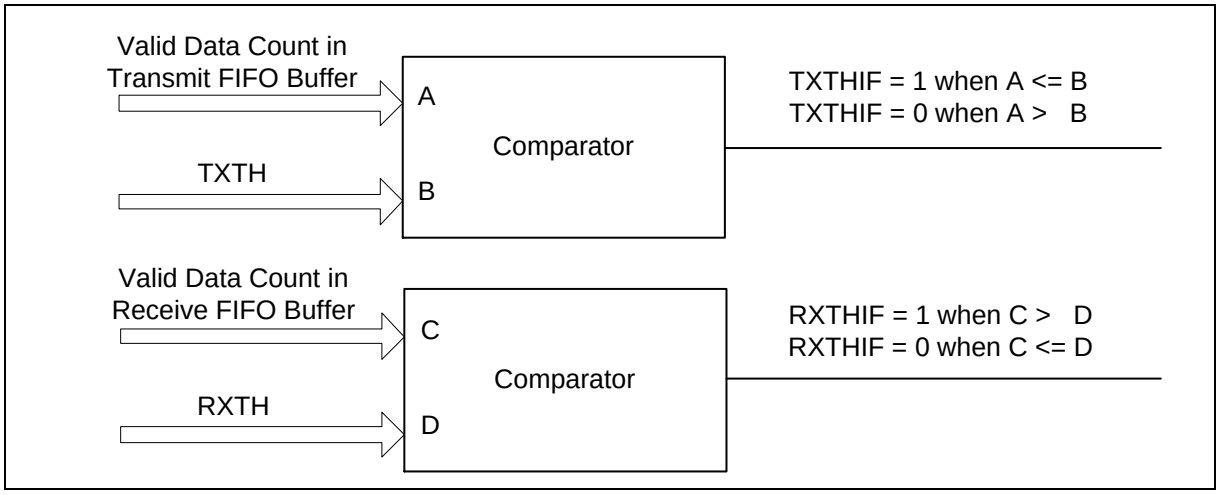


Figure 6.27-12 FIFO Threshold Comparator

In Master mode, when the first datum is written to the SPIx_TX register, the TXEMPTY flag (SPIx_STATUS[16]) will be cleared to 0. The transmission will start after 1 PCLK clock cycles and 6 peripheral clock cycles. User can write the next data into SPIx_TX register immediately. The SPI controller will insert a suspend interval between two successive transactions. The period of suspend interval is decided by the setting of SUSPITV (SPIx_CTL[7:4]). If the SUSPITV (SPIx_CTL[7:4]) equals 0, SPI controller can perform continuous transfer. User can write data into SPIx_TX register as long as the TXFULL (SPIx_STATUS[17]) is 0.

The example 1 of Figure 6.28-20 indicates the updated condition of TXEMPTY (SPIx_STATUS[16]) and the relationship among the FIFO buffer, shift register and the skew buffer for 4~16 bits of data length. The TXEMPTY (SPIx_STATUS[16]) is set to 0 when the Data0 is written into the FIFO buffer. The Data0 will be loaded into the shift register by the core logic and the TXEMPTY (SPIx_STATUS[16]) will be 1. The Data0 in shift register will be shifted into skew buffer by bit for transmission until the transfer is done.

The Example 2 of Figure 6.28-20 indicates the updated condition of TXFULL (SPIx_STATUS[17]) when there are 8 data in the FIFO buffer and the next data of Data9 is not written into the FIFO buffer when the TXFULL = 1.

The example 1 of Figure 6.27-14 indicates the updated condition of TXEMPTY (SPIx_STATUS[16]) and the relationship among the FIFO buffer, shift register and the skew buffer for 17~32 bits of data length.

The Example 2 of Figure 6.27-14 indicates the updated condition of TXFULL (SPIx_STATUS[17]) when there are 4 data in the FIFO buffer and the next data of Data5 is not written into the FIFO buffer when the TXFULL = 1.

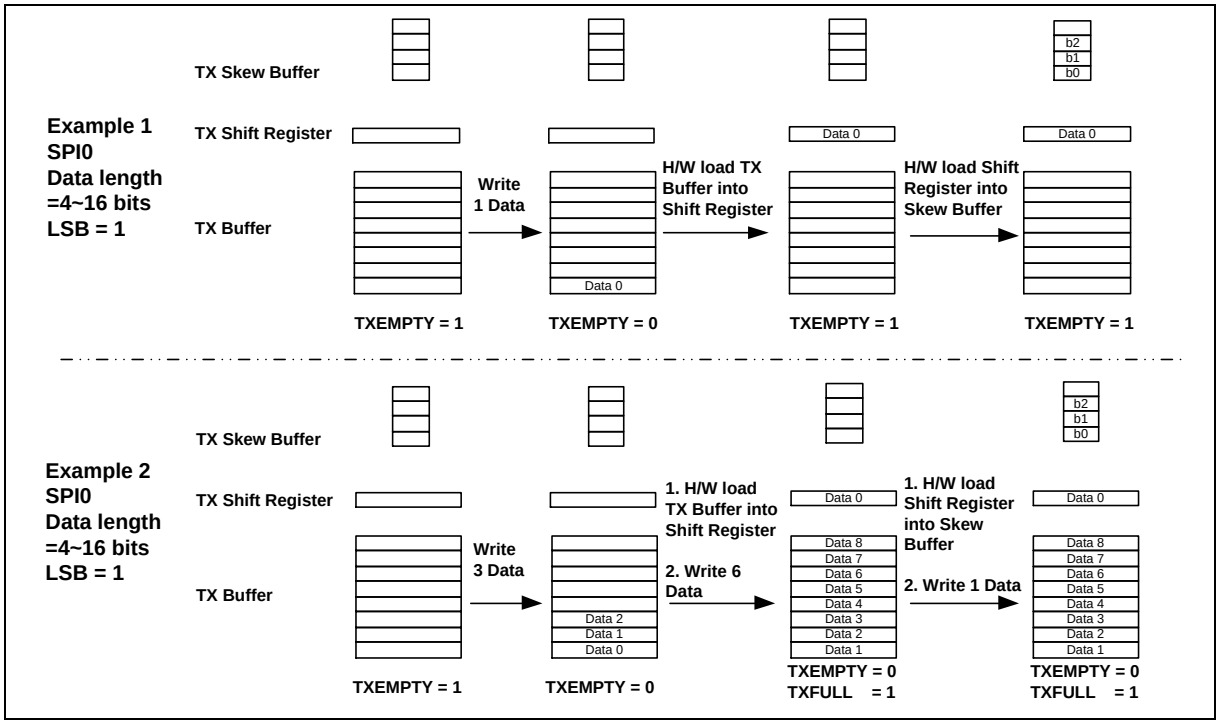


Figure 6.27-13 Transmit FIFO Buffer Example for 4~16 bits of data length

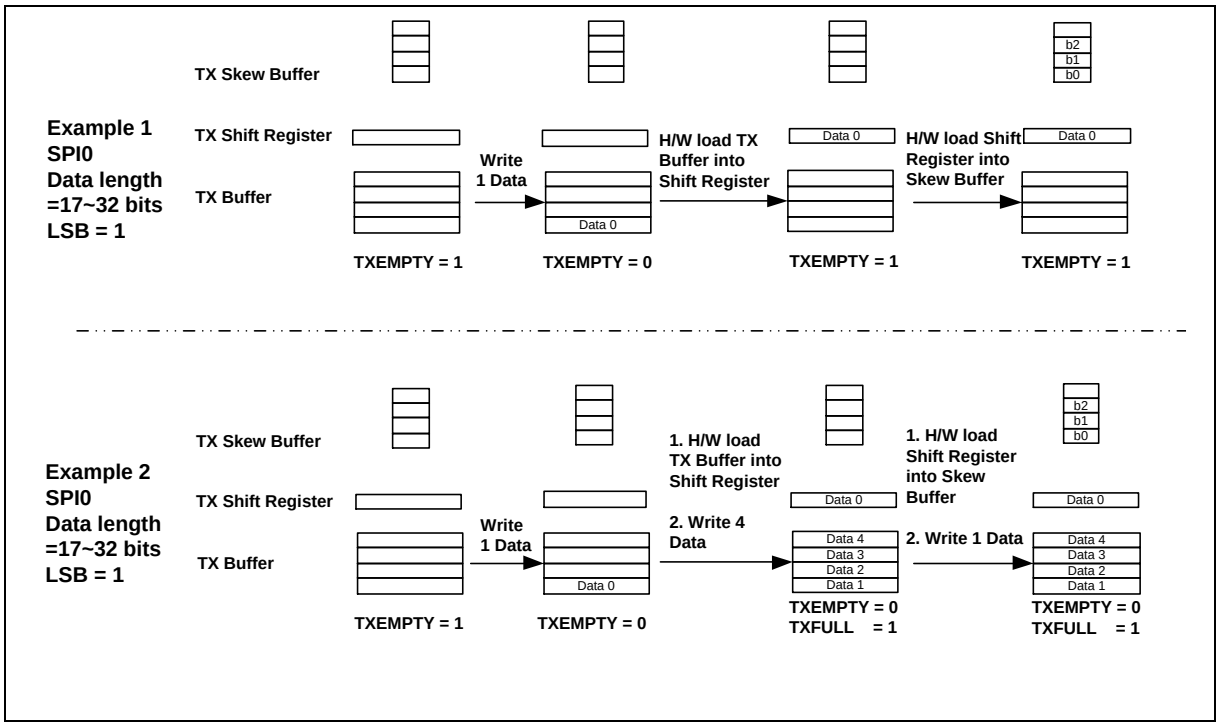


Figure 6.27-14 Transmit FIFO Buffer Example for 17~32 bits of data length

The subsequent transactions will be triggered automatically if the transmitted data are updated in time. If the SPIx_TX register is not updated after all data transfer are done, the transfer will stop.

In Master mode, during receiving operation, the serial data are received from SPIx_MISO pin and stored to receive FIFO buffer.

The received data (Data0's b0, b1, …b31) is stored into skew buffer first according the serial clock (SPIx_CLK) and then it is shift into the shift register bit by bit. The core logic will load the data in shift register into FIFO buffer when the received data bit count reach the value of DWIDTH (SPIx_CTL[12:8]).

The RXEMPTY (SPIx_STATUS[8]) will be cleared to 0 while the receive FIFO buffer contains unread data (see the Example 1 of Receive FIFO Buffer Example in Figure 6.27-15). The received data can be read by software from SPIx_RX register as long as the RXEMPTY (SPIx_STATUS[8]) is 0. If the receive FIFO buffer contains 8 unread data, the RXFULL (SPIx_STATUS[9]) will be set to 1 (see the Example 2 of Receive FIFO Buffer Example in Figure 6.27-15).

The RXEMPTY (SPIx_STATUS[8]) will be cleared to 0 while the receive FIFO buffer contains unread data (see the Example 1 of Receive FIFO Buffer Example in Figure 6.27-16). The received data can be read by software from SPIx_RX register as long as the RXEMPTY (SPIx_STATUS[8]) is 0. If the receive FIFO buffer contains 4 unread data, the RXFULL (SPIx_STATUS[9]) will be set to 1 (see the Example 2 of Receive FIFO Buffer Example in Figure 6.27-16).

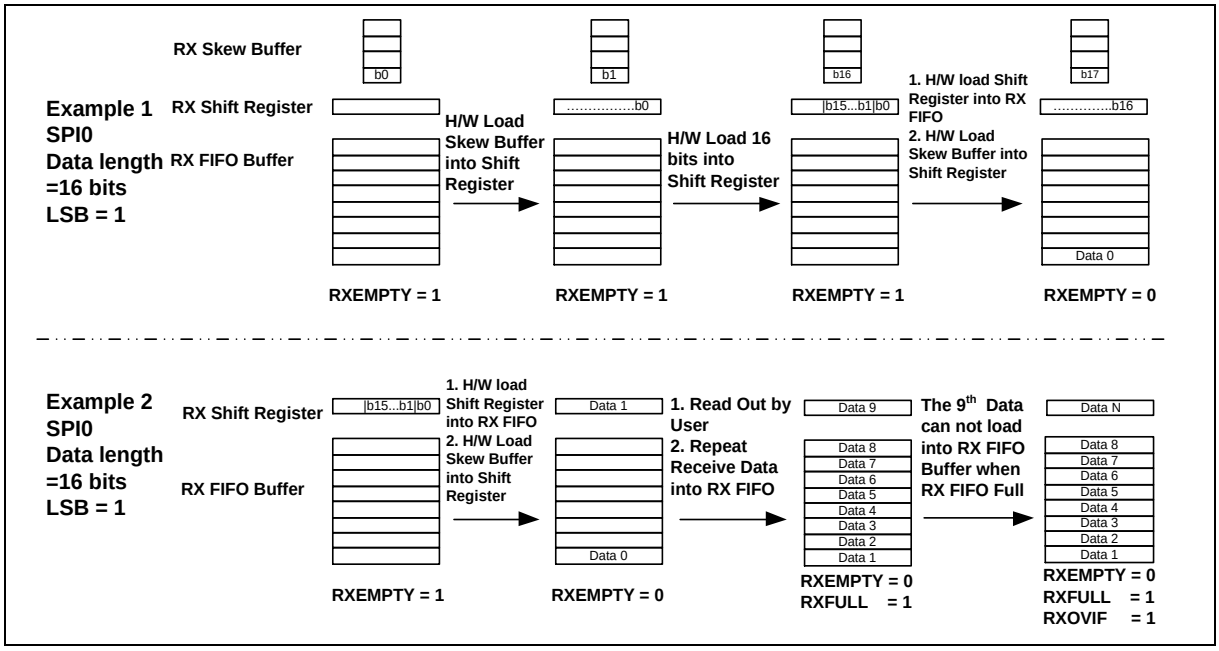


Figure 6.27-15 Receive FIFO Buffer Example for 16 bits of Data Length

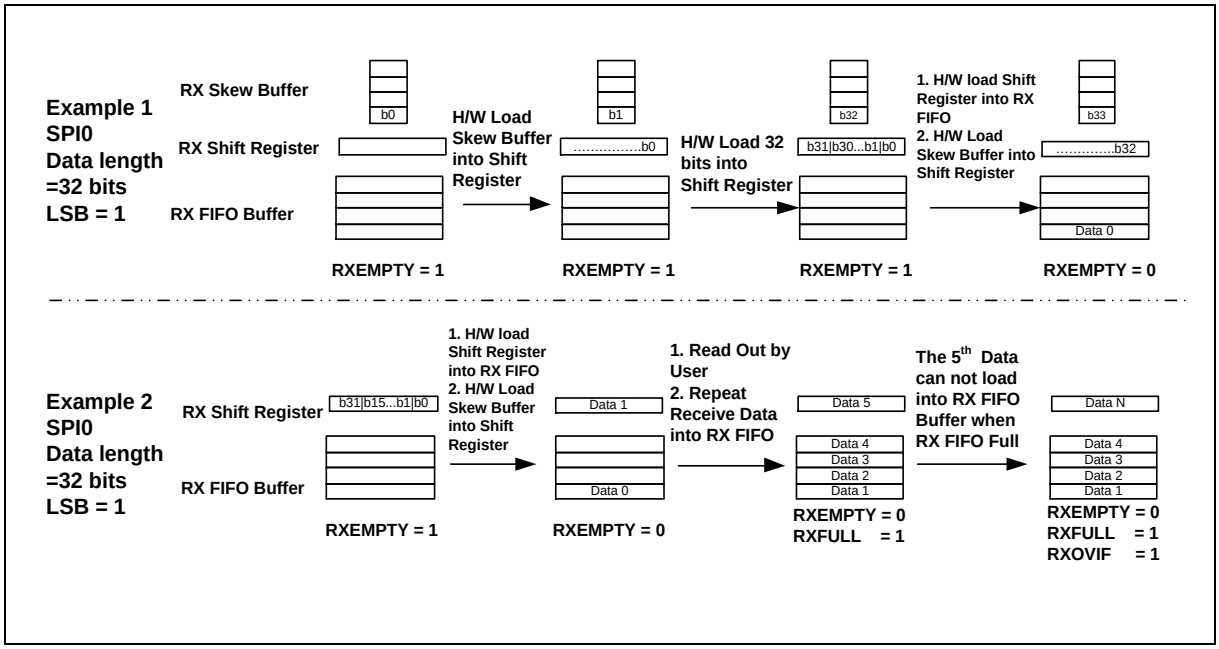


Figure 6.27-16 Receive FIFO Buffer Example for 32 bits of Data Length

In Slave mode, during transmission operation, when data is written to the SPIx_TX register by software, the data will be loaded into transmit FIFO buffer and the TXEMPTY (SPIx_STATUS[16]) will be set to 0. The transmission will start when the slave device receives clock signal from master. Data can be written to SPIx_TX register as long as the TXFULL (SPIx_STATUS[17]) is 0. After all data have been drawn out by the SPI transmission logic unit and the SPIx_TX register is not updated by software, the TXEMPTY (SPIx_STATUS[16]) will be set to 1.

If there is no any data written to the SPIx_TX register, the transmit underflow interrupt flag, TXUFIF (SPIx_STATUS[19]) will be set to 1 when the slave selection signal is active. The output data will be held by TXUFPOL (SPIx_FIFOCTL[6]) setting during this transfer until the slave selection signal goes to inactive state. When the transmit underflow event occurs, the slave under run interrupt flag, SLVURIF (SPIx_STATUS[7]), will be set to 1 as SPIx_SS goes to inactive state.

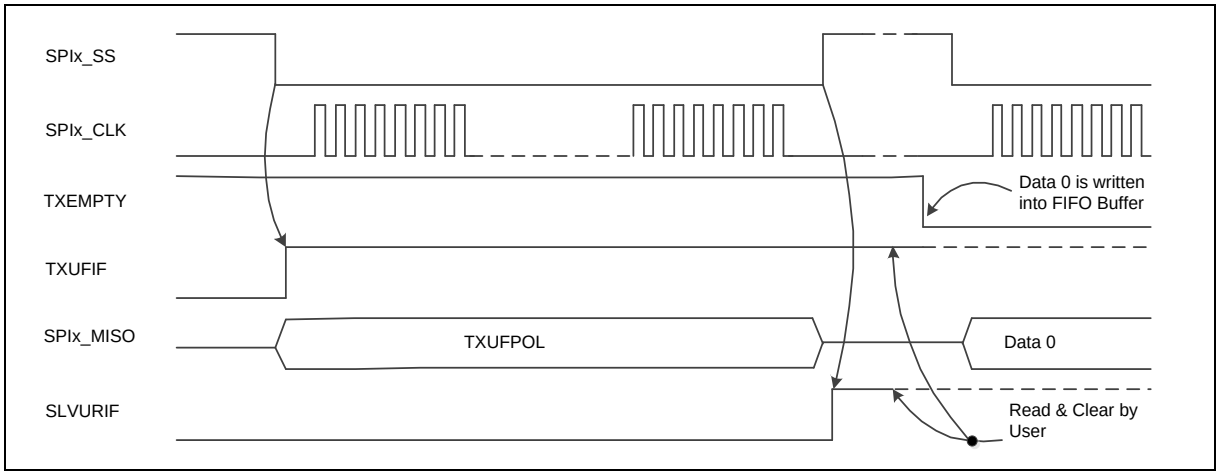


Figure 6.27-17 TX Underflow Event and Slave Under Run Event

In SPI Slave 3-Wire mode, the first 2-bit data is un-predicted (kept on the level of last bit in previously transfer) if the data is written into TX FIFO among 3 peripheral clock cycles before the SPI bus clock is presented. The other bits are held by TXUFPOL (SPIx_FIFOCTL[6]) because there is TX underflow

event. The written data will be transmitted in the next transfer.

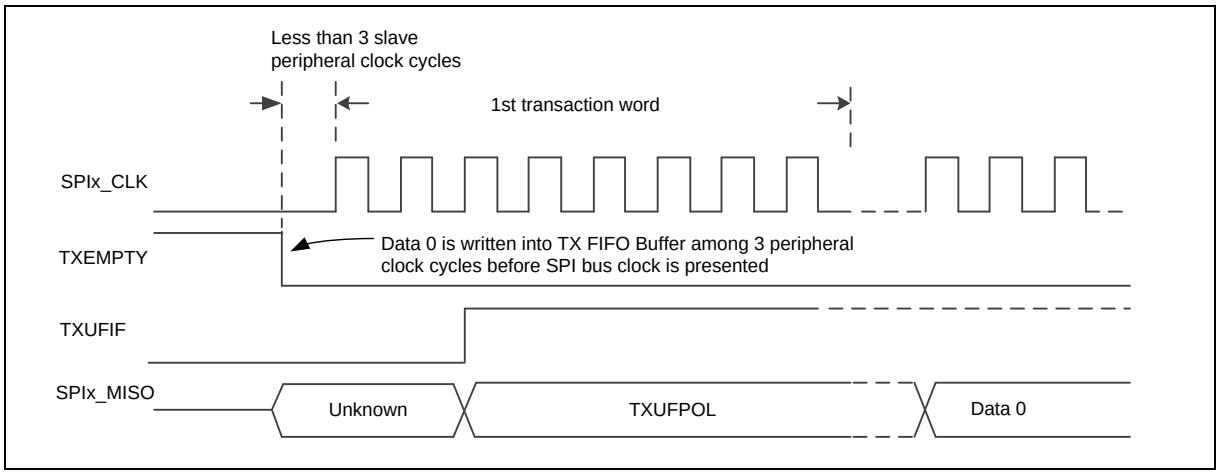


Figure 6.27-18 TX Underflow Event (SPI Slave 3-Wire Mode Enabled)

In Slave mode, during receiving operation, the serial data is received from SPIx_MOSI pin and stored to SPIx_RX register. The reception mechanism is similar to Master mode reception operation. If the receive FIFO buffer contains 4 unread data, the RXFULL (SPIx_STATUS[9]) will be set to 1 and the RXOVIF (SPIx_STATUS[11]) will be set to 1 if there is more serial data received from SPIx_MOSI and follow-up data will be dropped (refer to the Receive FIFO Buffer Example figure).

If the receive bit count mismatches with the DWIDTH (SPIx_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (SPIx_STATUS[6]) will be set to 1 in SPI Slave mode. RX data will be written into RX FIFO and SLVBENUM (SPIx_STATUS2[29:24]) will be updated when SLVBERX (SPIx_FIFOCTL[10]) is enabled and SPI slave bit count error event happened. RX data will be dropped and SLVBENUM (SPIx_STATUS2[29:24]) will be fixed to 0x0 when the control register SLVBERX (SPIx_FIFOCTL[10]) is disabled and SPI slave bit count error event happened. The status register SLVBENUM (SPIx_STATUS2[29:24]) indicates that effective bit number of uncompleted RX data when SPI slave bit count error happened. InFigure 6.27-19, the timing diagram of SPI slave bit count error and SLVBENUM is shown, and SLVBENUM will be updated to 0x8 when SPI slave device receives 8 bits data and DWIDTH (SPIx_CTL[12:8]) sets to 16 bits at SPI bit count error event. The SPI slave bit count error event (SLVBEIF) will generate RX FIFO write operation pulse to write RX data from RX shift register to RX FIFO, and RX FIFO count (RXCNT) will be updated to one.

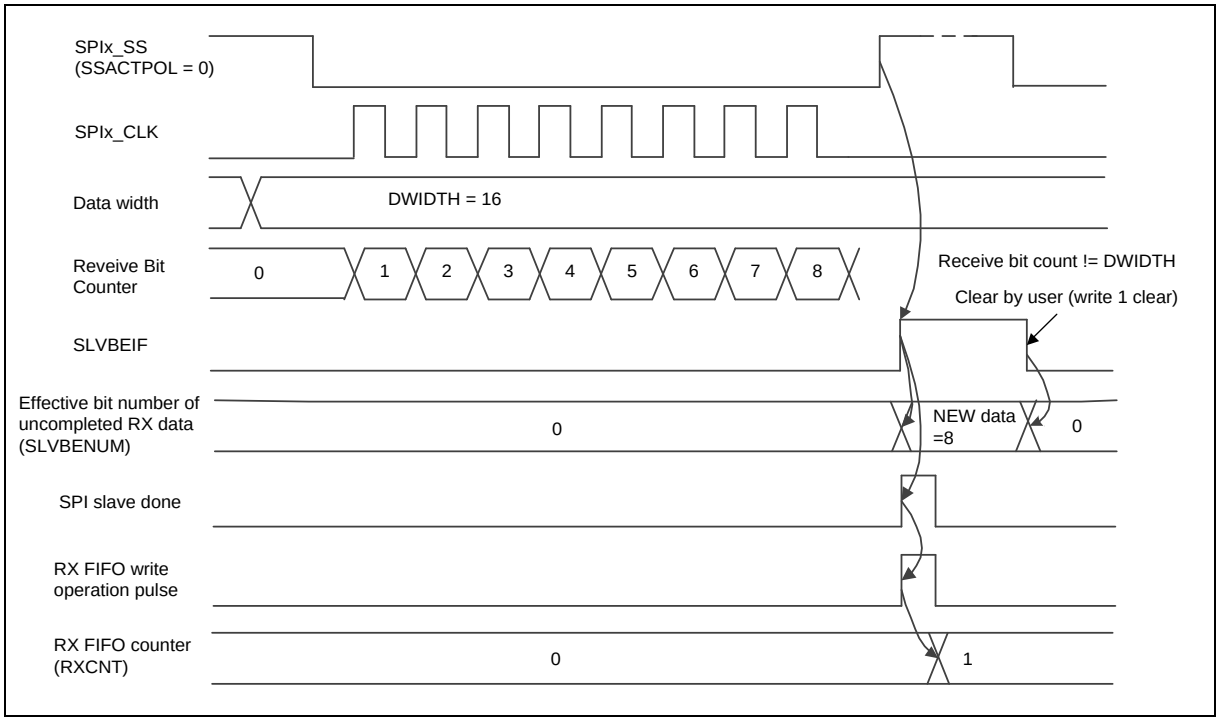


Figure 6.27-19 Slave Mode Bit Count Error and Effective Bit Number of Uncompleted RX data

A receive time-out function is built in this controller. When the receive FIFO is not empty and no read operation in receive FIFO over 64 SPI peripheral clock periods in Master mode or over 576 SPI peripheral clock periods in Slave mode, the receive time-out occurs and the RXTOIF (SPIx_STATUS[12]) will be set to 1. When the receive FIFO is read by user, the time-out status will be cleared automatically.

6.27.5.9 Interrupt

- SPI unit transfer interrupt

  As the SPI controller finishes a unit transfer, the unit transfer interrupt flag UNITIF (SPIx_STATUS[1]) will be set to 1. The unit transfer interrupt event will generate an interrupt to CPU if the unit transfer interrupt enable bit UNITIEN (SPIx_CTL[17]) is set. The unit transfer interrupt flag can be cleared only by writing 1 to it.

- SPI slave selection active/inactive interrupt

  In Slave mode, the slave selection active/inactive interrupt flag, SSACTIF (SPIx_STATUS[2]) and SSINAIF (SPIx_STATUS[3]), will be set to 1 when the SPIEN (SPIx_CTL[0]) and SLAVE (SPIx_CTL[18]) are set to 1 and the slave selection signal goes to active/inactive state. The SPI controller will issue an interrupt if the SSINAIEN (SPIx_SSCTL[13]) or SSACTIEN (SPIx_SSCTL[12]), is set to 1.

- Slave bit count error interrupt

  In Slave mode, if the transmit/receive bit count mismatches with the DWIDTH (SPIx_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (SPIx_STATUS[6]) will be set to 1. The uncompleted transaction will be dropped from TX shift register. When the control register SLVBERX (SPIx_FIFOCTL[10]) is disabled and SPI slave bit count error event happened, the uncompleted transaction data will be dropped from RX shift registers. When control register SLVBERX (SPIx_FIFOCTL[10]) is enabled and SPI slave bit count error event happened, the uncompleted transaction data will be written from RX shift registers into RX FIFO. The SPI controller will issue an interrupt if the SLVBEIEN (SPIx_SSCTL[8]) is set to 1.

Note: If the slave selection signal is active but there is no any serial clock input, the SLVBEIF (SPIx_STATUS[6]) will be set to 1 when the slave selection signal goes to inactive state.

- TX underflow interrupt

  In SPI Slave mode, if there is no any data is written to the SPIx_TX register, the TXUFIF (SPIx_STATUS[19]) will be set to 1 when the slave selection signal is active. The SPI controller will issue a TX underflow interrupt if the TXUFIEN (SPIx_FIFOCTL[7]) is set to 1.

  Note: If underflow event occurs in SPI Slave mode, there are two conditions which make SPI Slave mode return to idle state and then goes for next transfer: (1) set TXRST to 1 (2) slave select signal is changed to inactive state while SLV3WIRE=0.

- Slave TX under run interrupt

  If the TX underflow event occurs, the SLVURIF (SPIx_STATUS[7]) will be set to 1 when SPIx_SS goes to inactive state. The SPI controller will issue a TX under run interrupt if the SLVURIEN (SPIx_SSCTL[9]) is set to 1.

  Note: In Slave 3-Wire mode, the slave selection signal is considered active all the time so that user shall poll the TXUFIF (SPIx_STATUS[19]) to know if there is TX underflow event or not.

- Receive Overrun interrupt

  In Slave mode, if the receive FIFO buffer contains 4 unread data for 17~32-bits of data length, the RXFULL (SPIx_STATUS[9]) will be set to 1 and the RXOVIF (SPIx_STATUS[11]) will be set to 1 if there is more serial data received from SPI bus and follow-up data will be dropped. The SPI controller will issue an interrupt if the RXOVIEN (SPIx_FIFOCTL[5]) is set to 1.

  If the receive FIFO buffer contains 8 unread data for 4~16-bits of data length, the RXFULL (SPIx_STATUS[9]) will be set to 1 and the RXOVIF (SPIx_STATUS[11]) will be set to 1 if there is more serial data received from SPI bus and follow-up data will be dropped.The SPI controller will issue an interrupt if the RXOVIEN (SPIx_FIFOCTL[5]) is set to 1.

- Receive FIFO time-out interrupt

  If there is a received data in the FIFO buffer and it is not read by software over 64 SPI peripheral clock periods in Master mode or over 576 SPI peripheral clock periods in Slave mode, it will send a RX time-out interrupt to the system if the RX time-out interrupt enable bit, RXTOIEN (SPIx_FIFOCTL[4]), is set to 1.

- Transmit FIFO interrupt

  In FIFO mode, if the valid data count of the transmit FIFO buffer is less than or equal to the setting value of TXTH (SPIx_FIFOCTL[30:28]), the transmit FIFO interrupt flag TXTHIF (SPIx_STATUS[18]) will be set to 1. The SPI controller will generate a transmit FIFO interrupt to the system if the transmit FIFO interrupt enable bit, TXTHIEN (SPIx_FIFOCTL[3]), is set to 1.

- Receive FIFO interrupt

  In FIFO mode, if the valid data count of the receive FIFO buffer is larger than the setting value of RXTH (SPIx_FIFOCTL[26:24]), the receive FIFO interrupt flag RXTHIF (SPIx_STATUS[10]) will be set to 1. The SPI controller will generate a receive FIFO interrupt to the system if the receive FIFO interrupt enable bit, RXTHIEN (SPIx_FIFOCTL[2]), is set to 1.

6.27.5.10 I²S Mode

The SPI0~SPI3 controllers support I²S mode with PCM mode A, PCM mode B, MSB justified and I²S data format. The bit count of an audio channel is determined by WDWIDTH (SPIx_I2SCTL[5:4]). The

transfer sequence is always first from the most significant bit, MSB. Data are read on rising clock edge and are driven on falling clock edge.

In I²S data format, the MSB is sent and latched on the second clock of an audio channel. The I2Sx_LRCLK signal indicates which audio channel is in transferring.

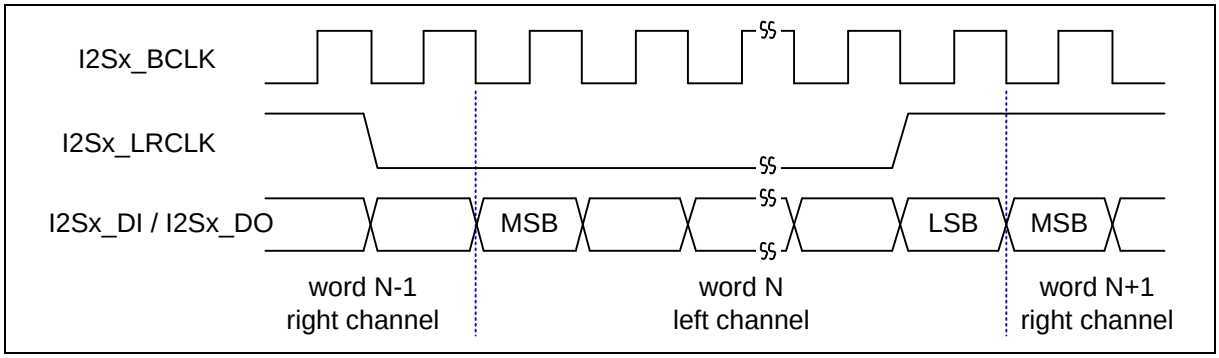


Figure 6.27-20 I²S Data Format Timing Diagram

In MSB justified data format, the MSB is sent and latched on the first clock of an audio channel.

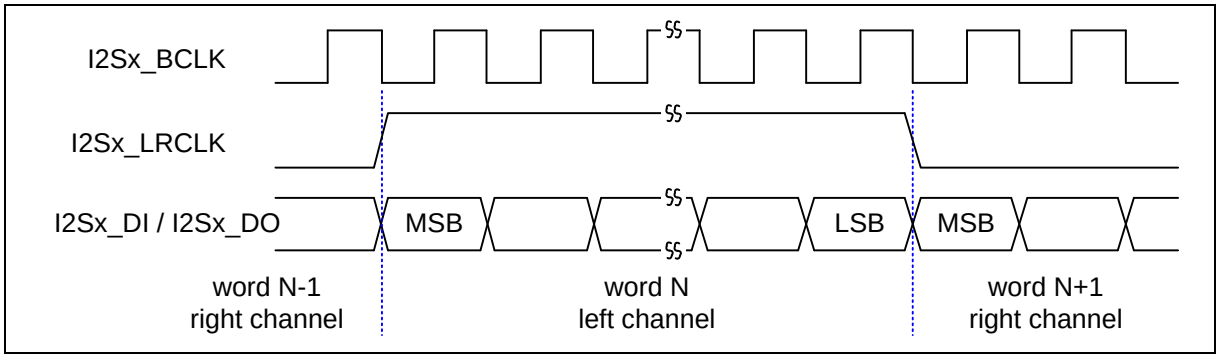


Figure 6.27-21 MSB Justified Data Format Timing Diagram

The I2Sx_LRCLK signal also supports PCM mode A and PCM mode B. The I2Sx_LRCLK signal in PCM mode indicates the beginning of an audio frame.

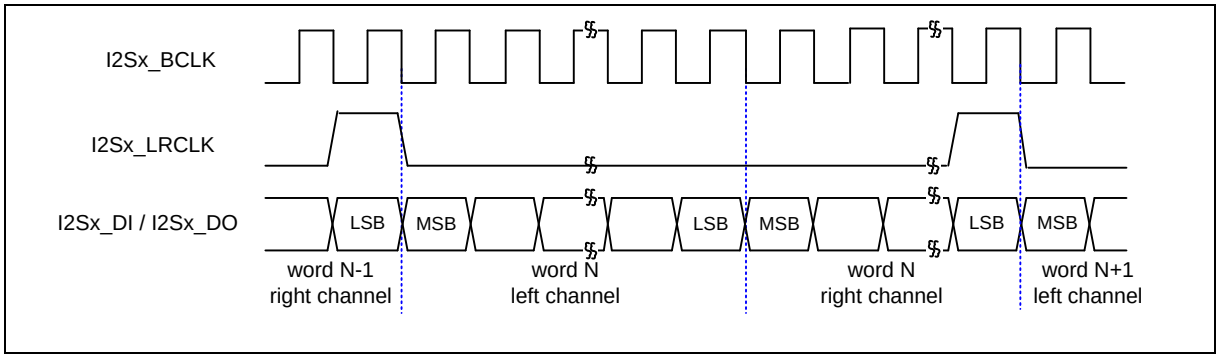


Figure 6.27-22 PCM Mode A Timing Diagram

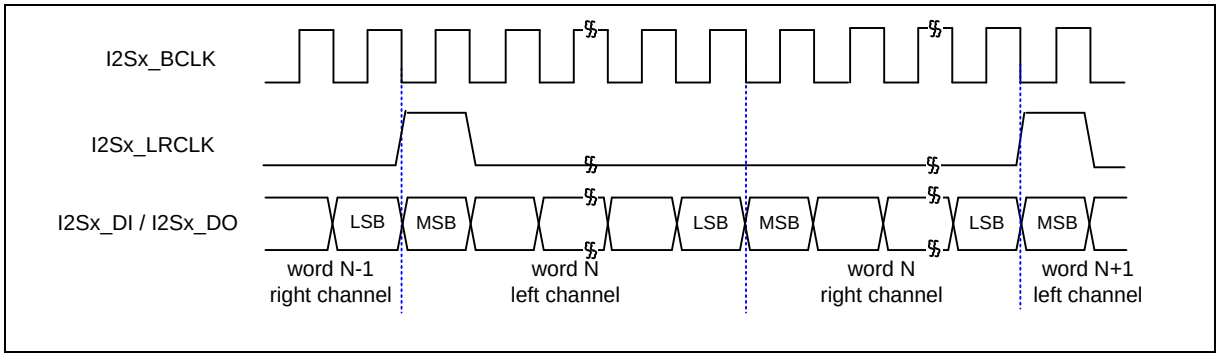


Figure 6.27-23 PCM Mode B Timing Diagram

6.27.5.11 I²S Mode FIFO Operation

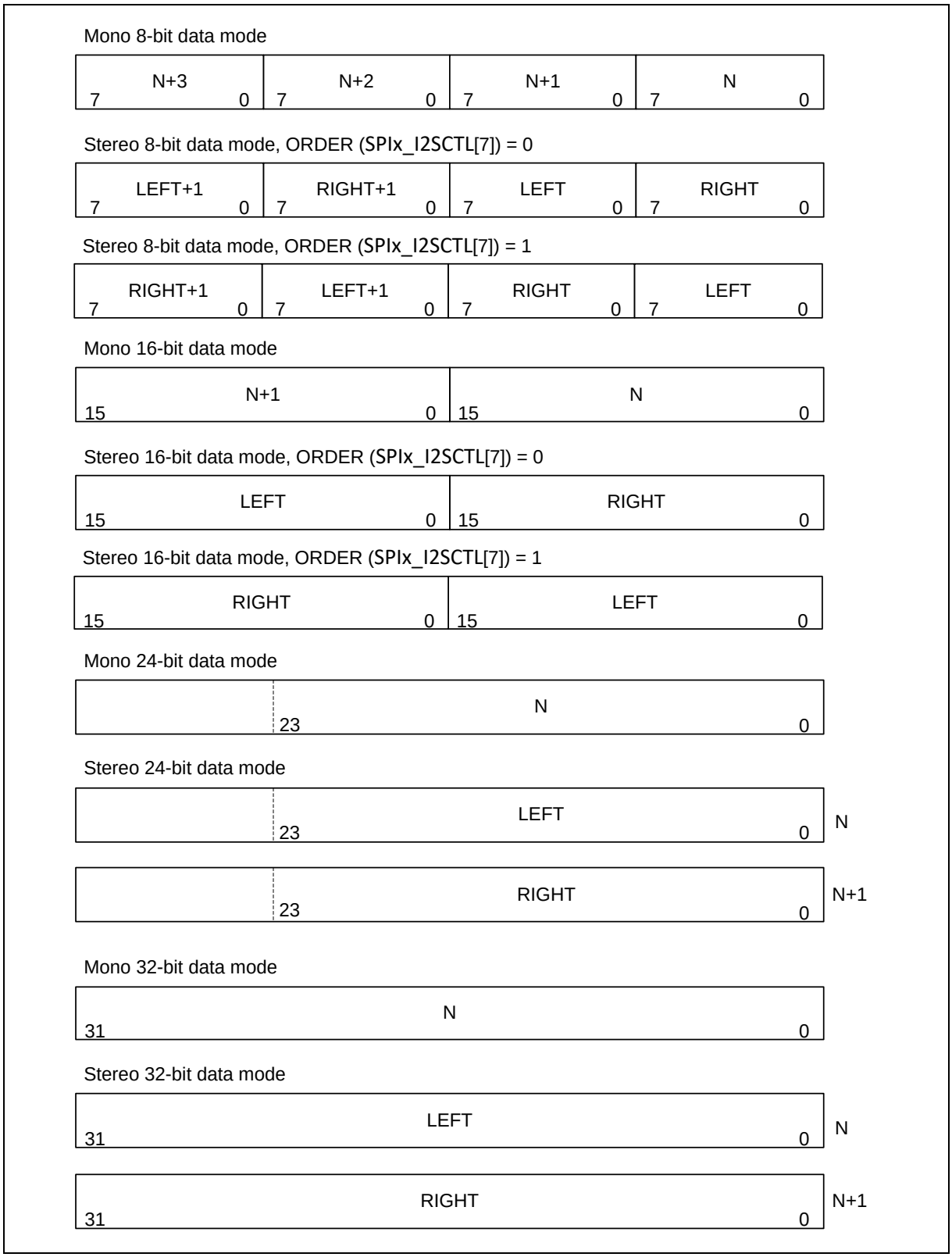


Figure 6.27-24 FIFO Contents for Various I²S Modes

6.27.5.12 Dummy Data Number for I²S / PCM Master mode and Monaural Mode

Before I²S / PCM master starts to send TX data to an external slave device, user sets control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, and write TX data to TX FIFO. After master sends dummy data (data with zero value) to an external slave device, master will send TX FIFO data to an external slave device. Table 6.27-2 shows the number of dummy data for monaural mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	0
16 bits	0
24 bits	1
32 bits	1

Table 6.27-2 Dummy Data Number for I²S / PCM Master Mode and Monaural Mode

6.27.5.13 Dummy Data Number for I²S / PCM Master mode and Stereo Mode

Before I²S / PCM master starts to send TX data to an external slave device, user sets control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, and write TX data to TX FIFO. After master sends dummy data (data with zero value) to an external slave device, master will send TX FIFO data to an external slave device. Table 6.27-3 shows the number of dummy data for stereo mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	0
16 bits	0
24 bits	1
32 bits	1

Table 6.27-3 Dummy Data Number for I²S / PCM Master Mode and Stereo Mode

6.27.5.14 Dummy Data Number for I²S Slave mode and Monaural Mode

Before I²S / PCM slave starts to send TX data to an external master device, user sets control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, SLAVE mode (SPIx_I2SCTL[8]), and write TX data to TX FIFO. After slave sends dummy data (data with zero value) to an external master device, slave will send TX FIFO data to an external master device. Table 6.27-4 shows the number of dummy data for I²S slave monaural mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	3
16 bits	2
24 bits	2
32 bits	2

Table 6.27-4 Dummy Data Number for I²S Slave Mode and Monaural Mode

6.27.5.15 Dummy Data Number for PCM Slave Mode and Monaural Mode

Before I²S / PCM slave starts to send TX data to an external master device, user sets control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, SLAVE mode (SPIx_I2SCTL[8]), and

write TX data to TX FIFO. After slave sends dummy data (data with zero value) to an external master device, slave will send TX FIFO data to an external master device. Table 6.27-5 shows the number of dummy data for PCM slave monaural mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	2
16 bits	1
24 bits	1
32 bits	1

Table 6.27-5 Dummy Data Number for PCM Slave Mode and Monaural Mode

6.27.5.16 Dummy Data Number for I²S Slave mode and Stereo Mode

Before I²S / PCM slave starts to send TX data to an external master device, user sets control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, SLAVE mode (SPIx_I2SCTL[8]), and write TX data to TX FIFO. After slave sends dummy data (data with zero value) to an external master device, slave will send TX FIFO data to an external master device. Table 6.27-6 shows number of dummy data for I²S slave stereo mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	3
16 bits	2
24 bits	2
32 bits	2

Table 6.27-6 Dummy Data Number for I²S Slave Mode and Stereo Mode

6.27.5.17 Dummy Data Number for PCM Slave mode and Stereo Mode

Before I²S / PCM slave starts to send TX data to an external master device, user sets control registers I2SEN (SPIx_I2SCTL[0]) enable, TXEN (SPIx_I2SCTL[1]) enable, SLAVE mode (SPIx_I2SCTL[8]), and write TX data to TX FIFO. After slave sends dummy data (data with zero value) to an external master device, slave will send TX FIFO data to an external master device. Table 6.27-7 shows number of dummy data for PCM slave stereo mode, and the unit of dummy data number is L channel + R channel.

Data Width (SPIx_I2SCTL[5:4])	Dummy Data Number (Unit = L Channel + R Channel)
8 bits	2
16 bits	1
24 bits	1
32 bits	1

Table 6.27-7 Dummy Data Number for PCM Slave Mode and Stereo Mode

6.27.6 Timing Diagram

The active state of slave selection signal can be defined by setting the SSACTPOL (SPIx_SSCTL[2]). The SPI clock which is in idle state can be configured as high or low state by setting the CLKPOL (SPIx_CTL[3]). It also provides the bit length of a transaction word in DWIDTH (SPIx_CTL[12:8]), and transmitting/receiving data from MSB or LSB first in LSB (SPIx_CTL[13]). User can also select which

edge of SPI clock to transmit/receive data in TXNEG/RXNEG (SPIx_CTL[2:1]). Four SPI timing diagrams for master/slave operations and the related settings are shown below.

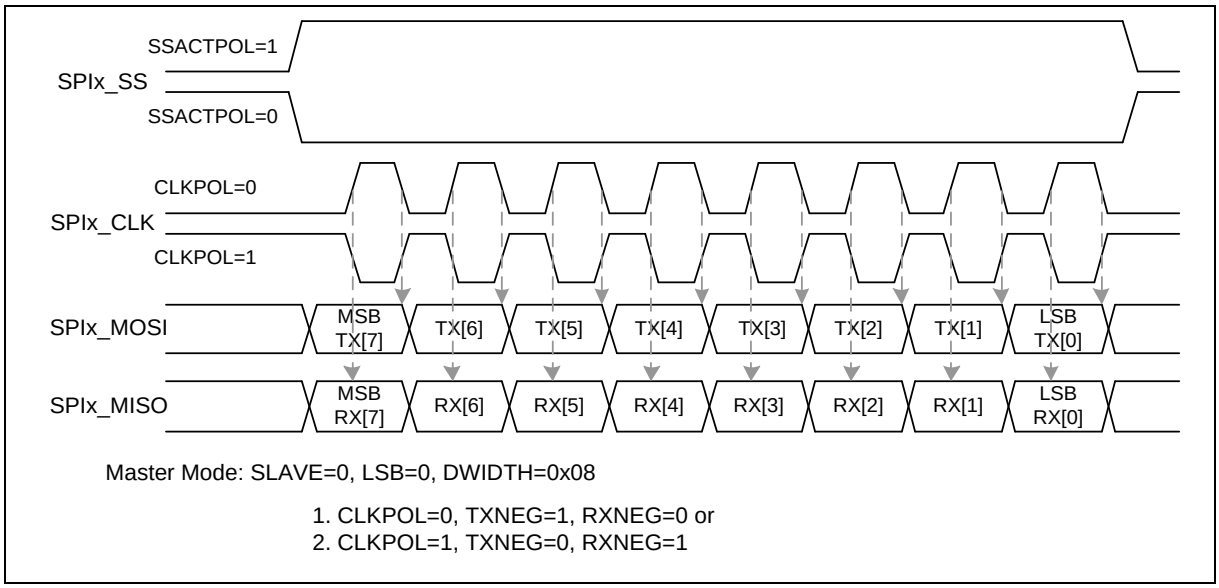


Figure 6.27-25 SPI Timing in Master Mode

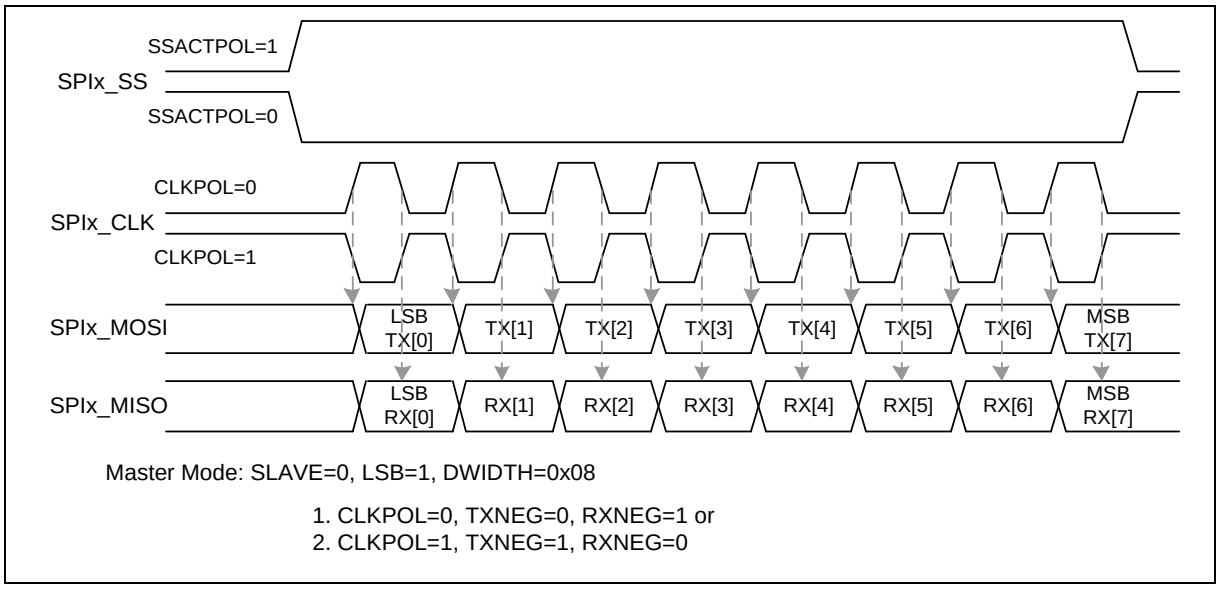


Figure 6.27-26 SPI Timing in Master Mode (Alternate Phase of SPIx_CLK)

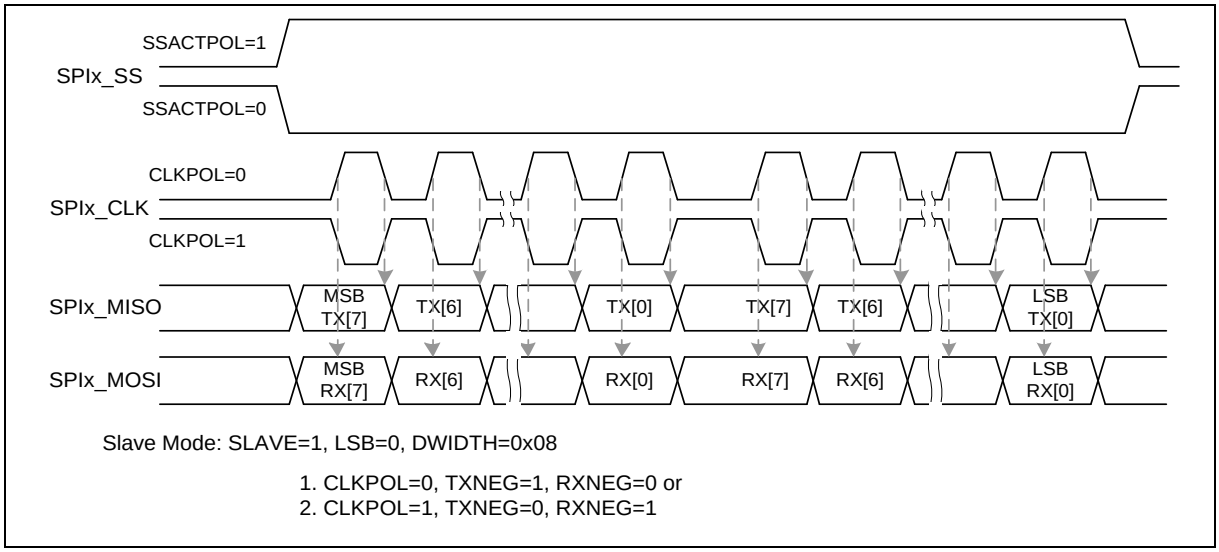


Figure 6.27-27 SPI Timing in Slave Mode

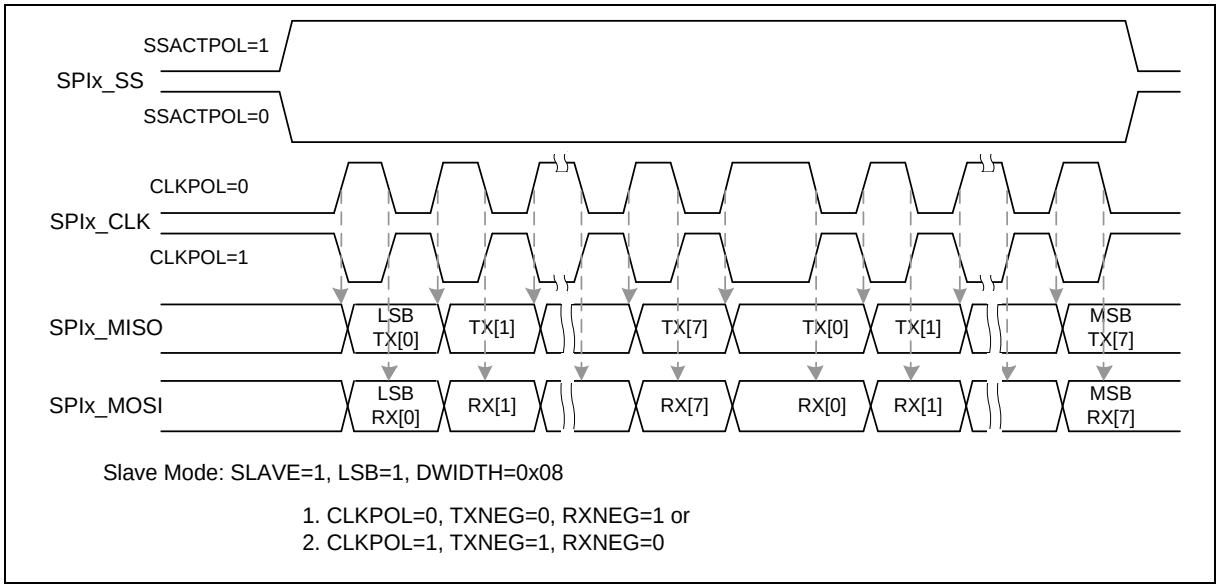


Figure 6.27-28 SPI Timing in Slave Mode (Alternate Phase of SPIx_CLK)

6.27.7 Programming Examples

Example 1:

The SPI controller is set as a full-duplex master to access an off-chip slave device with the following specifications:

- Data bit is latched on positive edge of SPI bus clock.
- Data bit is driven on negative edge of SPI bus clock.
- Data is transferred from MSB first.
- SPI bus clock is idle at low state.
- Only one byte of data to be transmitted/received in a transaction.
- Uses the first SPI slave select pin to connect with an off-chip slave device. The slave

selection signal is active low.

The operation flow is as follows:

1. Set DIVIDER (SPIx_CLKDIV[8:0]) to determine the output frequency of SPI clock.
2. Write the SPIx_SSCTL register a proper value for the related settings of Master mode:
   1) Clear AUTOSS (SPIx_SSCTL[3]) to 0 to disable the Automatic Slave Selection function.
   2) Configure slave selection signal as active low by clearing SSACTPOL (SPIx_SSCTL[2]) to 0.
   3) Enable slave selection signal by setting SS (SPIx_SSCTL[0]) to 1 to activate the off-chip slave device.
3. Write the related settings into the SPIx_CTL register to control the SPI master actions.
   1) Configure this SPI controller as master device by setting SLAVE (SPIx_CTL[18]) to 0.
   2) Force the SPI clock idle state at low by clearing CLKPOL (SPIx_CTL[3]) to 0.
   3) Select data transmitted on negative edge of SPI bus clock by setting TXNEG (SPIx_CTL[2]) to 1.
   4) Select data latched on positive edge of SPI bus clock by clearing RXNEG (SPIx_CTL[1]) to 0.
   5) Set the bit length of a transaction as 8-bit in DWIDTH bit field (SPIx_CTL[12:8] = 0x08).
   6) Set MSB transfer first by clearing LSB (SPIx_CTL[13]) to 0.
4. Set SPIEN (SPIx_CTL[0]) to 1 to enable the data transfer with the SPI interface.
5. If this SPI master attempts to transmit (write) one byte data to the off-chip slave device, write the byte data that will be transmitted into the SPIx_TX register.
6. Waiting for SPI interrupt if the UNITIEN (SPIx_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (SPIx_STATUS[1]).
7. Read out the received one byte data from SPIx_RX register.
8. Go to 5 to continue another data transfer or set SS (SPIx_SSCTL[0]) to 0 to inactivate the off-chip slave device.

Example 2:

The SPI controller is set as a full-duplex slave device and connects with an off-chip master device. The off-chip master device communicates with the on-chip SPI slave controller through the SPI interface with the following specifications:

- Data bit is latched on positive edge of SPI bus clock.
- Data bit is driven on negative edge of SPI bus clock.
- Data is transferred from LSB first.
- SPI bus clock is idle at high state.
- Only one byte of data to be transmitted/received in a transaction.
- Slave selection signal is active high.

The operation flow is as follows:

1. Write the SPIx_SSCTL register a proper value for the related settings of Slave mode.
2. Select high level for the input of slave selection signal by setting SSACTPOL (SPIx_SSCTL[2]) to 1.
3. Write the related settings into the SPIx_CTL register to control this SPI slave actions

1) Set the SPI controller as slave device by setting SLAVE (SPIx_CTL[18]) to 1.
2) Select the SPI clock idle state at high by setting CLKPOL (SPIx_CTL[3]) to 1.
3) Select data transmitted on negative edge of SPI bus clock by setting TXNEG (SPIx_CTL[2]) to 1.
4) Select data latched on positive edge of SPI bus clock by clearing RXNEG (SPIx_CTL[1]) to 0.
5) Set the bit length of a transaction as 8-bit in DWIDTH bit field (SPIx_CTL[12:8] = 0x08).

4. Set LSB transfer first by setting LSB (SPIx_CTL[13]) to 1.
5. Set the SPIEN (SPIx_CTL[0]) to 1. Wait for the slave select trigger input and SPI clock input from the off-chip master device to start the data transfer.
6. If this SPI slave attempts to transmit (be read) one byte data to the off-chip master device, write the byte data that will be transmitted into the SPIx_TX register.
7. If this SPI slave just only attempts to receive (be written) one byte data from the off-chip master device and does not care what data will be transmitted, the SPIx_TX register does not need to be updated by software.
8. Waiting for SPI interrupt if the UNITIEN (SPIx_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (SPIx_STATUS[1]).
9. Read out the received one byte data from SPIx_RX register.
10. Go to 6 to continue another data transfer or stop data transfer.

6.27.8 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SPI Base Address: **SPIx_BA = 0x4006_1000 + (0x0000_1000 * x)** **x=0, 1, 2, 3**				
SPIx_CTL	SPIx_BA+0x00	R/W	SPI Control Register	0x0000_0034
SPIx_CLKDIV	SPIx_BA+0x04	R/W	SPI Clock Divider Register	0x0000_0000
SPIx_SSCTL	SPIx_BA+0x08	R/W	SPI Slave Select Control Register	0x0000_0000
SPIx_PDMACTL	SPIx_BA+0x0C	R/W	SPI PDMA Control Register	0x0000_0000
SPIx_FIFOCTL	SPIx_BA+0x10	R/W	SPI FIFO Control Register	0x2200_0000
SPIx_STATUS	SPIx_BA+0x14	R/W	SPI Status Register	0x0005_0110
SPIx_STATUS2	SPIx_BA+0x18	R	SPI Status2 Register	0x0000_0000
SPIx_TX	SPIx_BA+0x20	W	SPI Data Transmit Register	0x0000_0000
SPIx_RX	SPIx_BA+0x30	R	SPI Data Receive Register	0x0000_0000
SPIx_I2SCTL	SPIx_BA+0x60	R/W	I^2S Control Register	0x0000_0000
SPIx_I2SCLK	SPIx_BA+0x64	R/W	I^2S Clock Divider Control Register	0x0000_0000
SPIx_I2SSTS	SPIx_BA+0x68	R/W	I^2S Status Register	0x0005_0100

6.27.9 Register Description

SPI Control Register (SPIx_CTL)

Register	Offset	R/W	Description	Reset Value
SPIx_CTL	SPIx_BA+0x00	R/W	SPI Control Register	0x0000_0034

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved			**DATDIR**	**REORDER**	**SLAVE**	**UNITIEN**	**Reserved**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXONLY	**HALFDPX**	**LSB**	**DWIDTH**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SUSPITV				**CLKPOL**	**TXNEG**	**RXNEG**	**SPIEN**

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**DATDIR**	**Data Port Direction Control** This bit is used to select the data input/output direction in half-duplex transfer and Dual/Quad transfer 0 = SPI data is input direction. 1 = SPI data is output direction.
[19]	**REORDER**	**Byte Reorder Function Enable Bit** 0 = Byte Reorder function Disabled. 1 = Byte Reorder function Enabled. A byte suspend interval will be inserted among each byte. The period of the byte suspend interval depends on the setting of SUSPITV. **Note:** Byte Reorder function is only available if DWIDTH is defined as 16, 24, and 32 bits.
[18]	**SLAVE**	**Slave Mode Control** 0 = Master mode. 1 = Slave mode.
[17]	**UNITIEN**	**Unit Transfer Interrupt Enable Bit** 0 = SPI unit transfer interrupt Disabled. 1 = SPI unit transfer interrupt Enabled.
[16]	**Reserved**	Reserved.
[15]	**RXONLY**	**Receive-only Mode Enable Bit** This bit field is only available in Master mode. In receive-only mode, SPI Master will generate SPI bus clock continuously for receiving data bit from SPI slave device and assert the BUSY status. 0 = Receive-only mode Disabled. 1 = Receive-only mode Enabled.
[14]	**HALFDPX**	**SPI Half-duplex Transfer Enable Bit** This bit is used to select full-duplex or half-duplex for SPI transfer. The bit field DATDIR

		(SPIx_CTL[20]) can be used to set the data direction in half-duplex transfer. 0 = SPI operates in full-duplex transfer. 1 = SPI operates in half-duplex transfer.
[13]	**LSB**	**Send LSB First** 0 = The MSB, which bit of transmit/receive register depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, bit 0 of the SPI TX register, is sent first to the SPI data output pin, and the first bit received from the SPI data input pin will be put in the LSB position of the RX register (bit 0 of SPI_RX).
[12:8]	**DWIDTH**	**Data Width** This field specifies how many bits can be transmitted/received in one transaction. The minimum bit length is 4 bits and can up to 32 bits. DWIDTH = 0x04 …. 4 bits. DWIDTH = 0x05 …. 5 bits. DWIDTH = 0x06 …. 6 bits. DWIDTH = 0x07 …. 7 bits. DWIDTH = 0x08 …. 8 bits. DWIDTH = 0x09 …. 9 bits. …… DWIDTH = 0x1F …. 31 bits. DWIDTH = 0x00 …. 32 bits. **Note:** This bit field will decide the depth of TX/RX FIFO configuration in SPI mode**.** Therefore, changing this bit field will clear TX/RX FIFO by hardware automatically.
[7:4]	**SUSPITV**	**Suspend Interval** The four bits provide configurable suspend interval between two successive transmit/receive transaction in a transfer. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the suspend interval is obtained according to the following equation. (Master only) (SUSPITV[3:0] + 0.5) * period of SPI_CLK clock cycle Example: SUSPITV = 0x0 …. 0.5 SPI_CLK clock cycle. SUSPITV = 0x1 …. 1.5 SPI_CLK clock cycle. …… SUSPITV = 0xE …. 14.5 SPI_CLK clock cycle. SUSPITV = 0xF …. 15.5 SPI_CLK clock cycle.
[3]	**CLKPOL**	**Clock Polarity** 0 = SPI bus clock is idle low. 1 = SPI bus clock is idle high.
[2]	**TXNEG**	**Transmit on Negative Edge** 0 = Transmitted data output signal is changed on the rising edge of SPI bus clock. 1 = Transmitted data output signal is changed on the falling edge of SPI bus clock.
[1]	**RXNEG**	**Receive on Negative Edge** 0 = Received data input signal is latched on the rising edge of SPI bus clock. 1 = Received data input signal is latched on the falling edge of SPI bus clock.

[0]	**SPIEN**	**SPI Transfer Control Enable Bit** In Master mode, the transfer will start when there is data in the FIFO buffer after this bit is set to 1. In Slave mode, this device is ready to receive data when this bit is set to 1. 0 = Transfer control Disabled. 1 = Transfer control Enabled. **Note:** Before changing the configurations of SPIx_CTL, SPIx_CLKDIV, SPIx_SSCTL and SPIx_FIFOCTL registers, user shall clear the SPIEN (SPIx_CTL[0]) and confirm the SPIENSTS (SPIx_STATUS[15]) is 0.

SPI Clock Divider Register (SPIx_CLKDIV)

Register	Offset	R/W	Description	Reset Value
SPIx_CLKDIV	SPIx_BA+0x04	R/W	SPI Clock Divider Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							DIVIDER
7	6	5	4	3	2	1	0
DIVIDER							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**DIVIDER**	**Clock Divider** The value in this field is the frequency divider for generating the peripheral clock, f_{spi_eclk}, and the SPI bus clock of SPI Master. The frequency is obtained according to the following equation. $f_{spi_eclk} = \frac{f_{spi_clock_src}}{(DIVIDER+1)}$ where $f_{spi_clock_src}$ is the peripheral clock source, which is defined in the clock control register, CLK_CLKSEL2 or CLK_CLKSEL3. **Note 1:** Not supported in I²S mode. **Note 2:** The time interval must be larger than or equal to 8 peripheral clock cycles between releasing SPI IP software reset and setting this clock divider register.

Note: DIVIDER should be set carefully because the peripheral clock frequency must be slower than or equal to system frequency.

SPI Slave Select Control Register (SPIx_SSCTL)

Register	Offset	R/W	Description	Reset Value
SPIx_SSCTL	SPIx_BA+0x08	R/W	SPI Slave Select Control Register	0x0000_0000

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SSINAIEN	SSACTIEN	Reserved		SLVURIEN	SLVBEIEN
7	6	5	4	3	2	1	0
Reserved			SLV3WIRE	AUTOSS	SSACTPOL	Reserved	SS

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SSINAIEN**	**Slave Select Inactive Interrupt Enable Bit** 0 = Slave select inactive interrupt Disabled. 1 = Slave select inactive interrupt Enabled.
[12]	**SSACTIEN**	**Slave Select Active Interrupt Enable Bit** 0 = Slave select active interrupt Disabled. 1 = Slave select active interrupt Enabled.
[11:10]	**Reserved**	Reserved.
[9]	**SLVURIEN**	**Slave Mode TX Under Run Interrupt Enable Bit** 0 = Slave mode TX under run interrupt Disabled. 1 = Slave mode TX under run interrupt Enabled.
[8]	**SLVBEIEN**	**Slave Mode Bit Count Error Interrupt Enable Bit** 0 = Slave mode bit count error interrupt Disabled. 1 = Slave mode bit count error interrupt Enabled.
[7:5]	**Reserved**	Reserved.
[4]	**SLV3WIRE**	**Slave 3-wire Mode Enable Bit** In Slave 3-wire mode, the SPI controller can work with 3-wire interface including SPIx_CLK, SPIx_MISO and SPIx_MOSI pins. (Slave only) 0 = 4-wire bi-direction interface. 1 = 3-wire bi-direction interface. **Note:** The value of this register equals to control register SLAVE (SPIx_I2SCTL[8]) when I^2S mode is enabled.
[3]	**AUTOSS**	**Automatic Slave Selection Function Enable Bit** 0 = Automatic slave selection function Disabled. Slave selection signal will be asserted/de-asserted according to SS (SPIx_SSCTL[0]).(Master Only) 1 = Automatic slave selection function Enabled.

[2]	**SSACTPOL**	**Slave Selection Active Polarity** This bit defines the active polarity of slave selection signal (SPIx_SS). 0 = The slave selection signal SPIx_SS is active low. 1 = The slave selection signal SPIx_SS is active high.
[1]	**Reserved**	Reserved.
[0]	**SS**	**Slave Selection Control** This is Master only. If AUTOSS bit is cleared to 0, 0 = set the SPIx_SS line to inactive state. 1 = set the SPIx_SS line to active state. If the AUTOSS bit is set to 1, 0 = Keep the SPIx_SS line at inactive state. 1 = SPIx_SS line will be automatically driven to active state for the duration of data transfer, and will be driven to inactive state for the rest of the time. The active state of SPIx_SS is

SPI PDMA Control Register (SPIx_PDMACTL)

Register	Offset	R/W	Description	Reset Value
SPIx_PDMACTL	SPIx_BA+0x0C	R/W	SPI PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					PDMARST	RXPDMAEN	TXPDMAEN

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the PDMA control logic of the SPI controller. This bit will be automatically cleared to 0.
[1]	**RXPDMAEN**	**Receive PDMA Enable Bit** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[0]	**TXPDMAEN**	**Transmit PDMA Enable Bit** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled. **Note 1:** In SPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, RX PDMA function cannot be enabled prior to TX PDMA function. User can enable TX PDMA function firstly or enable both functions simultaneously. **Note 2:** In SPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, TX PDMA function cannot be disabled prior to RX PDMA function. User can disable RX PDMA function firstly or disable both functions simultaneously.

SPI FIFO Control Register (SPIx_FIFOCTL)

Register	Offset	R/W	Description	Reset Value
SPIx_FIFOCTL	SPIx_BA+0x10	R/W	SPI FIFO Control Register	0x2200_0000

31	30	29	28	27	26	25	24
Reserved	TXTH			Reserved	RXTH		
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					SLVBERX	TXFBCLR	RXFBCLR
7	6	5	4	3	2	1	0
TXUFIEN	TXUFPOL	RXOVIEN	RXTOIEN	TXTHIEN	RXTHIEN	TXRST	RXRST

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TXTH**	**Transmit FIFO Threshold** If the valid data count of the transmit FIFO buffer is less than or equal to the TXTH setting, the TXTHIF bit will be set to 1, else the TXTHIF bit will be cleared to 0. The MSB of this bit field is only meaningful while SPI mode 4~16 bits of data length.
[27]	**Reserved**	Reserved.
[26:24]	**RXTH**	**Receive FIFO Threshold** If the valid data count of the receive FIFO buffer is larger than the RXTH setting, the RXTHIF bit will be set to 1, else the RXTHIF bit will be cleared to 0. The MSB of this bit field is only meaningful while SPI mode 4~16 bits of data length.
[23:11]	**Reserved**	Reserved.
[10]	**SLVBERX**	**RX FIFO Write Data Enable Bit When Slave Mode Bit Count Error** This is SPI Slave mode Only. 0 = Uncompleted RX data will be dropped from RX FIFO when bit count error event happened in SPI Slave mode. 1 = Uncompleted RX data will be written into RX FIFO when bit count error event happened in SPI Slave mode. User can read SLVBENUM (SPIx_STATUS2[29:24]) to know that the effective bit number of uncompleted RX data when SPI slave bit count error happened.
[9]	**TXFBCLR**	**Transmit FIFO Buffer Clear** 0 = No effect. 1 = Clear transmit FIFO pointer. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1. **Note:** The TX shift register will not be cleared.
[8]	**RXFBCLR**	**Receive FIFO Buffer Clear** 0 = No effect. 1 = Clear receive FIFO pointer. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1.

		Note: The RX shift register will not be cleared.
[7]	**TXUFIEN**	**TX Underflow Interrupt Enable Bit** When TX underflow event occurs in Slave mode, TXUFIF (SPIx_STATUS[19]) will be set to 1. This bit is used to enable the TX underflow interrupt. 0 = Slave TX underflow interrupt Disabled. 1 = Slave TX underflow interrupt Enabled.
[6]	**TXUFPOL**	**TX Underflow Data Polarity** 0 = The SPI data out is keep 0 if there is TX underflow event in Slave mode. 1 = The SPI data out is keep 1 if there is TX underflow event in Slave mode. **Note 1:** The TX underflow event occurs if there is no any data in TX FIFO when the slave selection signal is active. **Note 2:** This bit should be set as 0 in I^2S mode. **Note 3:** When TX underflow event occurs, SPIx_MISO pin state will be determined by this setting even though TX FIFO is not empty afterward. Data stored in TX FIFO will be sent through SPIx_MISO pin in the next transfer frame.
[5]	**RXOVIEN**	**Receive FIFO Overrun Interrupt Enable Bit** 0 = Receive FIFO overrun interrupt Disabled. 1 = Receive FIFO overrun interrupt Enabled.
[4]	**RXTOIEN**	**Receive Time-out Interrupt Enable Bit** 0 = Receive time-out interrupt Disabled. 1 = Receive time-out interrupt Enabled.
[3]	**TXTHIEN**	**Transmit FIFO Threshold Interrupt Enable Bit** 0 = TX FIFO threshold interrupt Disabled. 1 = TX FIFO threshold interrupt Enabled.
[2]	**RXTHIEN**	**Receive FIFO Threshold Interrupt Enable Bit** 0 = RX FIFO threshold interrupt Disabled. 1 = RX FIFO threshold interrupt Enabled.
[1]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset transmit FIFO pointer and transmit circuit. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (SPIx_STATUS[23]) to check if reset is accomplished or not. **Note:** If TX underflow event occurs in SPI Slave mode, this bit can be used to make SPI return to idle state.
[0]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset receive FIFO pointer and receive circuit. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (SPIx_STATUS[23]) to check if reset is accomplished or not.

SPI Status Register (SPIx_STATUS)

Register	Offset	R/W	Description	Reset Value
SPIx_STATUS	SPIx_BA+0x14	R/W	SPI Status Register	0x0005_0110

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
TXCNT				RXCNT			
23	22	21	20	19	18	17	16
TXRXRST	Reserved			TXUFIF	TXTHIF	TXFULL	TXEMPTY
15	14	13	12	11	10	9	8
SPIENSTS	Reserved		RXTOIF	RXOVIF	RXTHIF	RXFULL	RXEMPTY
7	6	5	4	3	2	1	0
SLVURIF	SLVBEIF	Reserved	SSLINE	SSINAIF	SSACTIF	UNITIF	BUSY

Bits	Description	
[31:28]	**TXCNT**	**Transmit FIFO Data Count (Read Only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27:24]	**RXCNT**	**Receive FIFO Data Count (Read Only)** This bit field indicates the valid data count of receive FIFO buffer.
[23]	**TXRXRST**	**TX or RX Reset Status (Read Only)** 0 = The reset function of TXRST or RXRST is done. 1 = Doing the reset function of TXRST or RXRST. **Note:** Both the reset operations of TXRST and RXRST need 3 system clock cycles + 2 peripheral clock cycles. User can check the status of this bit to monitor the reset function is doing or done.
[22:20]	**Reserved**	Reserved.
[19]	**TXUFIF**	**TX Underflow Interrupt Flag** When the TX underflow event occurs, this bit will be set to 1, the state of data output pin depends on the setting of TXUFPOL. 0 = No effect. 1 = No data in Transmit FIFO and TX shift register when the slave selection signal is active. **Note 1:** This bit will be cleared by writing 1 to it. **Note 2:** If reset slave's transmission circuit when slave selection signal is active, this flag will be set to 1 after 2 peripheral clock cycles + 3 system clock cycles since the reset operation is done.
[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TXTH.
[17]	**TXFULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.

[16]	**TXEMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** 0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[15]	**SPIENSTS**	**SPI Enable Status (Read Only)** 0 = SPI controller Disabled. 1 = SPI controller Enabled. **Note:** The SPI peripheral clock is asynchronous with the system clock. In order to make sure the SPI control logic is disabled, this bit indicates the real status of SPI controller.
[14:13]	**Reserved**	Reserved.
[12]	**RXTOIF**	**Receive Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI peripheral clock periods in Master mode or over 576 SPI peripheral clock periods in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to it.
[11]	**RXOVIF**	**Receive FIFO Overrun Interrupt Flag** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. 0 = No FIFO is overrun. 1 = Receive FIFO is overrun. **Note:** This bit will be cleared by writing 1 to it.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the receive FIFO buffer is smaller than or equal to the setting value of RXTH. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH.
[9]	**RXFULL**	**Receive FIFO Buffer Full Indicator (Read Only)** 0 = Receive FIFO buffer is not full. 1 = Receive FIFO buffer is full.
[8]	**RXEMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[7]	**SLVURIF**	**Slave Mode TX Under Run Interrupt Flag** In Slave mode, if TX underflow event occurs and the slave select line goes to inactive state, this interrupt flag will be set to 1. 0 = No Slave TX under run event. 1 = Slave TX under run event occurred. **Note:** This bit will be cleared by writing 1 to it.
[6]	**SLVBEIF**	**Slave Mode Bit Count Error Interrupt Flag** In Slave mode, when the slave select line goes to inactive state, if bit counter is mismatch with DWIDTH, this interrupt flag will be set to 1. 0 = No Slave mode bit count error event. 1 = Slave mode bit count error event occurred. **Note:** If the slave select active but there is no any bus clock input, the SLVBEIF also active when the slave select goes to inactive state. This bit will be cleared by writing 1 to it.
[5]	**Reserved**	Reserved.
[4]	**SSLINE**	**Slave Select Line Bus Status (Read Only)**

		0 = The slave select line status is 0. 1 = The slave select line status is 1. **Note:** This bit is only available in Slave mode. If SSACTPOL (SPIx_SSCTL[2]) is set 0, and the SSLINE is 1, the SPI slave select is in inactive status.
[3]	**SSINAIF**	**Slave Select Inactive Interrupt Flag** 0 = Slave select inactive interrupt was cleared or not occurred. 1 = Slave select inactive interrupt event occurred. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[2]	**SSACTIF**	**Slave Select Active Interrupt Flag** 0 = Slave select active interrupt was cleared or not occurred. 1 = Slave select active interrupt event occurred. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[1]	**UNITIF**	**Unit Transfer Interrupt Flag** 0 = No transaction has been finished since this bit was cleared to 0. 1 = SPI controller has finished one unit transfer. **Note:** This bit will be cleared by writing 1 to it.
[0]	**BUSY**	**Busy Status (Read Only)** 0 = SPI controller is in idle state. 1 = SPI controller is in busy state. The following lists the bus busy conditions: e. SPIEN (SPIx_CTL[0]) = 1 and TXEMPTY = 0. f. For SPI Master mode, SPIEN (SPIx_CTL[0]) = 1 and TXEMPTY = 1 but the current transaction is not finished yet. g. For SPI Master mode, SPIEN (SPIx_CTL[0]) = 1 and RXONLY = 1. h. For SPI Slave mode, SPIEN (SPIx_CTL[0]) = 1 and there is serial clock input into the SPI core logic when slave select is active. **i.** For SPI Slave mode, SPIEN (SPIx_CTL[0]) = 1 and the transmit buffer or transmit shift register is not empty even if the slave select is inactive. **Note:** By applications, this SPI busy flag should be used with other status registers in SPIx_STATUS such as TXCNT, RXCNT, TXTHIF, TXFULL, TXEMPTY, RXTHIF, RXFULL, RXEMPTY, and UNITIF. Therefore the SPI transfer done events of TX/RX operations can be obtained at correct timing point.

SPI Status2 Register (SPIx_STATUS2)

Register	Offset	R/W	Description	Reset Value
SPIx_STATUS2	SPIx_BA+0x18	R	SPI Status2 Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		SLVBENUM					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**SLVBENUM**	**Effective Bit Number of Uncompleted RX data** This status register indicates that effective bit number of uncompleted RX data when SLVBERX (SPIx_FIFOCTL[10]) is enabled and RX bit count error event happen in SPI Slave mode. (SPI Slave mode only) This status register will be fixed to 0x0 when SLVBERX (SPIx_FIFOCTL[10]) is disabled. **Note:** This register will be cleared to 0x0 when user writes 0x1 to SLVBEIF (SPIx_STATUS[6]).
[23:0]	**Reserved**	Reserved.

SPI Data Transmit Register (SPIx_TX)

Register	Offset	R/W	Description	Reset Value
SPIx_TX	SPIx_BA+0x20	W	SPI Data Transmit Register	0x0000_0000

31	30	29	28	27	26	25	24
TX							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TX							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TX							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TX							

Bits	Description	
[31:0]	**TX**	**Data Transmit Register** The data transmit registers pass through the transmitted data into the 4-level transmit FIFO buffers. The number of valid bits depends on the setting of DWIDTH (SPIx_CTL[12:8]) in SPI mode or WDWIDTH (SPIx_I2SCTL[5:4]) in I^2S mode. In SPI mode, if DWIDTH is set to 0x08, the bits TX[7:0] will be transmitted. If DWIDTH is set to 0x00, the SPI controller will perform a 32-bit transfer. In I^2S mode, if WDWIDTH (SPIx_I2SCTL[5:4]) is set to 0x2, the data width of audio channel is 24-bit and corresponding to TX[23:0]. If WDWIDTH is set as 0x0, 0x1, or 0x3, all bits of this field are valid and referred to the data arrangement in I^2S mode FIFO operation section **Note:** In Master mode, SPI controller will start to transfer the SPI bus clock after 1 APB clock and 6 peripheral clock cycles after user writes to this register.

SPI Data Receive Register (SPIx_RX)

Register	Offset	R/W	Description	Reset Value
SPIx_RX	SPIx_BA+0x30	R	SPI Data Receive Register	0x0000_0000

31	30	29	28	27	26	25	24
RX							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RX							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RX							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RX							

Bits	Description	
[31:0]	**RX**	**Data Receive Register (Read Only)** There are 4-level FIFO buffers in this controller. The data receive register holds the data received from SPI data input pin. If the RXEMPTY (SPIx_STATUS[8] or SPIx_I2SSTS[8]) is not set to 1, the receive FIFO buffers can be accessed through software by reading this register.

I²S Control Register (SPIx_I2SCTL)

Register	Offset	R/W	Description	Reset Value
SPIx_I2SCTL	SPIx_BA+0x60	R/W	I²S Control Register	0x0000_0000

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
SLVERRIEN	Reserved	FORMAT		Reserved		LZCIEN	RZCIEN
23	22	21	20	19	18	17	16
RXLCH	Reserved					LZCEN	RZCEN
15	14	13	12	11	10	9	8
MCLKEN	Reserved						SLAVE
7	6	5	4	3	2	1	0
ORDER	MONO	WDWIDTH		MUTE	RXEN	TXEN	I2SEN

Bits	Description	
[31]	**SLVERRIEN**	**Bit Number Error Interrupt Enable Bit for Slave Mode** Interrupt occurs if this bit is set to 1 and bit number error event occurs. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[30]	**Reserved**	Reserved.
[29:28]	**FORMAT**	**Data Format Selection** 00 = I²S data format. 01 = MSB justified data format. 10 = PCM mode A. 11 = PCM mode B.
[27:26]	**Reserved**	Reserved.
[25]	**LZCIEN**	**Left Channel Zero Cross Interrupt Enable Bit** Interrupt occurs if this bit is set to 1 and left channel zero cross event occurs. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[24]	**RZCIEN**	**Right Channel Zero Cross Interrupt Enable Bit** Interrupt occurs if this bit is set to 1 and right channel zero cross event occurs. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[23]	**RXLCH**	**Receive Left Channel Enable Bit** When monaural format is selected (MONO = 1), I²S controller will receive right channel data if RXLCH is set to 0, and receive left channel data if RXLCH is set to 1. 0 = Receive right channel data in Mono mode. 1 = Receive left channel data in Mono mode.
[22:18]	**Reserved**	Reserved.

[17]	**LZCEN**	**Left Channel Zero Cross Detection Enable Bit** If this bit is set to 1, when left channel data sign bit changes or next shift data bits are all 0 then LZCIF flag in SPIx_I2SSTS register is set to 1. This function is only available in transmit operation. 0 = Left channel zero cross detection Disabled. 1 = Left channel zero cross detection Enabled.
[16]	**RZCEN**	**Right Channel Zero Cross Detection Enable Bit** If this bit is set to 1, when right channel data sign bit change or next shift data bits are all 0 then RZCIF flag in SPIx_I2SSTS register is set to 1. This function is only available in transmit operation. 0 = Right channel zero cross detection Disabled. 1 = Right channel zero cross detection Enabled.
[15]	**MCLKEN**	**Master Clock Enable Bit** If MCLKEN is set to 1, I^2S controller will generate master clock on SPIx_I2SMCLK pin for external audio devices. 0 = Master clock Disabled. 1 = Master clock Enabled.
[14:9]	**Reserved**	Reserved.
[8]	**SLAVE**	**Slave Mode** I^2S can operate as master or slave. For Master mode, I2Sx_BCLK and I2Sx_LRCLK pins are output mode and send bit clock from this chip to audio CODEC chip. In Slave mode, I2Sx_BCLK and I2Sx_LRCLK pins are input mode and I2Sx_BCLK and I2Sx_LRCLK signals are received from outer audio CODEC chip. 0 = Master mode. 1 = Slave mode.
[7]	**ORDER**	**Stereo Data Order in FIFO** 0 = Left channel data at high byte. 1 = Left channel data at low byte.
[6]	**MONO**	**Monaural Data** 0 = Data is stereo format. 1 = Data is monaural format.
[5:4]	**WDWIDTH**	**Word Width** 00 = data size is 8-bit. 01 = data size is 16-bit. 10 = data size is 24-bit. 11 = data size is 32-bit.
[3]	**MUTE**	**Transmit Mute Enable Bit** 0 = Transmit data is shifted from buffer. 1 = Transmit channel zero.
[2]	**RXEN**	**Receive Enable Bit** 0 = Data receive Disabled. 1 = Data receive Enabled.
[1]	**TXEN**	**Transmit Enable Bit** 0 = Data transmit Disabled. 1 = Data transmit Enabled.

[0]	**I2SEN**	**I²S Controller Enable Bit** 0 = I²S mode Disabled. 1 = I²S mode Enabled. **Note 1:** If enabling this bit, I2Sx_BCLK will start to output in Master mode. **Note 2:** Before changing the configurations of SPIx_I2SCTL, SPIx_I2SCLK, and SPIx_FIFOCTL registers, user shall clear the I2SEN (SPIx_I2SCTL[0]) and confirm the I2SENSTS (SPIx_I2SSTS[15]) is 0.

I²S Clock Divider Control Register (SPIx_I2SCLK)

Register	Offset	R/W	Description	Reset Value
SPIx_I2SCLK	SPIx_BA+0x64	R/W	I²S Clock Divider Control Register	0x0000_0000

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
Reserved						I2SSLAVE	I2SMODE
23	22	21	20	19	18	17	16
Reserved						BCLKDIV	
15	14	13	12	11	10	9	8
BCLKDIV							
7	6	5	4	3	2	1	0
Reserved	MCLKDIV						

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**I2SSLAVE**	**I²S Clock Divider Number Selection for I²S Slave Mode and I²S Master Mode** User sets I2SSLAVE to set frequency of peripheral clock of I²S Master mode and I²S Slave mode when BCLKDIV (SPIx_I2SCLK[17:8]) is set. I2SSLAVE needs to set before I2SEN (SPIx_I2SCTL[0]) is enabled. 0 = The frequency of peripheral clock is set to I²S Master mode. 1 = The frequency of peripheral clock is set to I²S Slave mode.
[24]	**I2SMODE**	**I²S Clock Divider Number Selection for I²S Mode and SPI Mode** User sets I2SMODE to set frequency of peripheral clock of I²S mode or SPI mode when BCLKDIV (SPIx_I2SCLK[17:8]) or DIVIDER (SPIx_CLKDIV[8:0]) is set. User needs to set I2SMODE before I2SEN (SPIx_I2SCTL[0]) or SPIEN (SPIx_CTL[0]) is enabled. 0 = The frequency of peripheral clock is set to SPI mode. 1 = The frequency of peripheral clock is set to I²S mode.
[23:18]	**Reserved**	Reserved.

[17:8]	**BCLKDIV**	**Bit Clock Divider** The I²S controller will generate bit clock in Master mode. The clock frequency of bit clock, f_{BCLK}, is determined by the following expression: $f_{BCLK} = \frac{f_{i2s_clock_src}}{2 \times (BCLKDIV + 1)}$ where $f_{i2s_clock_src}$ is the frequency of I²S peripheral clock source, which is defined in the clock control register, CLK_CLKSEL2 or CLK_CLKSEL3. In I²S Slave mode, this field is used to define the frequency of peripheral clock and it's determined by $f_{i2s_clock_src} \div \left(\frac{BCLKDIV}{2} + 1\right)$. The peripheral clock frequency in I²S Slave mode must be equal to or faster than 6 times of input bit clock. **Note:** The time interval must be larger than or equal to 8 peripheral clock cycles between releasing SPI IP software reset and setting this clock divider register.
[7]	**Reserved**	Reserved.
[6:0]	**MCLKDIV**	**Master Clock Divider** If MCLKEN is set to 1, I²S controller will generate master clock for external audio devices. The frequency of master clock, f_{MCLK}, is determined by the following expressions: If MCLKDIV >= 1,. $f_{MCLK} = \frac{f_{i2s_clock_src}}{2 \times MCLKDIV}$ If MCLKDIV = 0,. $f_{MCLK} = f_{i2s_clock_src}$ where $f_{i2s_clock_src}$ is the frequency of I²S peripheral clock source, which is defined in the clock control register, CLK_CLKSEL2 or CLK_CLKSEL3. In general, the master clock rate is 256 times sampling clock rate.

Note: BCLKDIV should be set carefully because the peripheral clock frequency must be slower than or equal to system frequency.

I²S Status Register (SPIx_I2SSTS)

Register	Offset	R/W	Description	Reset Value
SPIx_I2SSTS	SPIx_BA+0x68	R/W	I²S Status Register	0x0005_0100

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
Reserved	TXCNT			Reserved	RXCNT		
23	22	21	20	19	18	17	16
TXRXRST	SLVERRIF	LZCIF	RZCIF	TXUFIF	TXTHIF	TXFULL	TXEMPTY
15	14	13	12	11	10	9	8
I2SENSTS	Reserved		RXTOIF	RXOVIF	RXTHIF	RXFULL	RXEMPTY
7	6	5	4	3	2	1	0
Reserved			RIGHT	Reserved			

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TXCNT**	**Transmit FIFO Data Count (Read Only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27]	**Reserved**	Reserved.
[26:24]	**RXCNT**	**Receive FIFO Data Count (Read Only)** This bit field indicates the valid data count of receive FIFO buffer.
[23]	**TXRXRST**	**TX or RX Reset Status (Read Only)** 0 = The reset function of TXRST or RXRST is done. 1 = Doing the reset function of TXRST or RXRST. **Note:** Both the reset operations of TXRST and RXRST need 3 system clock cycles + 2 peripheral clock cycles. User can check the status of this bit to monitor the reset function is doing or done.
[22]	**SLVERRIF**	**Bit Number Error Interrupt Flag for Slave Mode** 0 = No bit number error event occurred. 1 = Bit number error event occurred. **Note:** This bit will be cleared by writing 1 to it.
[21]	**LZCIF**	**Left Channel Zero Cross Interrupt Flag** 0 = No zero cross event occurred on left channel. 1 = Zero cross event occurred on left channel.
[20]	**RZCIF**	**Right Channel Zero Cross Interrupt Flag** 0 = No zero cross event occurred on right channel. 1 = Zero cross event occurred on right channel.
[19]	**TXUFIF**	**Transmit FIFO Underflow Interrupt Flag** When the transmit FIFO buffer is empty and there is no datum written into the FIFO buffer, if there is more bus clock input, this bit will be set to 1. **Note:** This bit will be cleared by writing 1 to it.

[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TXTH. **Note:** If TXTHIEN = 1 and TXTHIF = 1, the SPI/I^{2}S controller will generate a SPI interrupt request.
[17]	**TXFULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.
[16]	**TXEMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** 0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[15]	**I2SENSTS**	**I^{2}S Enable Status (Read Only)** 0 = The SPI/I^{2}S control logic is disabled. 1 = The SPI/I^{2}S control logic is enabled. **Note:** The SPI peripheral clock is asynchronous with the system clock. In order to make sure the SPI/I^{2}S control logic is disabled, this bit indicates the real status of SPI/I^{2}S control logic for user.
[14:13]	**Reserved**	Reserved.
[12]	**RXTOIF**	**Receive Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI peripheral clock period in Master mode or over 576 SPI peripheral clock period in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to it.
[11]	**RXOVIF**	**Receive FIFO Overrun Interrupt Flag** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. **Note:** This bit will be cleared by writing 1 to it.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the receive FIFO buffer is smaller than or equal to the setting value of RXTH. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH. **Note:** If RXTHIEN = 1 and RXTHIF = 1, the SPI/I^{2}S controller will generate a SPI interrupt request.
[9]	**RXFULL**	**Receive FIFO Buffer Full Indicator (Read Only)** 0 = Receive FIFO buffer is not full. 1 = Receive FIFO buffer is full.
[8]	**RXEMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[7:5]	**Reserved**	Reserved.
[4]	**RIGHT**	**Right Channel (Read Only)** This bit indicates the current transmit data is belong to which channel. 0 = Left channel.

		1 = Right channel.
[3:0]	**Reserved**	Reserved.

6.28 Quad Serial Peripheral Interface (QSPI)

6.28.1 Overview

The Quad Serial Peripheral Interface (QSPI) applies to synchronous serial data communication and allows full duplex transfer. Devices communicate in Master/Slave mode with the 4-wire bi-direction interface. The chip contains two QSPI controllers performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device.

The QSPI controller supports 2-bit Transfer mode to perform full-duplex 2-bit data transfer and also supports Dual and Quad I/O Transfer mode and the controller supports the PDMA function to access the data buffer.

6.28.2 Features

- Supports Master or Slave mode operation
- Master mode up to 100 MHz (when chip works at V_{DD} = 2.7V~3.6V)
- Slave mode up to 50 MHz (when chip works at V_{DD} = 2.7V~3.6V)
- Supports 2-bit Transfer mode
- Supports Dual and Quad I/O Transfer mode
- Configurable bit length of a transaction word from 8 to 32-bit
- Provides separate 8-level depth transmit and receive FIFO buffers
- Supports MSB first or LSB first transfer sequence
- Supports Byte Reorder function
- Supports Byte or Word Suspend mode
- Supports PDMA transfer
- Supports 3-Wire, no slave selection signal, bi-direction interface
- Supports one data channel half-duplex transfer
- Supports Transmit Double Transfer Rate Mode (TX DTR mode)
- Supports receive-only mode

6.28.3 Block Diagram

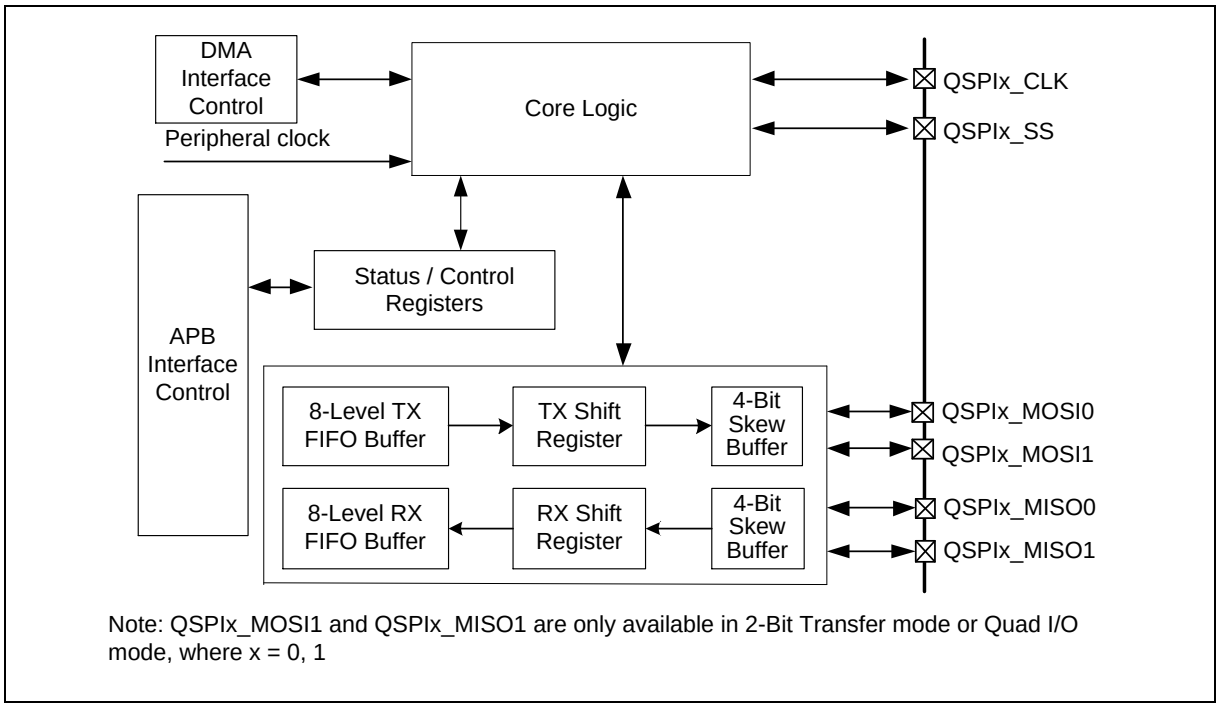


Figure 6.28-1 QSPI Block Diagram

TX FIFO Buffer:

The transmit FIFO buffer is an 8-level depth, 32-bit wide, first-in, first-out register buffer. The data can be written to the transmit FIFO buffer in advance through software by writing the QSPIx_TX register.

RX FIFO Buffer:

The receive FIFO buffer is also an 8-level depth, 32-bit wide, first-in, first-out register buffer. The receive control logic will store the receive data to this buffer. The FIFO buffer data can be read from QSPIx_RX register by software.

TX Shift Register:

The transmit shift register is a 32-bit wide register buffer. The transmit data is loaded from the TX FIFO buffer and shifted out bit-by-bit to the skew buffer.

RX Shift Register:

The receive shift register is also a 32-bit wide register buffer. The receive data is shift in bit-by-bit from the skew buffer and is loaded into RX FIFO buffer when a transaction done.

Skew Buffer:

The skew buffer is a 4-level depth, 1-bit wide buffer. There are two skew buffers in transmitting and received side. In received side, it is used to shift bits into RX shift register from QSPI bus. In transmitting side, it is used to shift bits into QSPI bus from TX shift register.

6.28.4 Basic Configuration

6.28.4.1 QSPI0 Basic Configuration

- Clock source Configuration

- Select the source of QSPI0 peripheral clock on QSPI0SEL (CLK_CLKSEL2[3:2]).
- Enable QSPI0 peripheral clock in QSPI0CKEN (CLK_APBCLK0[12]).

- Reset Configuration
  - Reset QSPI0 controller in QSPI0RST (SYS_IPRST1[12]).

6.28.4.2 QSPI1 Basic Configuration

- Clock source Configuration
  - Select the source of QSPI1 peripheral clock on QSPI1SEL (CLK_CLKSEL2[11:10]).
  - Enable QSPI1 peripheral clock in QSPI1CKEN (CLK_APBCLK1[4]).
- Reset Configuration
  - Reset QSPI1 controller in QSPI1RST (SYS_IPRST2[4]).

6.28.5 Functional Description

6.28.5.1 Terminology

QSPI Peripheral Clock and QSPI Bus Clock

The QSPI controller needs the peripheral clock to drive the QSPI logic unit to perform the data transfer. The peripheral clock rate is determined by the settings of clock divisor (QSPIx_CLKDIV) and the clock source which can be HXT, PLL, PCLK or HIRC. QSPIxSEL of CLK_CLKSEL2 register determines the clock source of the peripheral clock. The DIVIDER (QSPIx_CLKDIV[8:0]) setting determines the divisor of the clock rate calculation.

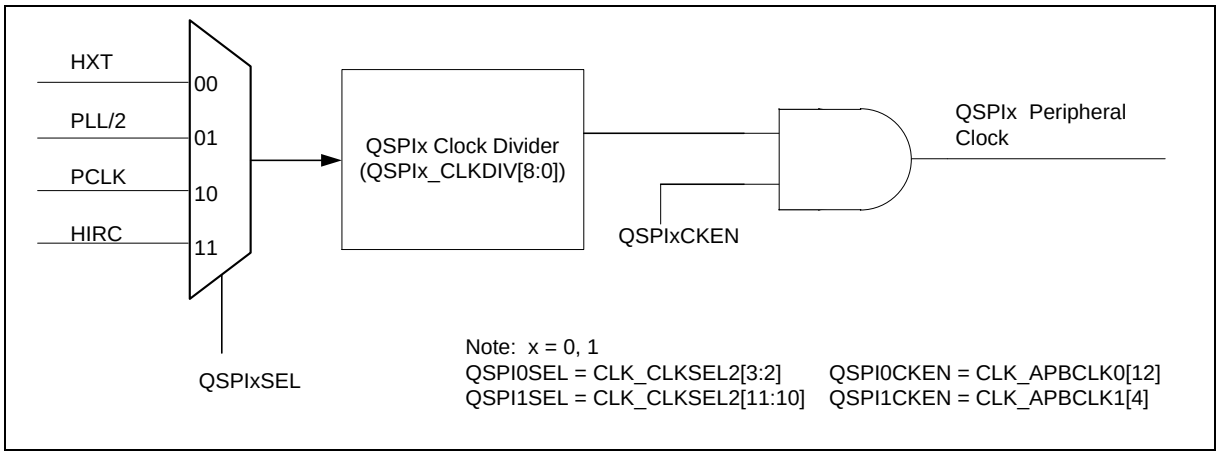


Figure 6.28-2 QSPI Peripheral Clock

In Master mode, the frequency of the QSPI bus clock is equal to the peripheral clock rate. In general, the QSPI bus clock is denoted as QSPI clock. In Slave mode, the QSPI bus clock is provided by a master device. The frequency of QSPI peripheral clock cannot be faster than the system clock rate regardless of Master or Slave mode. If the clock source of peripheral clock is not system clock, the frequency of QSPI peripheral clock shall be slower than the system clock frequency regardless of Master or Slave mode.

Master/Slave mode

The QSPI controllers can be set as Master or Slave mode by setting the SLAVE (QSPIx_CTL[18]) to communicate with the off-chip SPI slave or master device. The HALFDPX (QSPIx_CTL[14]) can be used to select the full-duplex or half-duplex in QSPI transmission. The application block diagrams in Master and Slave mode are shown below.

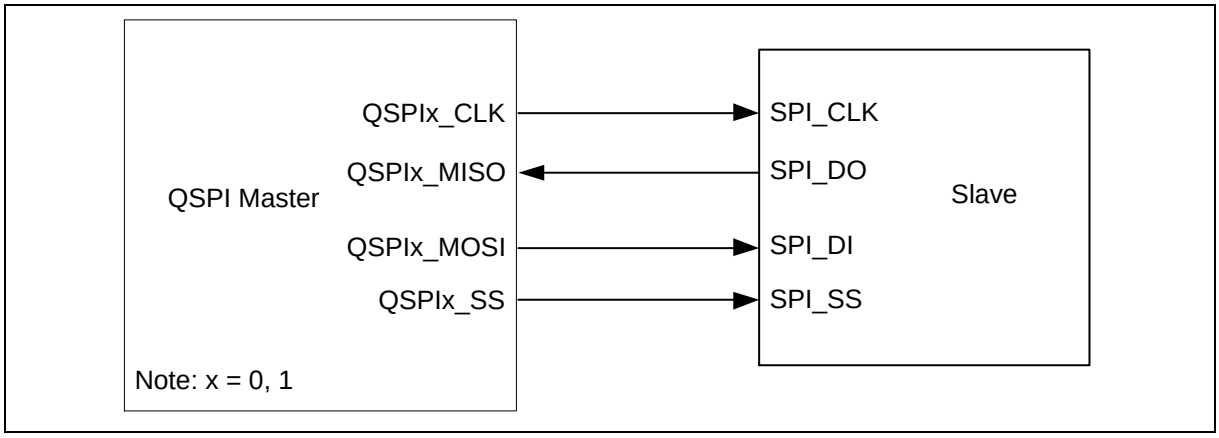


Figure 6.28-3 QSPI Full-Duplex Master Mode Application Block Diagram

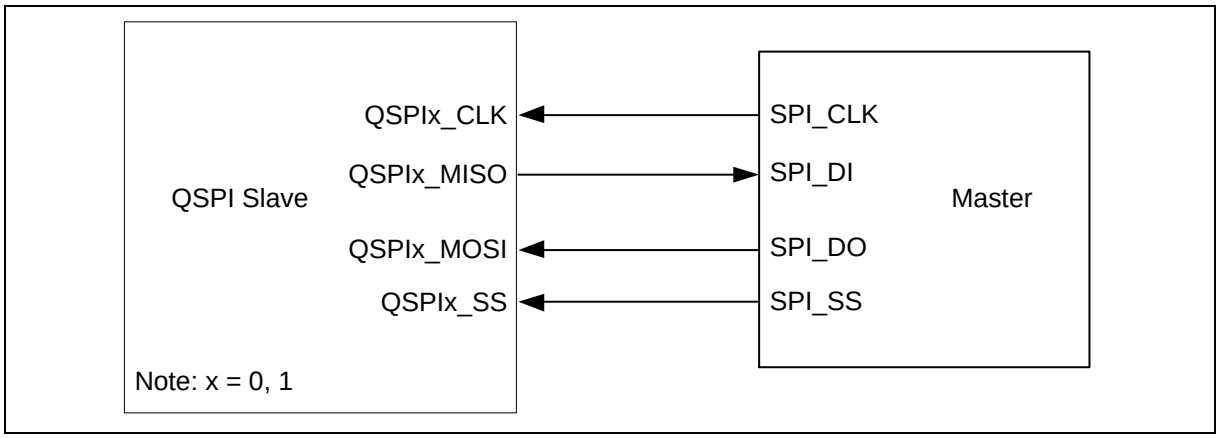


Figure 6.28-4 QSPI Full-Duplex Slave Mode Application Block Diagram

Slave Selection

In Master mode, the QSPI controller can drive off-chip slave device through the slave select output pin QSPIx_SS. In Slave mode, the off-chip master device drives the slave selection signal from the QSPIx_SS input port to this QSPI controller. The duration between the slave select active edge and the first QSPI clock input shall over 3 QSPI peripheral clock cycles of slave.

In Master/Slave mode, the active state of slave selection signal can be programmed to low or high active in SSACTPOL (QSPIx_SSCTL[2]). The selection of slave select conditions depends on what type of device is connected. In Slave mode, to recognize the inactive state of the slave selection signal, the inactive period of the slave selection signal must be larger than or equal to 3 peripheral clock cycles between two successive transactions.

In Slave mode, the QSPIx_MISO signal type is controlled by the QSPIx_SS pin. If the QSPIx_SS pin is inactive state, the QSPIx_MISO is set to tri-state to avoid interference with other Slave devices. If the QSPIx_SS pin is active state, the QSPIx_MISO is set to output mode for data transfer.

Timing Condition

The CLKPOL (QSPIx_CTL[3]) defines the QSPI clock idle state. If CLKPOL = 1, the output QSPI clock is idle at high state; if CLKPOL = 0, it is idle at low state.

TXNEG (QSPIx_CTL[2]) defines the data transmitted out either on negative edge or on positive edge of QSPI clock. RXNEG (QSPIx_CTL[1]) defines the data received either on negative edge or on positive edge of QSPI clock.

Note: The settings of TXNEG and RXNEG are mutually exclusive. In other words, do not transmit and

receive data at the same clock edge.

Transmit/Receive Bit Length

The bit length of a transaction word is defined in DWIDTH (QSPIx_CTL[12:8]) and can be configured up to 32-bit length in a transaction word for transmitting and receiving.

When QSPI controller finishes a transaction, i.e. receives or transmits a specific count of bits defined in DWIDTH (QSPIx_CTL[12:8]), the unit transfer interrupt flag UNITIF (QSPIx_STATUS[1]) will be set to 1.

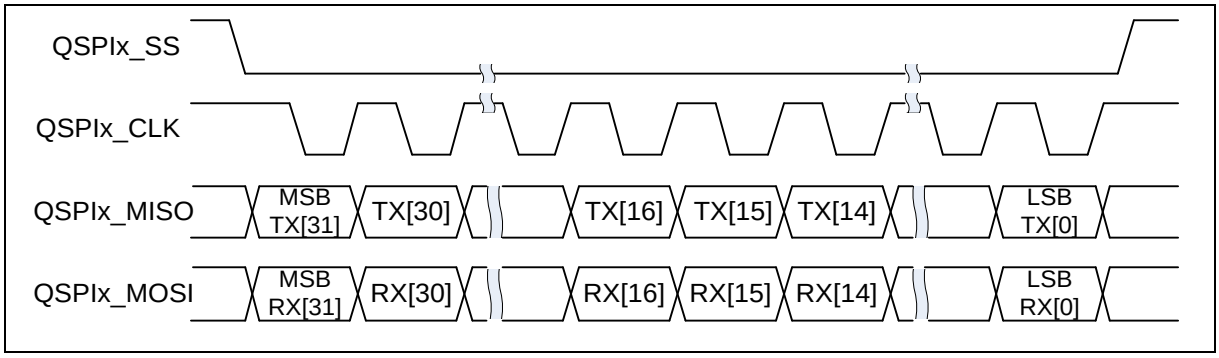


Figure 6.28-532-bit in One Transaction

LSB/MSB First

LSB (QSPIx_CTL[13]) defines the bit transfer sequence in a transaction. If the LSB (QSPIx_CTL[13]) is set to 1, the transfer sequence is LSB first. The bit 0 will be transferred firstly. If the LSB (QSPIx_CTL[13]) is cleared to 0, the transfer sequence is MSB first.

Suspend Interval

SUSPITV (QSPIx_CTL[7:4]) provides a configurable suspend interval, 0.5 ~ 15.5 QSPI clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SUSPITV is 0x3 (3.5 QSPI clock cycles).

6.28.5.2 Automatic Slave Selection

In Master mode, if AUTOSS (QSPIx_SSCTL[3]) is set, the slave selection signal will be generated automatically and output to the QSPIx_SS pin according to whether SS (QSPIx_SSCTL[0]) is enabled or not. The slave selection signal will be set to active state by the QSPI controller when the QSPI data transfer is started by writing to FIFO. It will be set to inactive state when QSPI bus is idle. If QSPI bus is not idle, i.e. TX FIFO, TX shift register or TX skew buffer is not empty, the slave selection signal will be set to inactive state between transactions if the value of SUSPITV (QSPIx_CTL[7:4]) is greater than or equal to 3.

In Master mode, if the value of SUSPITV is less than 3 and the AUTOSS is set as 1, the slave selection signal will be kept at active state between two successive transactions.

If the AUTOSS bit is cleared, the slave selection output signal will be determined by the SS setting. The active state of the slave selection output signal is specified in SSACTPOL (QSPIx_SSCTL[2]).

The duration between the slave selection signal active edge and the first QSPI bus clock edge is 1 QSPI bus clock cycle and the duration between the last QSPI bus clock and the slave selection signal inactive edge is 1.5 QSPI bus clock cycle.

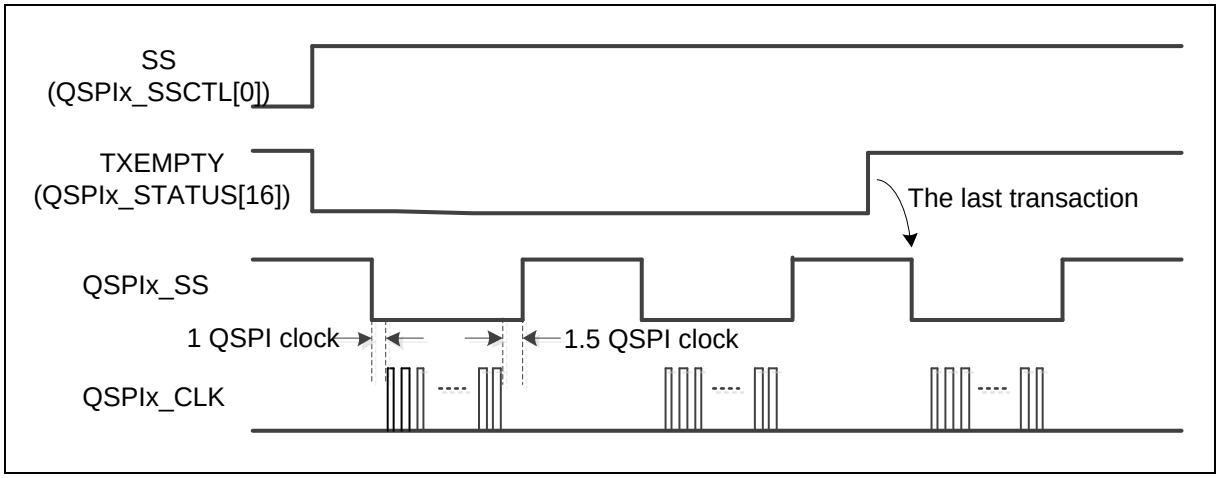


Figure 6.28-6 Automatic Slave Selection (SSACTPOL = 0, SUSPITV > 0x2)

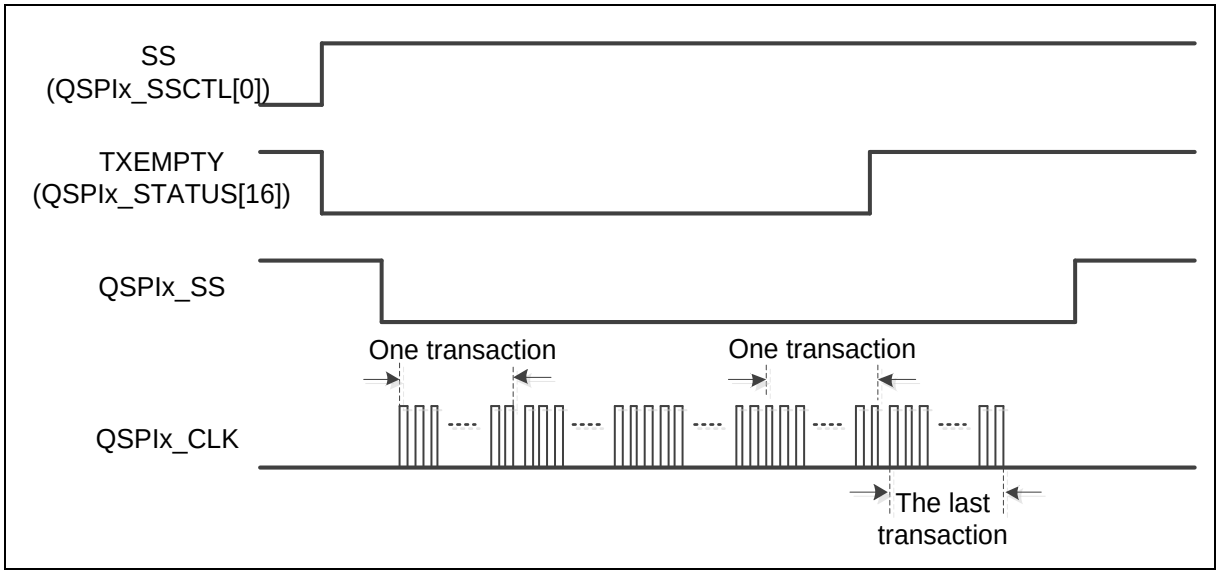


Figure 6.28-7 Automatic Slave Selection (SSACTPOL = 0, SUSPITV < 0x3)

6.28.5.3 Byte Reorder and Suspend Function

When the transfer is set as MSB first (LSB = 0) and the REORDER (QSPIx_CTL[19]) is set to 1, the data stored in the TX buffer and RX buffer will be rearranged in the order as [Byte0, Byte1, Byte2, Byte3] in 32-bit transfer (DWIDTH = 0). The sequence of transmitted/received data will be Byte0, Byte1, Byte2, and then Byte3. If the DWIDTH is set as 24-bit transfer mode, the data in TX buffer and RX buffer will be rearranged as [unknown byte, Byte0, Byte1, Byte2]. The QSPI controller will transmit/receive data with the sequence of Byte0, Byte1 and then Byte2. Each byte will be transmitted/received with MSB first. The rule of 16-bit mode is the same as above. Byte Reorder function is only available when DWIDTH is configured as 16, 24, and 32 bits.

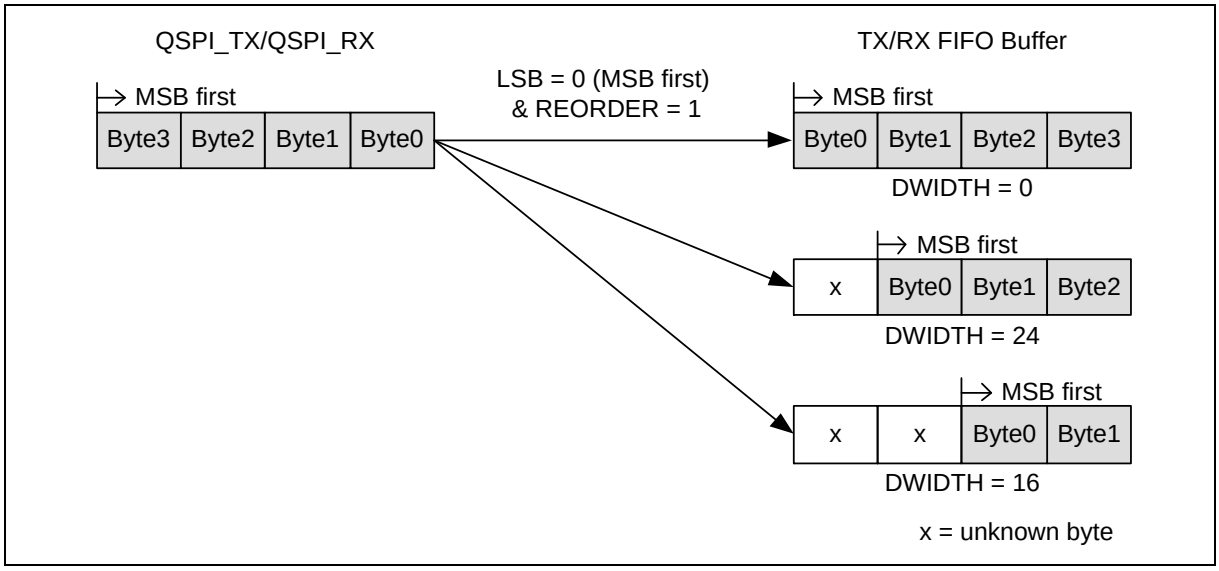


Figure 6.28-8 Byte Reorder Function

In Master mode, if REORDER (QSPIx_CTL[19]) is set to 1, a suspend interval of 0.5 ~ 15.5 QSPI clock periods will be inserted by hardware between two successive bytes in a transaction word. The suspend interval is configured in SUSPITV (QSPIx_CTL[7:4]).

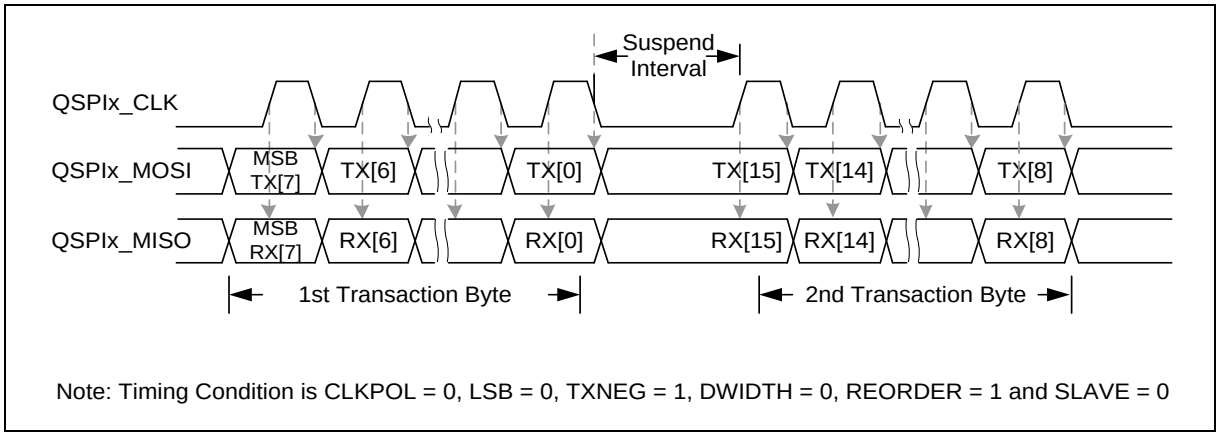


Figure 6.28-9 Timing Waveform for Byte Suspend

6.28.5.4 Half-Duplex Communication

The QSPI controller can communicate in half-duplex mode by setting HALFDPX (QSPIx_CTL[14]) bit. In half-duplex mode, there is only one data line for receiving or transmitting data direction which is defined by DATDIR (QSPIx_CTL[20]). In half-duplex configuration, the QSPIx_MISO pin is free for other applications and it can be configured as GPIO. Enabling or disabling the control bit HALFDPX (QSPIx_CTL[14]) will produce TXFBCLR (QSPIx_FIFOCTL[9]) and RXFBCLR (QSPIx_FIFOCTL[8]) at the same time automatically.

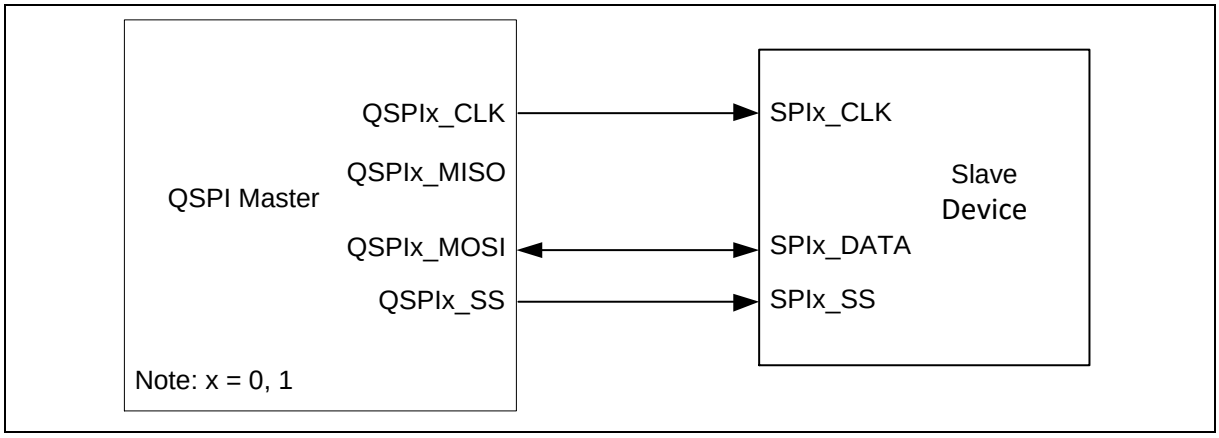


Figure 6.28-10 QSPI Half-Duplex Master Mode Application Block Diagram

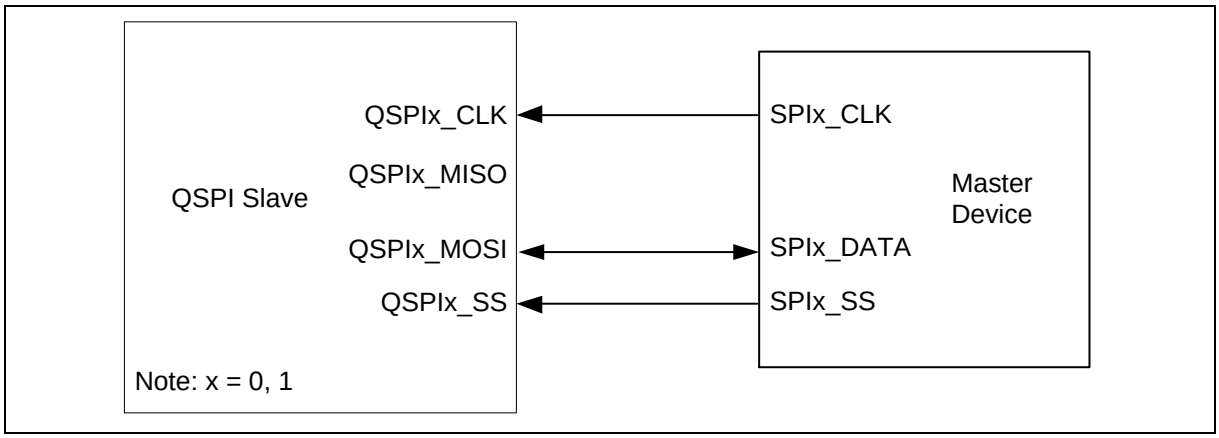


Figure 6.28-11 QSPI Half-Duplex Slave Mode Application Block Diagram

6.28.5.5 Receive-Only Mode

In QSPI Master device, it can communicate in receive-only mode by setting RXONLY (QSPIx_CTL[15]). In this configuration, the QSPI Master device will generate QSPI bus clock continuously as long as the receive-only mode is enabled for receiving data bit from SPI slave device. If AUTOSS (QSPIx_SSCTL[3]) is enabled in receive-only mode, QSPI Master will keep activating the slave select signal.

The remaining QSPIx_MOSI pin of QSPI Master device is not used for communication and can be configured as GPIO. The status BUSY (QSPIx_STATUS[0]) will be asserted in receive-only mode due to the generation of QSPI bus clock. Entering this mode will produce the TXFBCLR (QSPIx_FIFOCTL[9]) and RXFBCLR (QSPIx_FIFOCTL[8]) at the same time automatically. When user enables this mode, the output QSPI bus clock will be sent out after 6 peripheral clock cycles. In this mode, the data which has been written into transmit FIFO will be loaded into transmit shift register and sent out.

When user sets RXONLY (QSPIx_CTL[15]) to enable, QSPI RX data with data bit width of DWIDTH (QSPIx_CTL[12:8]) will be received into RX FIFO and QSPI clock will be sent to SPI slave device until RX FIFO is full.

For two-bit transfer mode TWOBIT (QSPIx_CTL[16]) is enabled, the QSPI master will send QSPI output clock to SPI slave and receive RX data until RX FIFO is full. After user reads RX data from RX FIFO, QSPI master will send QSPI output clock to SPI slave and receive RX data again when RX FIFO counter RXCNT is less than or equal to 4.

6.28.5.6 Slave 3-Wire Mode

When SLV3WIRE (QSPIx_SSCTL[4]) is set by software to enable the Slave 3-Wire mode, the QSPI controller can work with no slave selection signal in Slave mode. The SLV3WIRE (QSPIx_SSCTL[4]) only takes effect in Slave mode. Only three pins, QSPIx_CLK, QSPIx_MISO, and QSPIx_MOSI, are required to communicate with a SPI master. The QSPIx_SS pin can be configured as a GPIO. When the SLV3WIRE (QSPIx_SSCTL[4]) is set to 1, the QSPI slave will be ready to transmit/receive data after the SPIEN (QSPIx_CTL[0]) is set to 1.

6.28.5.7 PDMA Transfer Function

QSPI controller supports PDMA transfer function.

When TXPDMAEN (QSPIx_PDMACTL[0]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

When RXPDMAEN (QSPIx_PDMACTL[1]) is set to 1, the controller will start the PDMA reception process. QSPI controller will issue request to PDMA controller automatically when there is data in the RX FIFO buffer.

Note: QSPI supports single request PDMA (Read/Write) only, burst request PDMA is not supported.

6.28.5.8 Two-bit Transfer Mode

The QSPI controller also supports 2-bit Transfer mode when setting TWOBIT (QSPIx_CTL[16]) to 1. In 2-bit Transfer mode, the QSPI controller performs full duplex data transfer. In other words, the two serial data bits can be transmitted and received simultaneously.

For example, in Master mode, the even data (TX Data (n)) stored in the TX FIFO of QSPIx will be transmitted through the QSPIx_MOSI0 pin and the odd data (TX Data (n+1)) stored in the QSPIx_TX register will be transmitted through the QSPIx_MOSI1 pin respectively. In the meanwhile, the even data received from QSPIx_MISO0 pin will be written to RX FIFO prior to the odd data received from QSPIx_MISO1 pin.

In Slave mode, the even and odd data stored in the QSPIx_TX register will be transmitted through the QSPIx_MISO0 pin and QSPIx_MISO1 pin respectively. In the meanwhile, the RX FIFO of QSPIx will store the even data received from the QSPIx_MOSI0 pin and the odd data from QSPIx_MOSI1 pin respectively. The data sequence of FIFO buffers is the same as the Master mode.

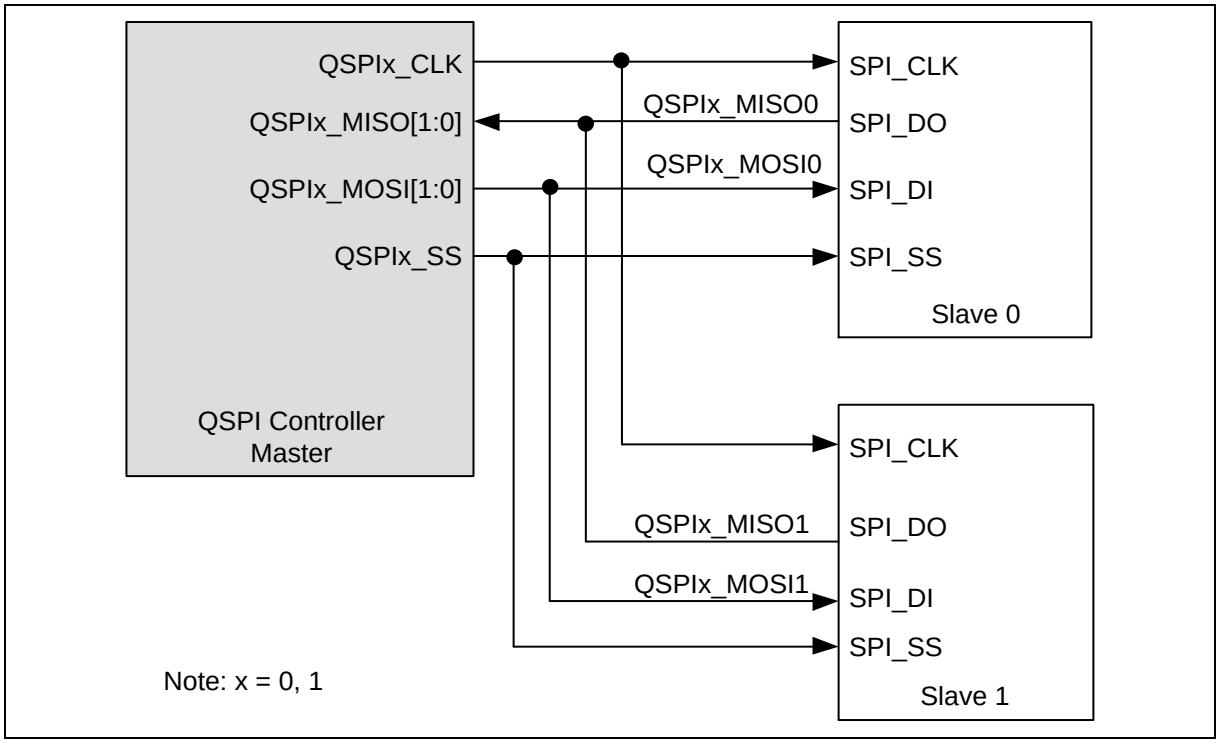


Figure 6.28-12 Two-bit Transfer Mode System Architecture

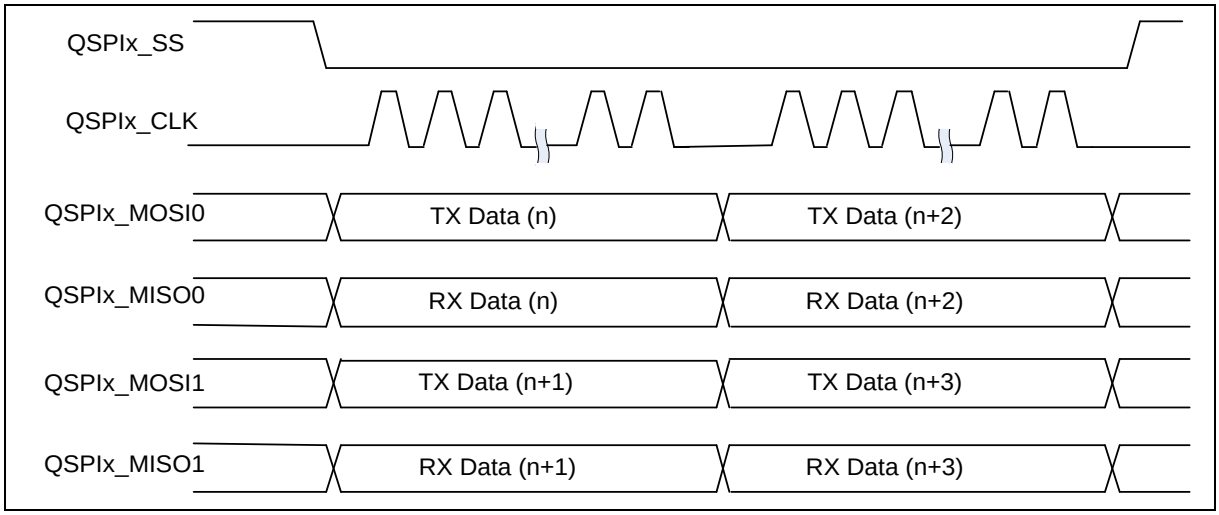


Figure 6.28-13 Two-bit Transfer Mode Timing (Master Mode)

6.28.5.9 Dual I/O Mode

The QSPI controller also supports Dual I/O transfer when setting the DUALIOEN (QSPIx_CTL[21]) to 1. Many general SPI Flashes support Dual I/O transfer. The DATDIR (QSPIx_CTL[20]) is used to define the direction of the transfer data. When the DATDIR bit is set to 1, the controller will send the data to external device. When the DATDIR bit is set to 0, the controller will read the data from the external device. This function supports 8, 16, 24, and 32 bits of length.

The Dual I/O mode is not supported when the Slave 3-Wire mode or the Byte Reorder function is enabled.

For Dual I/O mode, if both the DUALIOEN (QSPIx_CTL[21]) and DATDIR (QSPIx_CTL[20]) are set as

1, the QSPIx_MOSI0 is the even bit data output and the QSPIx_MISO0 will be set as the odd bit data output. If the DUALIOEN (QSPIx_CTL[21]) is set as 1 and DATDIR (QSPIx_CTL[20]) is set as 0, both the QSPIx_MISO0 and QSPIx_MOSI0 will be set as data input ports.

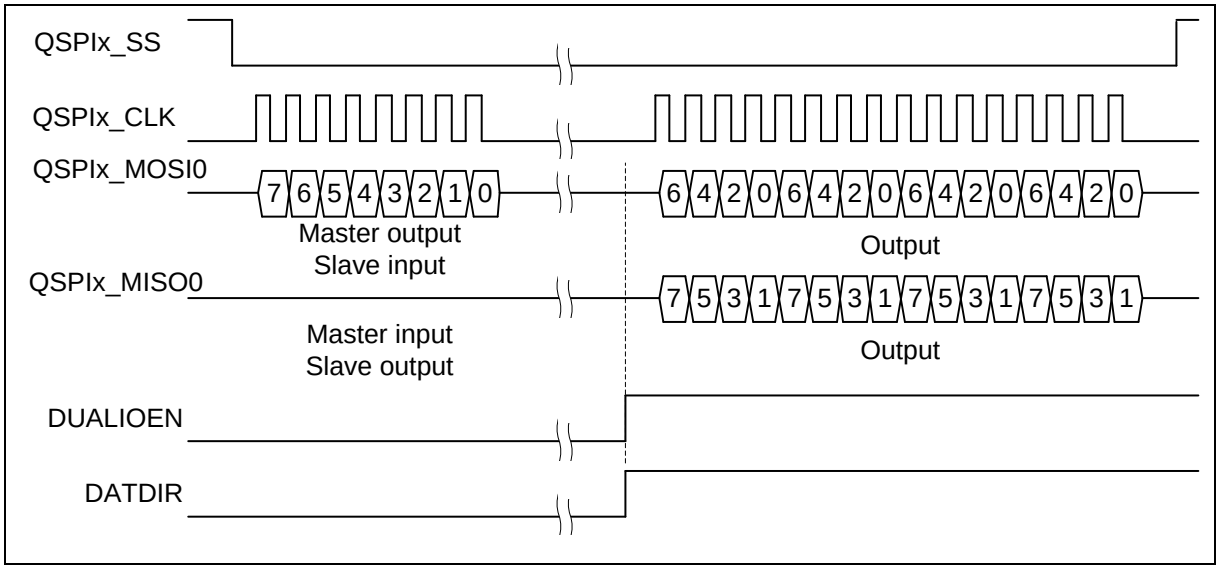


Figure 6.28-14 Bit Sequence of Dual Output Mode

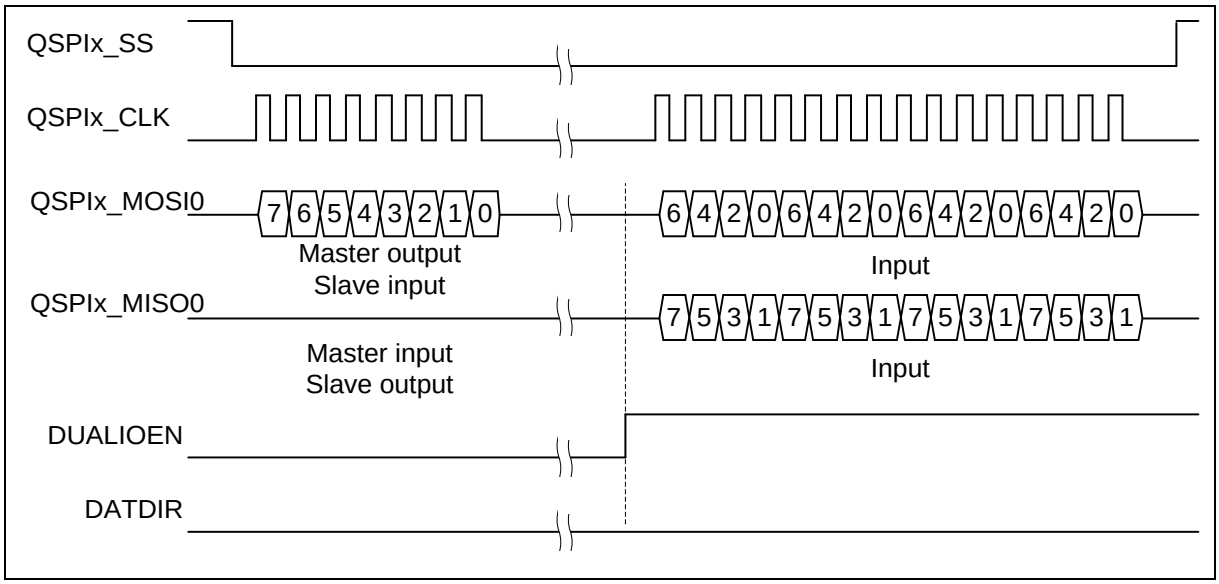


Figure 6.28-15 Bit Sequence of Dual Input Mode

6.28.5.10 Quad I/O Mode

The QSPI controller also supports Quad I/O transfer when setting the QUADIOEN (QSPIx_CTL[22]) to 1. Many general SPI Flashes support Quad I/O transfer. The DATDIR bit (QSPIx_CTL[20]) is used to define the direction of the transfer data. When the DATDIR (QSPIx_CTL[20]) is set to 1, the controller will send the data to external device. When the DATDIR (QSPIx_CTL[20]) is set to 0, the controller will read the data from the external device. This function supports 8, 16, 24, and 32 bits of length.

The Quad I/O mode is not supported when the Slave 3-Wire mode or the Byte Reorder function is enabled. The DUALIOEN (QSPIx_CTL[21]) and QUADIOEN (QSPIx_CTL[22]) shall not be set to 1 simultaneously.

For Quad I/O mode, if both the QUADIOEN (QSPIx_CTL[22]) and DATDIR (QSPIx_CTL[20]) are set as 1, the QSPIx_MOSI0 and QSPIx_MOSI1 are the even bit data output and the QSPIx_MISO0 and QSPIx_MISO1 will be set as the odd bit data output. If the QUADIOEN (QSPIx_CTL[22]) is set as 1 and DATDIR (QSPIx_CTL[20]) is set as 0, all the QSPIx_MISO0, QSPIx_MISO1, QSPIx_MOSI0 and QSPIx_MOSI1 pins will be set as data input ports.

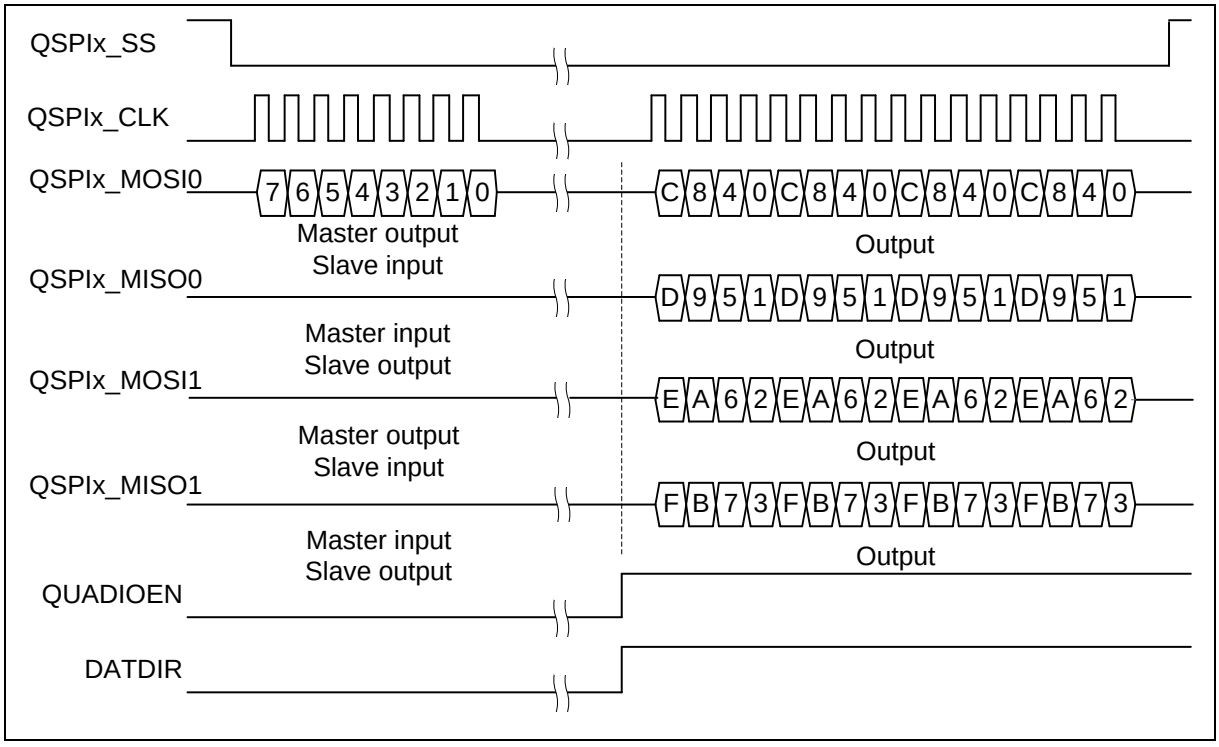


Figure 6.28-16 Bit Sequence of Quad Output Mode

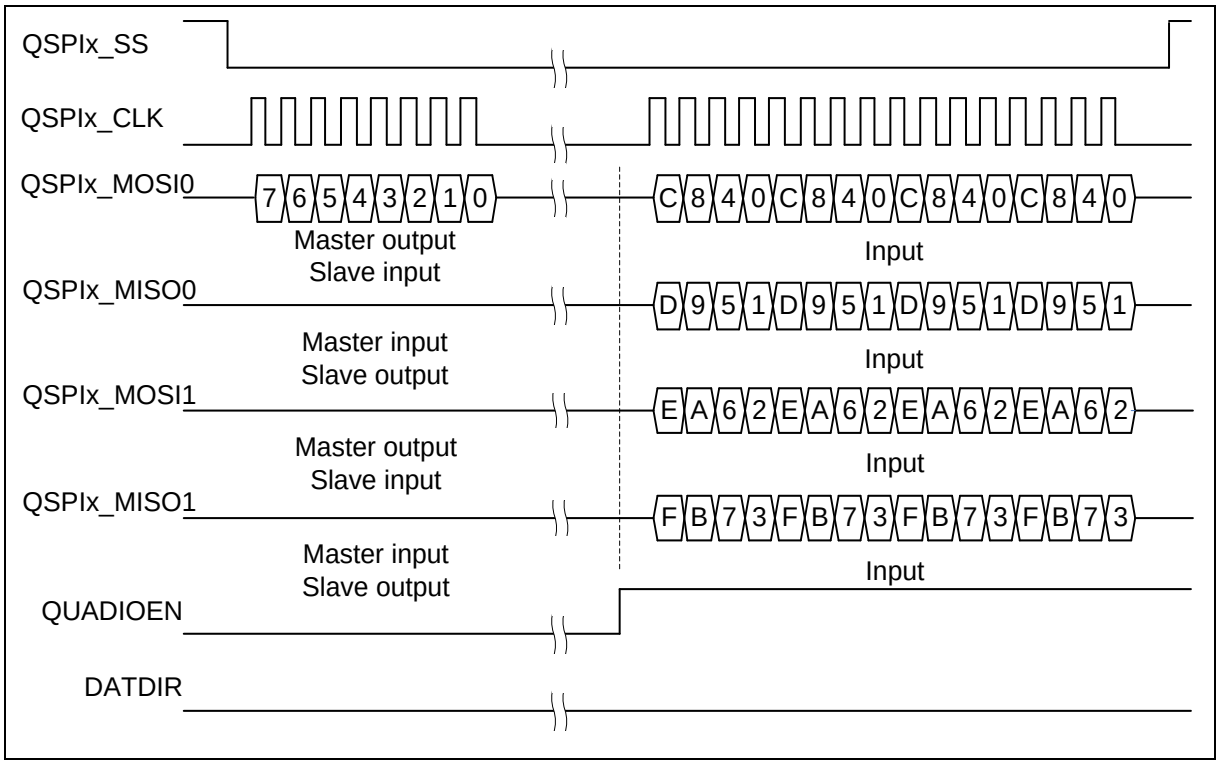


Figure 6.28-17 Bit Sequence of Quad Input Mode

6.28.5.11 Transmit Double Transfer Rate Mode

QSPI Master mode supports TX DTR mode (Transmit Double Transfer Rate Mode) when user sets control register TXDTREN (QSPIx_CTL[23]) to 1. The timing diagram of TX DTR mode is shown in Figure 6.28-18.

In TX DTR mode, the frequency of QSPI bus clock (QSPIx_CLK) equals to the half frequency of QSPI peripheral clock (QSPI_ECLK). TX output data of QSPI in TX DTR mode will be sent near rising edge and falling edge of QSPIx_CLK, where the rising edge and falling edge of QSPIx_CLK are the half period of QSPI peripheral clock (QSPI_ECLK) behind data driving edge of QSPI output data QSPIx_MOSI.

User can select idle polarity of QSPIx_CLK in TX DTR mode using the control register CLKPOL (QSPIx_CTL[3]). When user sets CLKPOL to 0, the idle polarity of QSPIx_CLK is 0. When user sets CLKPOL to 1, the idle polarity of QSPIx_CLK is 1. In TX DTR mode, TXNEG (QSPIx_CTL[2]) equals to CLKPOL (QSPIx_CTL[3]), and SUSPITV (QSPIx_CTL[7:4]) equals to 0. In TX DTR mode of SPI master, user does not use RX data that is received from SPI slave device.

Note: QSPI Master mode supports TXDTR mode, and QSPI Slave mode does not support this mode and there is extra half clock period between the slave select and serial clock output than the TXDTREN (QSPIx_CTL[23]) is disabled.

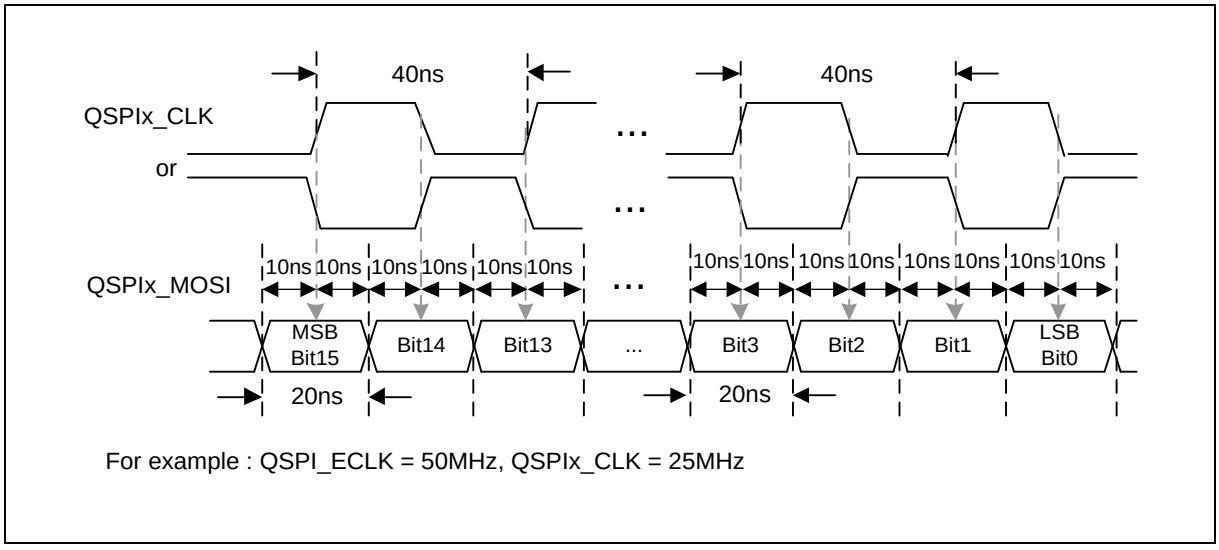


Figure 6.28-18 Timing Diagram of QSPI Output Data for Transmit Double Transfer Rate Mode

6.28.5.12 FIFO Buffer Operation

The QSPI controllers are equipped with eight 32-bit wide transmit and receive FIFO buffers. The data stored in the transmit FIFO buffer will be read and sent out by the transmission control logic. If the transmit FIFO buffer is full, the TXFULL (QSPIx_STATUS[17]) will be set to 1. When the QSPI transmission logic unit draws out the last datum of the transmit FIFO buffer, so that the transmit FIFO buffer is empty, the TXEMPTY (QSPIx_STATUS[16]) will be set to 1. Note that the TXEMPTY (QSPIx_STATUS[16]) flag is set to 1 while the last transaction is still in progress. In Master mode, the BUSY (QSPIx_STATUS[0]) is set to 1 when the FIFO buffer is written any data or there is any transaction on the QSPI bus. (e.g. the slave selection signal is active and the QSPI controller is receiving data in Slave mode). It will set to 0 when the transmit FIFO is empty and the current transaction has done. Thus, the status of BUSY (QSPIx_STATUS[0]) should be checked by software to make sure whether the QSPI is in idle or not.

The receive control logic will store the QSPI input data into the receive FIFO buffer. There are FIFO related status bits, like RXEMPTY (QSPIx_STATUS[8]) and RXFULL (QSPIx_STATUS[9]), to indicate the current status of RX FIFO buffer.

The transmitting and receiving threshold can be configured by setting TXTH (QSPIx_FIFOCTL[30:28]) and RXTH (QSPIx_FIFOCTL[26:24]). When the count of valid data stored in transmit FIFO buffer is less than or equal to TXTH (QSPIx_FIFOCTL[30:28]) setting, TXTHIF (QSPIx_STATUS[18]) will be set to 1. When the count of valid data stored in receive FIFO buffer is larger than RXTH (QSPIx_FIFOCTL[26:24]) setting, RXTHIF (QSPIx_STATUS[10]) will be set to 1.

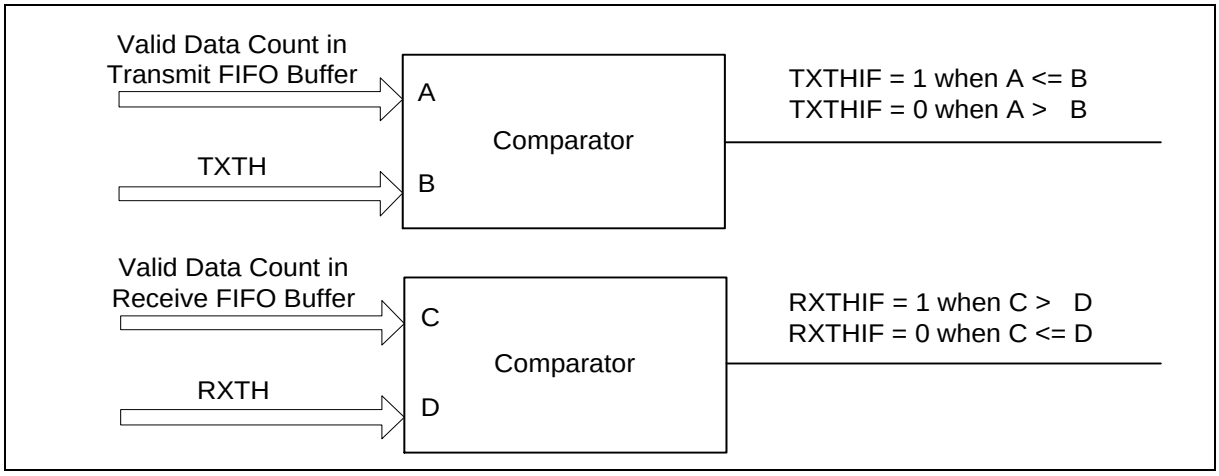


Figure 6.28-19 FIFO Threshold Comparator

In Master mode, when the first datum is written to the QSPIx_TX register, the TXEMPTY flag (QSPIx_STATUS[16]) will be cleared to 0. The transmission will start after 1 PCLK clock cycles and 6 peripheral clock cycles. User can write the next data into QSPIx_TX register immediately. The QSPI controller will insert a suspend interval between two successive transactions. The period of suspend interval is decided by the setting of SUSPITV (QSPIx_CTL[7:4]). If the SUSPITV (QSPIx_CTL[7:4]) equals 0, QSPI controller can perform continuous transfer. User can write data into QSPIx_TX register as long as the TXFULL (QSPIx_STATUS[17]) is 0.

In the example 1 of Figure 6.28-20, it indicates the updated condition of TXEMPTY (QSPIx_STATUS[16]) and the relationship among the FIFO buffer, shift register and the skew buffer. The TXEMPTY (QSPIx_STATUS[16]) is set to 0 when the Data0 is written into the FIFO buffer. The Data0 will be loaded into the shift register by the core logic and the TXEMPTY (QSPIx_STATUS[16]) will be to 1. The Data0 in shift register will be shift into skew buffer by bit for transmission until the transfer is done.

In the Example 2, it indicates the updated condition of TXFULL (QSPIx_STATUS[17]) when there are 8 data in the FIFO buffer and the next data of Data9 does not be written into the FIFO buffer when the TXFULL = 1.

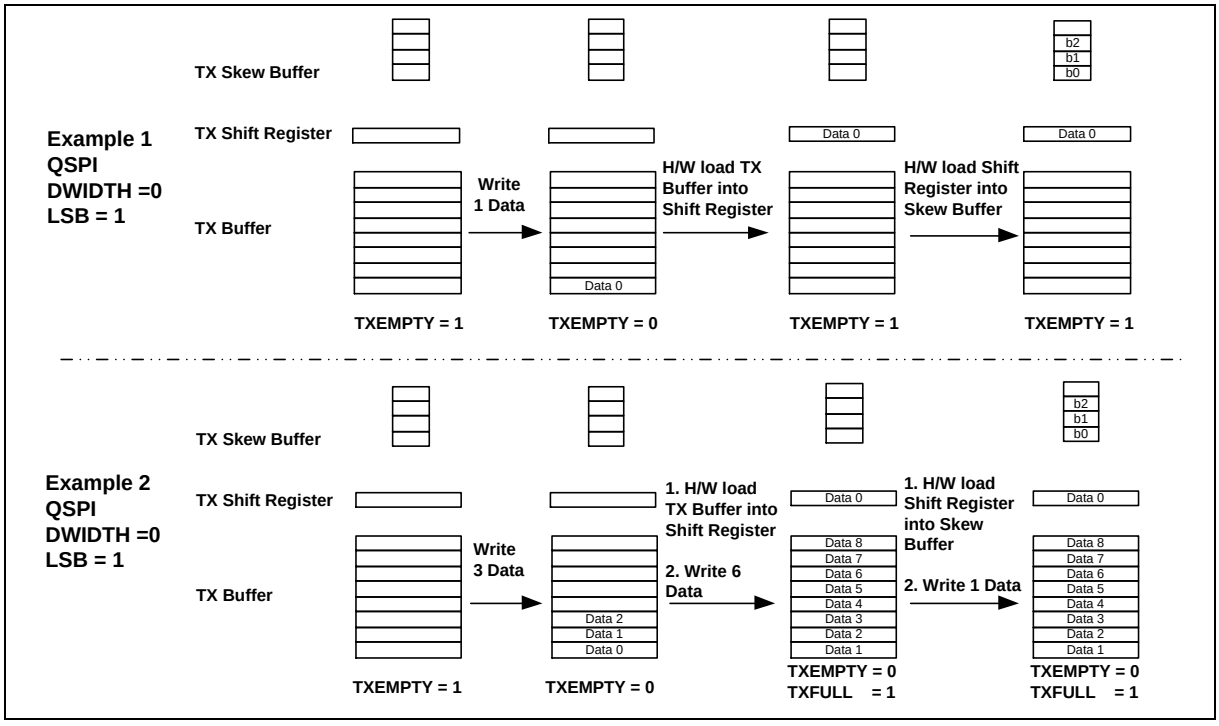


Figure 6.28-20 Transmit FIFO Buffer Example

The subsequent transactions will be triggered automatically if the transmitted data are updated in time. If the QSPIx_TX register is not updated after all data transfer are done, the transfer will stop.

In Master mode, during receiving operation, the serial data are received from QSPIx_MISO pin and stored to receive FIFO buffer.

The received data (Data0's b0, b1, …b31) is stored into skew buffer first according the serial clock (QSPIx_CLK) and then it is shift into the shift register bit by bit. The core logic will load the data in shift register into FIFO buffer when the received data bit count reach the value of DWIDTH (QSPIx_CTL[12:8]). The RXEMPTY (QSPIx_STATUS[8]) will be cleared to 0 while the receive FIFO buffer contains unread data (see the Example 1 of Receive FIFO Buffer Example in Figure 6.28-21). The received data can be read by software from QSPIx_RX register as long as the RXEMPTY (QSPIx_STATUS[8]) is 0. If the receive FIFO buffer contains 8 unread data, the RXFULL (QSPIx_STATUS[9]) will be set to 1 (see the Example 2 of Receive FIFO Buffer Example).

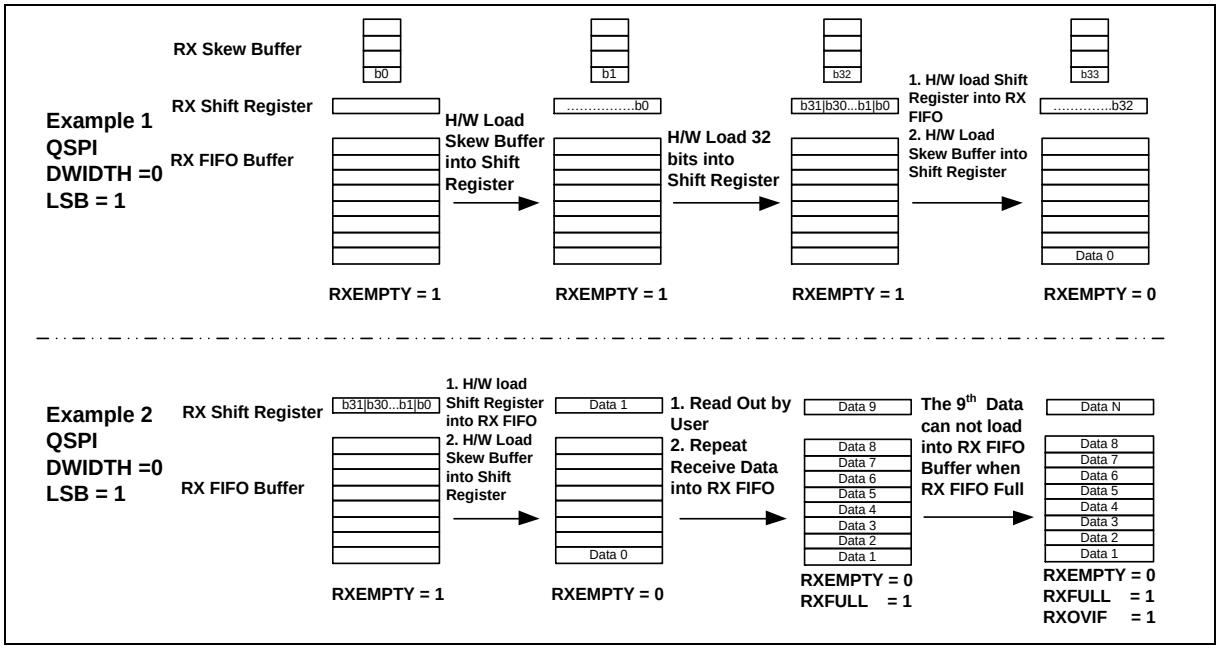


Figure 6.28-21 Receive FIFO Buffer Example

In Slave mode, during transmission operation, when data is written to the QSPIx_TX register by software, the data will be loaded into transmit FIFO buffer and the TXEMPTY (QSPIx_STATUS[16]) will be set to 0. The transmission will start when the slave device receives clock signal from master. Data can be written to QSPIx_TX register as long as the TXFULL (QSPIx_STATUS[17]) is 0. After all data have been drawn out by the QSPI transmission logic unit and the QSPIx_TX register is not updated by software, the TXEMPTY (QSPIx_STATUS[16]) will be set to 1.

If there is no any data written to the QSPIx_TX register, the transmit underflow interrupt flag, TXUFIF (QSPIx_STATUS[19]) will be set to 1 when the slave selection signal is active. The output data will be held by TXUFPOL (QSPIx_FIFOCTL[6]) setting during this transfer until the slave selection signal goes to inactive state. When the transmit underflow event occurs, the slave under run interrupt flag, SLVURIF (QSPIx_STATUS[7]), will be set to 1 as QSPIx_SS goes to inactive state.

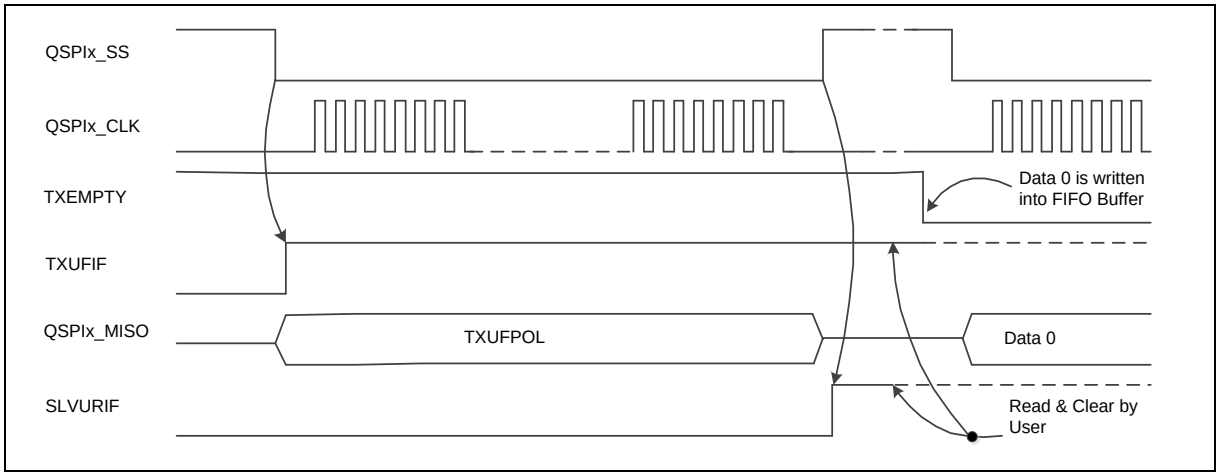


Figure 6.28-22 TX Underflow Event and Slave Under Run Event

In 2-bit Transfer mode, the transmit data is loaded into shift register after 2 datum have been written into the TX FIFO buffer. It uses two shift registers and two 4-level skew buffers concurrently. The detail timing of 2-bit Transfer mode, please refer to the section of Two-bit Transfer mode.

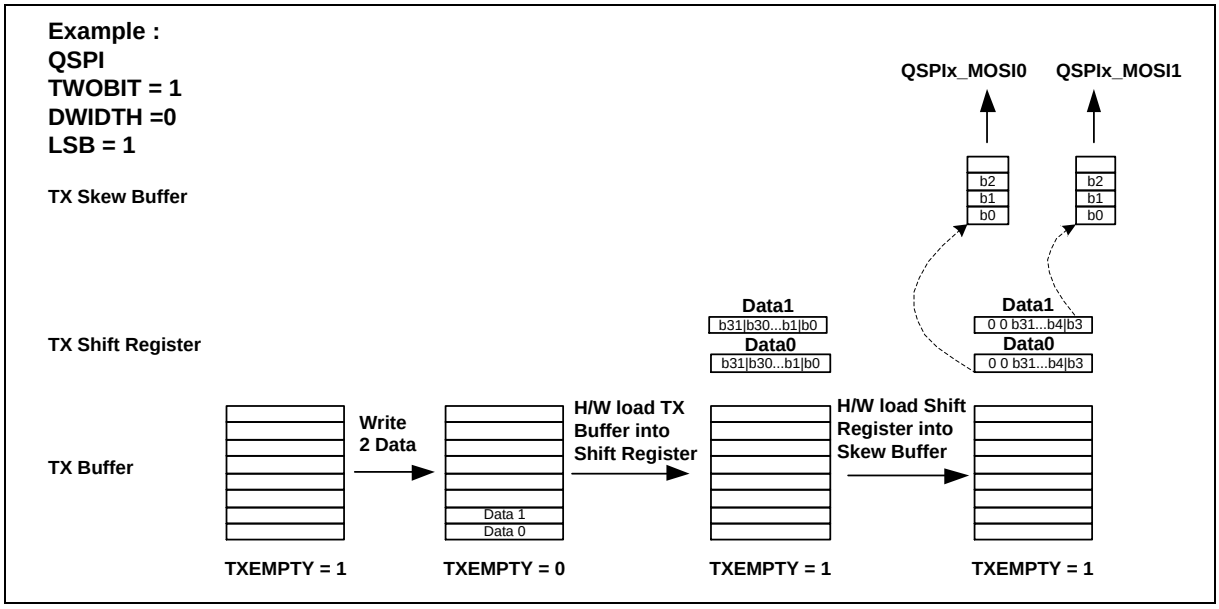


Figure 6.28-23 Two-bit Transfer Mode FIFO Buffer Example

In QSPI Slave 3-Wire mode, the first 2-bit data is un-predicted (keep on the level of last bit in previously transfer) if the data is written into TX FIFO among 3 peripheral clock cycles before the QSPI bus clock is presented. The other bits are held by TXUFPOL (QSPIx_FIFOCTL[6]) because there is TX underflow event. The written data will be transmitted in the next transfer.

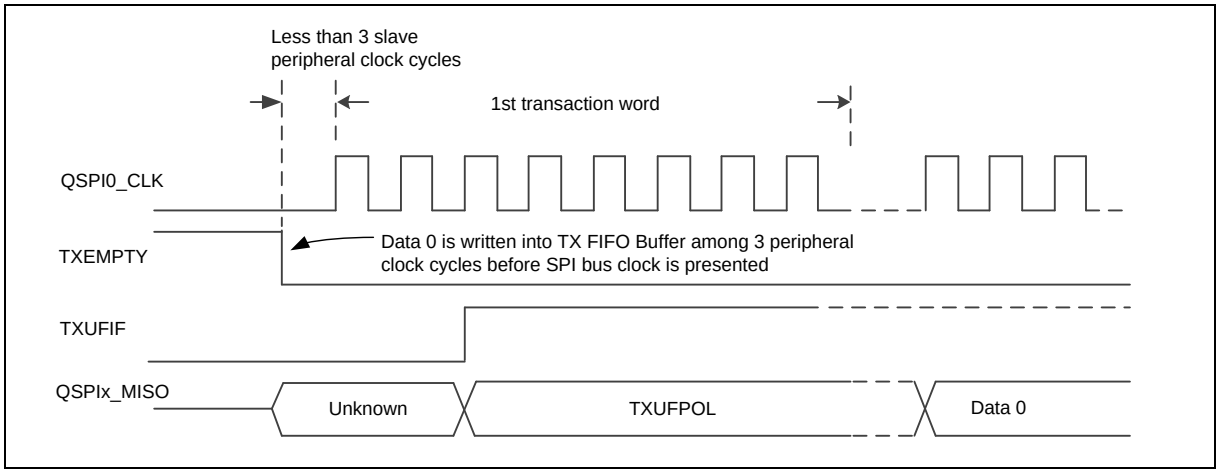


Figure 6.28-24 TX Underflow Event (QSPI Slave 3-Wire Mode Enabled)

In Slave mode, during receiving operation, the serial data is received from QSPIx_MOSI pin and stored to QSPIx_RX register. The reception mechanism is similar to Master mode reception operation. If the receive FIFO buffer contains 8 unread data, the RXFULL (QSPIx_STATUS[9]) will be set to 1 and the RXOVIF (QSPIx_STATUS[11]) will be set to 1 if there is more serial data received from QSPIx_MOSI and follow-up data will be dropped (refer to the Receive FIFO Buffer Example figure).

If the receive bit count mismatch with the DWIDTH (QSPIx_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (QSPIx_STATUS[6]) will be set to 1 in QSPI Slave mode. RX data will be written into RX FIFO and SLVBENUM (QSPIx_STATUS2[29:24]) will be updated when SLVBERX (QSPIx_FIFOCTL[10]) is enabled and QSPI slave bit count error event happened. RX data will be dropped and SLVBENUM (QSPIx_STATUS2[29:24]) will be fixed to 0x0 when the control register SLVBERX (QSPIx_FIFOCTL[10]) is disabled and QSPI slave bit count error event happened. The status register SLVBENUM (QSPIx_STATUS2[29:24]) indicates that effective bit number of uncompleted RX

data when QSPI slave bit count error happened. InFigure 6.28-25, the timing diagram of QSPI slave bit count error and SLVBENUM is shown, and SLVBENUM will be updated to 0x8 when QSPI slave device receives 8 bits data and DWIDTH (QSPIx_CTL[12:8]) is set to 16 bits at QSPI bit count error event. The QSPI slave bit count error event (SLVBEIF) will generates RX FIFO write operation pulse to write RX data from RX shift register to RX FIFO, and RX FIFO count (RXCNT) will update to one.

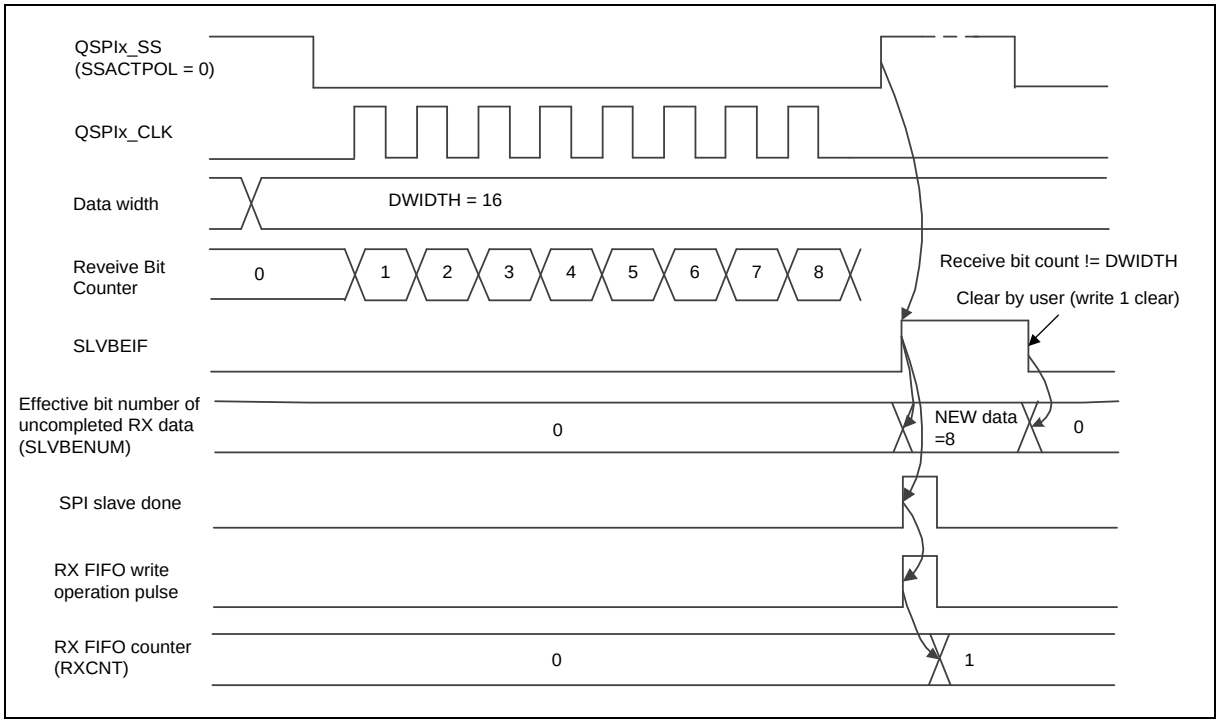


Figure 6.28-25 Slave Mode Bit Count Error and Effective Bit Number of Uncompleted RX Data

When the Slave select signal is active and the value of SLVTOCNT (QSPIx_SSCTL[31:16]) is not 0, the Slave time-out counter in the QSPI controller logic will start after the serial clock input. This counter will be cleared after one transaction done or the SLVTOCNT is set to 0. If the value of the time-out counter is equal to the value of SLVTOCNT before one transaction done, the slave time-out event occurs and the SLVTOIF (QSPIx_STATUS[5]) will be set to 1.

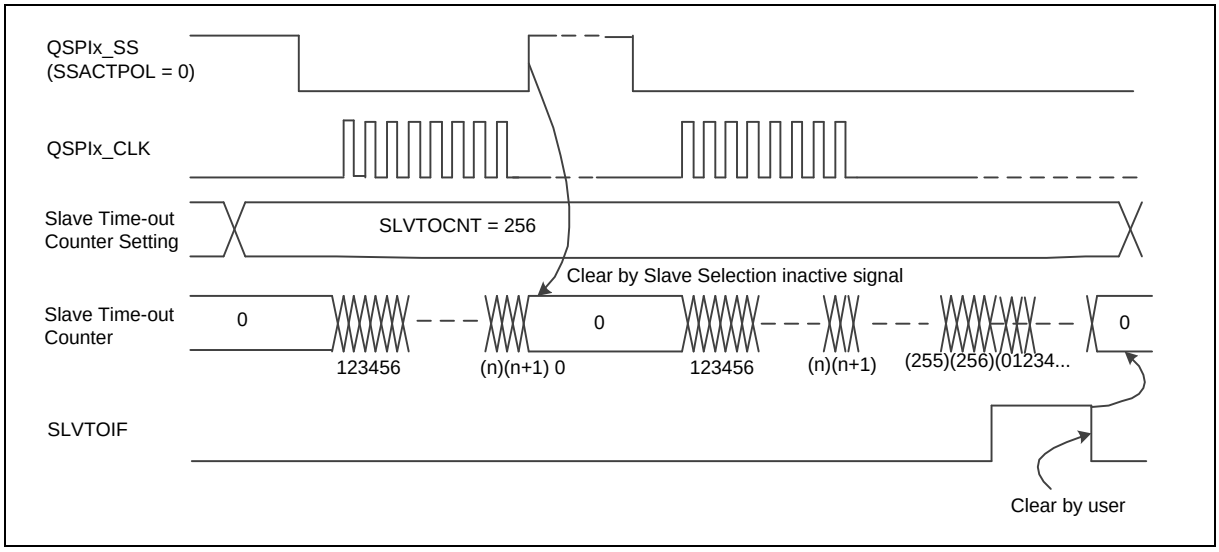


Figure 6.28-26 Slave Time-out Event

A receive time-out function is built-in in this controller. When the receive FIFO is not empty and no read operation in receive FIFO over 64 QSPI peripheral clock periods in Master mode or over 576 QSPI peripheral clock periods in Slave mode, the receive time-out occurs and the RXTOIF (QSPIx_STATUS[12]) will be set to 1. When the receive FIFO is read by user, the time-out status will be cleared automatically.

6.28.5.13 Interrupt

- QSPI unit transfer interrupt

  As the QSPI controller finishes a unit transfer, the unit transfer interrupt flag UNITIF (QSPIx_STATUS[1]) will be set to 1. The unit transfer interrupt event will generate an interrupt to CPU if the unit transfer interrupt enable bit UNITIEN (QSPIx_CTL[17]) is set. The unit transfer interrupt flag can be cleared only by writing 1 to it.

- QSPI slave selection active/inactive interrupt

  In Slave mode, the slave selection active/inactive interrupt flag, SSACTIF (QSPIx_STATUS[2]) and SSINAIF (QSPIx_STATUS[3]), will be set to 1 when the SPIEN (QSPIx_CTL[0]) and SLAVE (QSPIx_CTL[18]) are set to 1 and the slave selection signal goes to active/inactive state. The QSPI controller will issue an interrupt if the SSINAIEN (QSPIx_SSCTL[13]) or SSACTIEN (QSPIx_SSCTL[12]), is set to 1.

- Slave time-out interrupt

  In Slave mode, there is slave time-out function for user to know that there is serial clock input but one transaction is not finished over the period of SLVTOCNT (QSPIx_SSCTL[31:16]) basing on Slave peripheral clock.

  When the slave selection signal is active and the value of SLVTOCNT (QSPIx_SSCTL[31:16]) is not 0, the slave time-out counter in the QSPI controller logic will start after the serial clock input. This counter will be cleared after one transaction done or the SLVTOCNT (QSPIx_SSCTL[31:16]) is set to 0. If the value of the time-out counter is greater than or equal to the value of SLVTOCNT (QSPIx_SSCTL[31:16]) before one transaction done, the slave time-out event occurs and the SLVTOIF (QSPIx_STATUS[5]) will be set to 1. The QSPI controller will issue an interrupt if the SLVTOIEN (QSPIx_SSCTL[5]) is set to 1.

- Slave bit count error interrupt

  In Slave mode, if the transmit/receive bit count mismatch with the DWIDTH (QSPIx_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (QSPIx_STATUS[6]) will be set to 1. The uncompleted transaction will be dropped from TX shift register. When the control register SLVBERX (QSPIx_FIFOCTL[10]) is disabled and QSPI slave bit count error event happened, the uncompleted transaction data will be dropped from RX shift registers. When control register SLVBERX (QSPIx_FIFOCTL[10]) is enabled and QSPI slave bit count error event happened, the uncompleted transaction data will be written from RX shift registers into RX FIFO. The QSPI controller will issue an interrupt if the SLVBEIEN (QSPIx_SSCTL[8]) is set to 1.

  Note: If the slave selection signal is active but there is no any serial clock input, the SLVBEIF (QSPIx_STATUS[6]) will be set to 1 when the slave selection signal goes to inactive state.

- TX underflow interrupt

  In QSPI Slave mode, if there is no any data is written to the QSPIx_TX register, the TXUFIF (QSPIx_STATUS[19]) will be set to 1 when the slave selection signal is active. The QSPI controller will issue a TX underflow interrupt if the TXUFIEN (QSPIx_FIFOCTL[7]) is set to 1.

  Note: If underflow event occurs in QSPI Slave mode, there are two conditions which make QSPI Slave mode return to idle state and then goes for next transfer: (1) set TXRST to 1

(2) slave select signal is changed to inactive state while SLV3WIRE=0.

- Slave TX under run interrupt

  If the TX underflow event occurs, the SLVURIF (QSPIx_STATUS[7]) will be set to 1 when QSPIx_SS goes to inactive state. The QSPI controller will issue a TX under run interrupt if the SLVURIEN (QSPIx_SSCTL[9]) is set to 1.

  Note: In Slave 3-Wire mode, the slave selection signal is considered active all the time so that user shall poll the TXUFIF (QSPIx_STATUS[19]) to know if there is TX underflow event or not.

- Receive Overrun interrupt

  In Slave mode, if the receive FIFO buffer contains 8 unread data, the RXFULL (QSPIx_STATUS[9]) will be set to 1 and the RXOVIF (QSPIx_STATUS[11]) will be set to 1 if there is more serial data received from QSPI bus and follow-up data will be dropped. The QSPI controller will issue an interrupt if the RXOVIEN (QSPIx_FIFOCTL[5]) is set to 1.

- Receive FIFO time-out interrupt

  If there is a received data in the FIFO buffer and it is not read by software over 64 QSPI peripheral clock periods in Master mode or over 576 QSPI peripheral clock periods in Slave mode, it will send a RX time-out interrupt to the system if the RX time-out interrupt enable bit, RXTOIEN (QSPIx_FIFOCTL[4]), is set to 1.

- Transmit FIFO interrupt

  In FIFO mode, if the valid data count of the transmit FIFO buffer is less than or equal to the setting value of TXTH (QSPIx_FIFOCTL[30:28]), the transmit FIFO interrupt flag TXTHIF (QSPIx_STATUS[18]) will be set to 1. The QSPI controller will generate a transmit FIFO interrupt to the system if the transmit FIFO interrupt enable bit, TXTHIEN (QSPIx_FIFOCTL[3]), is set to 1.

- Receive FIFO interrupt

  In FIFO mode, if the valid data count of the receive FIFO buffer is larger than the setting value of RXTH (QSPIx_FIFOCTL[26:24]), the receive FIFO interrupt flag RXTHIF (QSPIx_STATUS[10]) will be set to 1. The QSPI controller will generate a receive FIFO interrupt to the system if the receive FIFO interrupt enable bit, RXTHIEN (QSPIx_FIFOCTL[2]), is set to 1.

6.28.6 Timing Diagram

The active state of slave selection signal can be defined by setting the SSACTPOL (QSPIx_SSCTL[2]). The QSPI clock which is in idle state can be configured as high or low state by setting the CLKPOL (QSPIx_CTL[3]). It also provides the bit length of a transaction word in DWIDTH (QSPIx_CTL[12:8]), and transmitting/receiving data from MSB or LSB first in LSB (QSPIx_CTL[13]). User can also select which edge of QSPI clock to transmit/receive data in TXNEG/RXNEG (QSPIx_CTL[2:1]). Four QSPI timing diagrams for master/slave operations and the related settings are shown below.

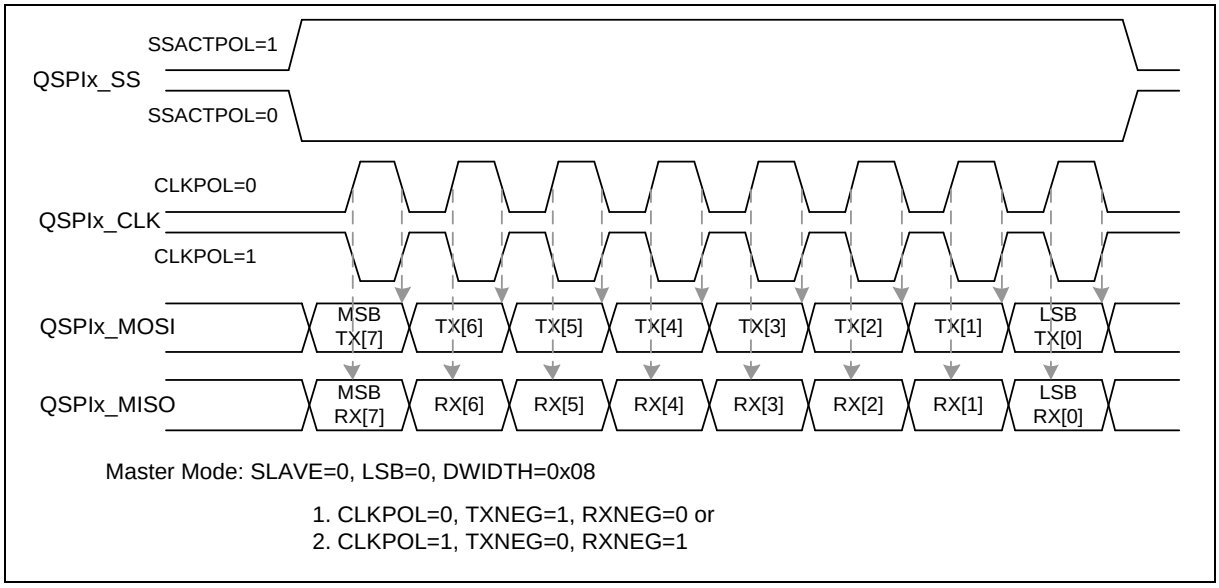


Figure 6.28-27 QSPI Timing in Master Mode

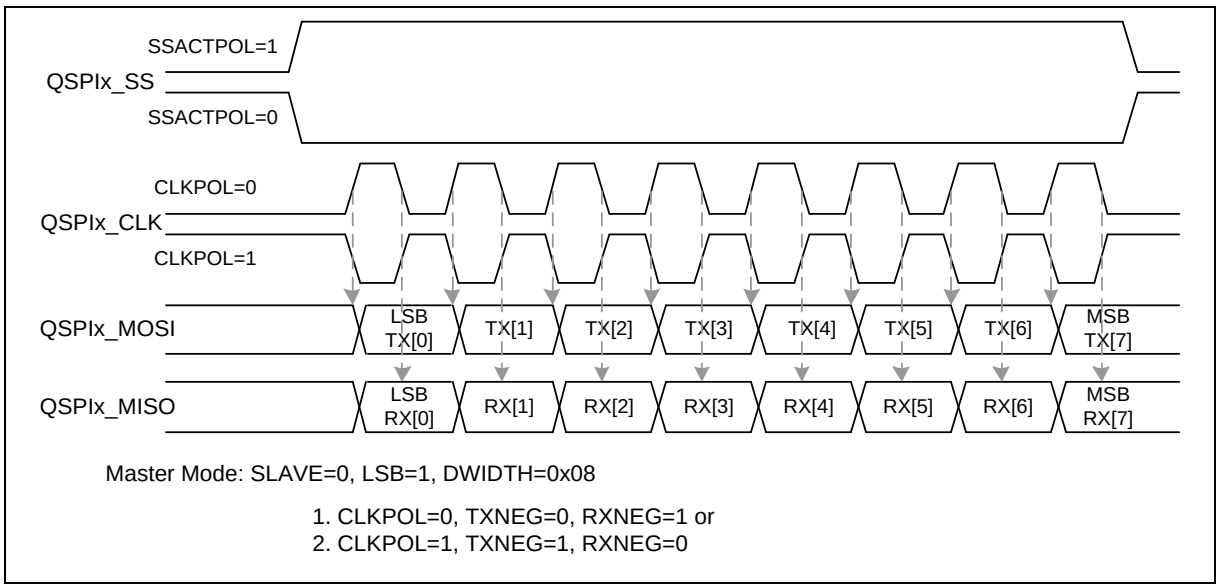


Figure 6.28-28 QSPI Timing in Master Mode (Alternate Phase of QSPIx_CLK)

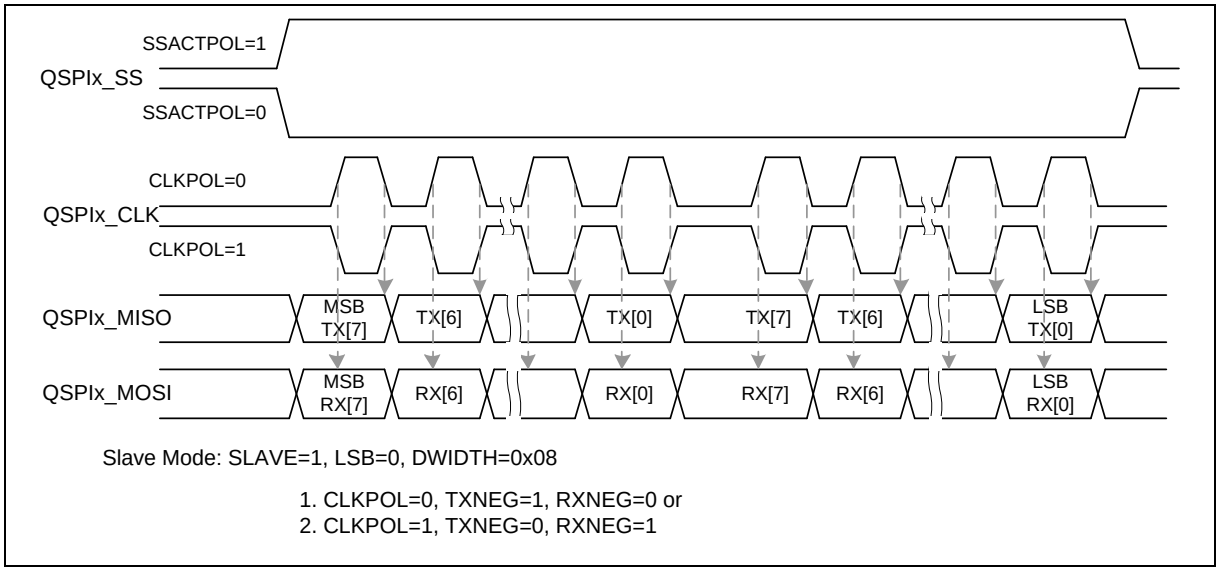


Figure 6.28-29 QSPI Timing in Slave Mode

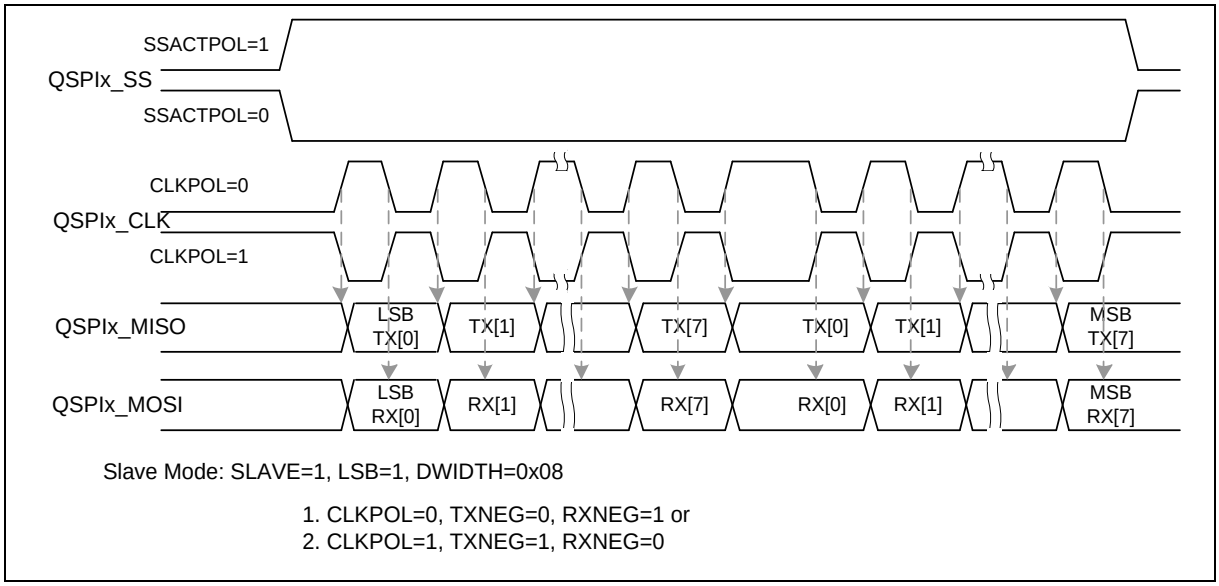


Figure 6.28-30 QSPI Timing in Slave Mode (Alternate Phase of QSPIx_CLK)

6.28.7 Programming Examples

Example 1:

The QSPI controller is set as a full-duplex master to access an off-chip slave device with the following specifications:

- Data bit is latched on positive edge of QSPI bus clock.
- Data bit is driven on negative edge of QSPI bus clock.
- Data is transferred from MSB first.
- QSPI bus clock is idle at low state.
- Only one byte of data to be transmitted/received in a transaction.
- Uses the QSPI slave select pin to connect with an off-chip slave device. The slave

selection signal is active low.

The operation flow is as follows:

1. Set DIVIDER (QSPIx_CLKDIV[8:0]) to determine the output frequency of QSPI clock.
2. Write the QSPIx_SSCTL register a proper value for the related settings of Master mode:
   1) Clear AUTOSS (QSPIx_SSCTL[3]) to 0 to disable the Automatic Slave Selection function.
   2) Configure slave selection signal as active low by clearing SSACTPOL (QSPIx_SSCTL[2]) to 0.
   3) Enable slave selection signal by setting SS (QSPIx_SSCTL[0]) to 1 to activate the off-chip slave device.
3. Write the related settings into the QSPIx_CTL register to control the QSPI master actions.
   1) Configure this QSPI controller as master device by setting SLAVE (QSPIx_CTL[18]) to 0.
   2) Force the QSPI clock idle state at low by clearing CLKPOL (QSPIx_CTL[3]) to 0.
   3) Select data transmitted on negative edge of QSPI bus clock by setting TXNEG (QSPIx_CTL[2]) to 1.
   4) Select data latched on positive edge of QSPI bus clock by clearing RXNEG (QSPIx_CTL[1]) to 0.
   5) Set the bit length of a transaction as 8-bit in DWIDTH bit field (QSPIx_CTL[12:8] = 0x08).
   6) Set MSB transfer first by clearing LSB (QSPIx_CTL[13]) to 0.
4. Set SPIEN (QSPIx_CTL[0]) to 1 to enable the data transfer with the QSPI interface.
5. If this QSPI master attempts to transmit (write) one byte data to the off-chip slave device, write the byte data that will be transmitted into the QSPIx_TX register.
6. Wait for QSPI interrupt if the UNITIEN (QSPIx_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (QSPIx_STATUS[1]).
7. Read out the received one byte data from QSPIx_RX register.
8. Go to 5 to continue with another data transfer or set SS (QSPIx_SSCTL[0]) to 0 to inactivate the off-chip slave device.

Example 2:

The QSPI controller is set as a full-duplex slave device and connects with an off-chip master device. The off-chip master device communicates with the on-chip QSPI slave controller through the QSPI interface with the following specifications:

- Data bit is latched on positive edge of QSPI bus clock.
- Data bit is driven on negative edge of QSPI bus clock.
- Data is transferred from LSB first.
- QSPI bus clock is idle at high state.
- Only one byte of data to be transmitted/received in a transaction.
- Slave selection signal is active high.

The operation flow is as follows:

1. Write the QSPIx_SSCTL register a proper value for the related settings of Slave mode.
2. Select high level for the input of slave selection signal by setting SSACTPOL (QSPIx_SSCTL[2]) to 1.
3. Write the related settings into the QSPIx_CTL register to control this QSPI slave actions

1) Set the QSPI controller as slave device by setting SLAVE (QSPIx_CTL[18]) to 1.
2) Select the QSPI clock idle state at high by setting CLKPOL (QSPIx_CTL[3]) to 1.
3) Select data transmitted on negative edge of QSPI bus clock by setting TXNEG (QSPIx_CTL[2]) to 1.
4) Select data latched on positive edge of QSPI bus clock by clearing RXNEG (QSPIx_CTL[1]) to 0.
5) Set the bit length of a transaction as 8-bit in DWIDTH bit field (QSPIx_CTL[12:8] = 0x08).

4. Set LSB transfer first by setting LSB (QSPIx_CTL[13]) to 1.
5. Set the SPIEN (QSPIx_CTL[0]) to 1. Wait for the slave select trigger input and QSPI clock input from the off-chip master device to start the data transfer.
6. If this QSPI slave attempts to transmit (be read) one byte data to the off-chip master device, write the byte data that will be transmitted into the QSPIx_TX register.
7. If this QSPI slave just only attempts to receive (be written) one byte data from the off-chip master device and does not care what data will be transmitted, the QSPIx_TX register does not need to be updated by software.
8. Wait for QSPI interrupt if the UNITIEN (QSPIx_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (QSPIx_STATUS[1]).
9. Read out the received one byte data from QSPIx_RX register.
10. Go to 6 to continue another data transfer or stop data transfer.

6.28.8 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
QSPI Base Address: **QSPIx_BA = 0x4006_0000 + (0x0000_9000 *x)** **x = 0, 1**				
QSPIx_CTL	QSPIx_BA+0x00	R/W	QSPI Control Register	0x0000_0034
QSPIx_CLKDIV	QSPIx_BA+0x04	R/W	QSPI Clock Divider Register	0x0000_0000
QSPIx_SSCTL	QSPIx_BA+0x08	R/W	QSPI Slave Select Control Register	0x0000_0000
QSPIx_PDMACTL	QSPIx_BA+0x0C	R/W	QSPI PDMA Control Register	0x0000_0000
QSPIx_FIFOCTL	QSPIx_BA+0x10	R/W	QSPI FIFO Control Register	0x4400_0000
QSPIx_STATUS	QSPIx_BA+0x14	R/W	QSPI Status Register	0x0005_0110
QSPIx_STATUS2	QSPIx_BA+0x18	R	QSPI Status2 Register	0x0000_0000
QSPIx_TX	QSPIx_BA+0x20	W	QSPI Data Transmit Register	0x0000_0000
QSPIx_RX	QSPIx_BA+0x30	R	QSPI Data Receive Register	0x0000_0000

6.28.9 Register Description

QSPI Control Register (QSPIx_CTL)

Register	Offset	R/W	Description	Reset Value
QSPIx_CTL	QSPIx_BA+0x00	R/W	QSPI Control Register	0x0000_0034

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TXDTREN	**QUADIOEN**	**DUALIOEN**	**DATDIR**	**REORDER**	**SLAVE**	**UNITIEN**	**TWOBIT**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXONLY	**HALFDPX**	**LSB**	**DWIDTH**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SUSPITV				**CLKPOL**	**TXNEG**	**RXNEG**	**SPIEN**

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**TXDTREN**	**Transmit Double Transfer Rate Mode Enable Bit** 0 = TX DTR mode Disabled. 1 = TX DTR mode Enabled. **Note:** QSPI Master mode supports TXDTR mode, and QSPI Slave mode does not support this mode.
[22]	**QUADIOEN**	**Quad I/O Mode Enable Bit** 0 = Quad I/O mode Disabled. 1 = Quad I/O mode Enabled.
[21]	**DUALIOEN**	**Dual I/O Mode Enable Bit** 0 = Dual I/O mode Disabled. 1 = Dual I/O mode Enabled.
[20]	**DATDIR**	**Data Port Direction Control** This bit is used to select the data input/output direction in half-duplex transfer and Dual/Quad transfer 0 = QSPI data is input direction. 1 = QSPI data is output direction.
[19]	**REORDER**	**Byte Reorder Function Enable Bit** 0 = Byte Reorder function Disabled. 1 = Byte Reorder function Enabled. A byte suspend interval will be inserted among each byte. The period of the byte suspend interval depends on the setting of SUSPITV. **Note:** Byte Reorder function is only available if DWIDTH is defined as 16, 24, and 32 bits.
[18]	**SLAVE**	**Slave Mode Control** 0 = Master mode. 1 = Slave mode.

[17]	**UNITIEN**	**Unit Transfer Interrupt Enable Bit** 0 = QSPI unit transfer interrupt Disabled. 1 = QSPI unit transfer interrupt Enabled.
[16]	**TWOBIT**	**2-bit Transfer Mode Enable Bit** 0 = 2-bit Transfer mode Disabled. 1 = 2-bit Transfer mode Enabled. **Note:** When 2-bit Transfer mode is enabled, the first serial transmitted bit data is from the first FIFO buffer data, and the 2nd serial transmitted bit data is from the second FIFO buffer data. As the same as transmitted function, the first received bit data is stored into the first FIFO buffer and the 2nd received bit data is stored into the second FIFO buffer at the same time.
[15]	**RXONLY**	**Receive-only Mode Enable Bit** This bit field is only available in Master mode. In receive-only mode, QSPI Master will generate QSPI bus clock continuously for receiving data bit from SPI slave device and assert the BUSY status. (Master Only) 0 = Receive-only mode Disabled. 1 = Receive-only mode Enabled.
[14]	**HALFDPX**	**QSPI Half-duplex Transfer Enable Bit** This bit is used to select full-duplex or half-duplex for QSPI transfer. The bit field DATDIR (QSPIx_CTL[20]) can be used to set the data direction in half-duplex transfer. 0 = QSPI operates in full-duplex transfer. 1 = QSPI operates in half-duplex transfer.
[13]	**LSB**	**Send LSB First** 0 = The MSB, which bit of transmit/receive register depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, bit 0 of the QSPIx TX register, is sent first to the QSPI data output pin, and the first bit received from the QSPI data input pin will be put in the LSB position of the RX register (bit 0 of QSPIx_RX).
[12:8]	**DWIDTH**	**Data Width** This field specifies how many bits can be transmitted/received in one transaction. The minimum bit length is 8 bits and can up to 32 bits. DWIDTH = 0x08 …. 8 bits. DWIDTH = 0x09 …. 9 bits. …… DWIDTH = 0x1F …. 31 bits. DWIDTH = 0x00 …. 32 bits.
[7:4]	**SUSPITV**	**Suspend Interval** The four bits provide configurable suspend interval between two successive transmit/receive transaction in a transfer. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the suspend interval is obtained according to the following equation (Master only). (SUSPITV[3:0] + 0.5) * period of QSPI_CLK clock cycle Example: SUSPITV = 0x0 …. 0.5 QSPI_CLK clock cycle. SUSPITV = 0x1 …. 1.5 QSPI_CLK clock cycle. …… SUSPITV = 0xE …. 14.5 QSPI_CLK clock cycle. SUSPITV = 0xF …. 15.5 QSPI_CLK clock cycle. **Note:** In TX DTR mode, SUSPITV equals to 0x0.
[3]	**CLKPOL**	**Clock Polarity**

		0 = QSPI bus clock is idle low. 1 = QSPI bus clock is idle high.
[2]	**TXNEG**	**Transmit on Negative Edge** 0 = Transmitted data output signal is changed on the rising edge of QSPI bus clock. 1 = Transmitted data output signal is changed on the falling edge of QSPI bus clock. **Note:** In TX DTR mode, TXNEG equals to CLKPOL (QSPIx_CTL[3]).
[1]	**RXNEG**	**Receive on Negative Edge** 0 = Received data input signal is latched on the rising edge of QSPI bus clock. 1 = Received data input signal is latched on the falling edge of QSPI bus clock.
[0]	**SPIEN**	**QSPI Transfer Control Enable Bit** In Master mode, the transfer will start when there is data in the FIFO buffer after this bit is set to 1. In Slave mode, this device is ready to receive data when this bit is set to 1. 0 = Transfer control Disabled. 1 = Transfer control Enabled. **Note:** Before changing the configurations of QSPIx_CTL, QSPIx_CLKDIV, QSPIx_SSCTL and QSPIx_FIFOCTL registers, user shall clear the SPIEN (QSPIx_CTL[0]) and confirm the SPIENSTS (QSPIx_STATUS[15]) is 0.

QSPI Clock Divider Register (QSPIx_CLKDIV)

Register	Offset	R/W	Description	Reset Value
QSPIx_CLKDIV	QSPIx_BA+0x04	R/W	QSPI Clock Divider Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**DIVIDER**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIVIDER							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**DIVIDER**	**Clock Divider** The value in this field is the frequency divider for generating the peripheral clock, f_{spi_eclk}, and the QSPI bus clock of QSPI Master. The frequency is obtained according to the following equation. $f_{spi_eclk} = \frac{f_{spi_clock_src}}{(DIVIDER+1)}$ where $f_{spi_clock_src}$ is the peripheral clock source, which is defined in the clock control register, CLK_CLKSEL2. **Note:** The time interval must be larger than or equal to 8 peripheral clock cycles between releasing QSPI IP software reset and setting this clock divider register.

Note: DIVIDER should be set carefully because the peripheral clock frequency must be slower than or equal to system frequency.

QSPI Slave Select Control Register (QSPIx_SSCTL)

Register	Offset	R/W	Description	Reset Value
QSPIx_SSCTL	QSPIx_BA+0x08	R/W	QSPI Slave Select Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SLVTOCNT							
23	22	21	20	19	18	17	16
SLVTOCNT							
15	14	13	12	11	10	9	8
Reserved		SSINAIEN	SSACTIEN	Reserved		SLVURIEN	SLVBEIEN
7	6	5	4	3	2	1	0
Reserved	SLVTORST	SLVTOIEN	SLV3WIRE	AUTOSS	SSACTPOL	Reserved	SS

Bits	Description	
[31:16]	**SLVTOCNT**	**Slave Mode Time-out Period** In Slave mode, these bits indicate the time-out period when there is bus clock input during slave select active. The clock source of the time-out counter is Slave peripheral clock. If the value is 0, it indicates the slave mode time-out function is disabled.
[15:14]	**Reserved**	Reserved.
[13]	**SSINAIEN**	**Slave Select Inactive Interrupt Enable Bit** 0 = Slave select inactive interrupt Disabled. 1 = Slave select inactive interrupt Enabled.
[12]	**SSACTIEN**	**Slave Select Active Interrupt Enable Bit** 0 = Slave select active interrupt Disabled. 1 = Slave select active interrupt Enabled.
[11:10]	**Reserved**	Reserved.
[9]	**SLVURIEN**	**Slave Mode TX Under Run Interrupt Enable Bit** 0 = Slave mode TX under run interrupt Disabled. 1 = Slave mode TX under run interrupt Enabled.
[8]	**SLVBEIEN**	**Slave Mode Bit Count Error Interrupt Enable Bit** 0 = Slave mode bit count error interrupt Disabled. 1 = Slave mode bit count error interrupt Enabled.
[7]	**Reserved**	Reserved.
[6]	**SLVTORST**	**Slave Mode Time-out Reset Control** 0 = When Slave mode time-out event occurs, the TX and RX control circuit will not be reset. 1 = When Slave mode time-out event occurs, the TX and RX control circuit will be reset by hardware.
[5]	**SLVTOIEN**	**Slave Mode Time-out Interrupt Enable Bit** 0 = Slave mode time-out interrupt Disabled. 1 = Slave mode time-out interrupt Enabled.

[4]	**SLV3WIRE**	**Slave 3-wire Mode Enable Bit** In Slave 3-wire mode, the QSPI controller can work with 3-wire interface including QSPIx_CLK, QSPIx_MISO and QSPIx_MOSI pins. 0 = 4-wire bi-direction interface. 1 = 3-wire bi-direction interface.
[3]	**AUTOSS**	**Automatic Slave Selection Function Enable BitThis bit is supported in Master mode only** 0 = Automatic slave selection function Disabled. Slave selection signal will be asserted/de-asserted according to SS (QSPIx_SSCTL[0]). 1 = Automatic slave selection function Enabled.
[2]	**SSACTPOL**	**Slave Selection Active Polarity** This bit defines the active polarity of slave selection signal (QSPIx_SS). 0 = The slave selection signal QSPIx_SS is active low. 1 = The slave selection signal QSPIx_SS is active high.
[1]	**Reserved**	Reserved.
[0]	**SS**	**Slave Selection Control** If AUTOSS bit is cleared to 0, 0 = Set the QSPIx_SS line to inactive state. 1 = Set the QSPIx_SS line to active state. If the AUTOSS bit is set to 1, 0 = Keep the QSPIx_SS line at inactive state. 1 = QSPIx_SS line will be automatically driven to active state for the duration of data transfer, and will be driven to inactive state for the rest of the time. The active state of QSPIx_SS is specified in SSACTPOL (QSPIx_SSCTL[2]).

QSPI PDMA Control Register (QSPIx_PDMACTL)

Register	Offset	R/W	Description	Reset Value
QSPIx_PDMACTL	QSPIx_BA+0x0C	R/W	QSPI PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					PDMARST	RXPDMAEN	TXPDMAEN

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the PDMA control logic of the QSPI controller. This bit will be automatically cleared to 0.
[1]	**RXPDMAEN**	**Receive PDMA Enable Bit** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[0]	**TXPDMAEN**	**Transmit PDMA Enable Bit** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled. **Note 1:** In QSPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, RX PDMA function cannot be enabled prior to TX PDMA function. User can enable TX PDMA function firstly or enable both functions simultaneously. **Note 2:** In QSPI Master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, TX PDMA function cannot be disabled prior to RX PDMA function. User can disable RX PDMA function firstly or disable both functions simultaneously.

QSPI FIFO Control Register (QSPIx_FIFOCTL)

Register	Offset	R/W	Description	Reset Value
QSPIx_FIFOCTL	QSPIx_BA+0x10	R/W	QSPI FIFO Control Register	0x4400_0000

31	30	29	28	27	26	25	24
Reserved	TXTH			Reserved	RXTH		
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					SLVBERX	TXFBCLR	RXFBCLR
7	6	5	4	3	2	1	0
TXUFIEN	TXUFPOL	RXOVIEN	RXTOIEN	TXTHIEN	RXTHIEN	TXRST	RXRST

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TXTH**	**Transmit FIFO Threshold** If the valid data count of the transmit FIFO buffer is less than or equal to the TXTH setting, the TXTHIF bit will be set to 1, else the TXTHIF bit will be cleared to 0.
[27]	**Reserved**	Reserved.
[26:24]	**RXTH**	**Receive FIFO Threshold** If the valid data count of the receive FIFO buffer is larger than the RXTH setting, the RXTHIF bit will be set to 1, else the RXTHIF bit will be cleared to 0.
[23:11]	**Reserved**	Reserved.
[10]	**SLVBERX**	**RX FIFO Write Data Enable Bit When Slave Mode Bit Count Error** 0 = Uncompleted RX data will be dropped from RX FIFO when bit count error event happen in QSPI slave mode. 1 = Uncompleted RX data will be written into RX FIFO when bit count error event happen in QSPI slave mode. User can read SLVBENUM (QSPIx_STATUS2[29:24]) to know that the effective bit number of uncompleted RX data when SPI slave bit count error happened.
[9]	**TXFBCLR**	**Transmit FIFO Buffer Clear** 0 = No effect. 1 = Clear transmit FIFO pointer. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1. **Note:** The TX shift register will not be cleared.
[8]	**RXFBCLR**	**Receive FIFO Buffer Clear** 0 = No effect. 1 = Clear receive FIFO pointer. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1. **Note:** The RX shift register will not be cleared.
[7]	**TXUFIEN**	**TX Underflow Interrupt Enable Bit**

		When TX underflow event occurs in Slave mode, TXUFIF (QSPIx_STATUS[19]) will be set to 1. This bit is used to enable the TX underflow interrupt. 0 = Slave TX underflow interrupt Disabled. 1 = Slave TX underflow interrupt Enabled.
[6]	**TXUFPOL**	**TX Underflow Data Polarity** 0 = The QSPI data out is kept 0 if there is TX underflow event in Slave mode. 1 = The QSPI data out is kept 1 if there is TX underflow event in Slave mode. **Note 1:** The TX underflow event occurs if there is no any data in TX FIFO when the slave selection signal is active. **Note 2:** When TX underflow event occurs, QSPIx_MISO pin state will be determined by this setting even though TX FIFO is not empty afterward. Data stored in TX FIFO will be sent through QSPIx_MISO pin in the next transfer frame.
[5]	**RXOVIEN**	**Receive FIFO Overrun Interrupt Enable Bit** 0 = Receive FIFO overrun interrupt Disabled. 1 = Receive FIFO overrun interrupt Enabled.
[4]	**RXTOIEN**	**Receive Time-out Interrupt Enable Bit** 0 = Receive time-out interrupt Disabled. 1 = Receive time-out interrupt Enabled.
[3]	**TXTHIEN**	**Transmit FIFO Threshold Interrupt Enable Bit** 0 = TX FIFO threshold interrupt Disabled. 1 = TX FIFO threshold interrupt Enabled.
[2]	**RXTHIEN**	**Receive FIFO Threshold Interrupt Enable Bit** 0 = RX FIFO threshold interrupt Disabled. 1 = RX FIFO threshold interrupt Enabled.
[1]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset transmit FIFO pointer and transmit circuit. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (QSPIx_STATUS[23]) to check if reset is accomplished or not. **Note:** If TX underflow event occurs in QSPI Slave mode, this bit can be used to make QSPI return to idle state.
[0]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset receive FIFO pointer and receive circuit. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (QSPIx_STATUS[23]) to check if reset is accomplished or not.

QSPI Status Register (QSPIx_STATUS)

Register	Offset	R/W	Description	Reset Value
QSPIx_STATUS	QSPIx_BA+0x14	R/W	QSPI Status Register	0x0005_0110

31	30	29	28	27	26	25	24
TXCNT				RXCNT			
23	22	21	20	19	18	17	16
TXRXRST	Reserved			TXUFIF	TXTHIF	TXFULL	TXEMPTY
15	14	13	12	11	10	9	8
SPIENSTS	Reserved		RXTOIF	RXOVIF	RXTHIF	RXFULL	RXEMPTY
7	6	5	4	3	2	1	0
SLVURIF	SLVBEIF	SLVTOIF	SSLINE	SSINAIF	SSACTIF	UNITIF	BUSY

Bits	Description	
[31:28]	**TXCNT**	**Transmit FIFO Data Count (Read only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27:24]	**RXCNT**	**Receive FIFO Data Count (Read only)** This bit field indicates the valid data count of receive FIFO buffer.
[23]	**TXRXRST**	**TX or RX Reset Status (Read only)** 0 = The reset function of TXRST or RXRST is done. 1 = Doing the reset function of TXRST or RXRST. **Note:** Both the reset operations of TXRST and RXRST need 3 system clock cycles + 2 peripheral clock cycles. User can check the status of this bit to monitor the reset function is doing or done.
[22:20]	**Reserved**	Reserved.
[19]	**TXUFIF**	**TX Underflow Interrupt Flag** When the TX underflow event occurs, this bit will be set to 1, the state of data output pin depends on the setting of TXUFPOL. 0 = No effect. 1 = No data in Transmit FIFO and TX shift register when the slave selection signal is active. **Note 1:** This bit will be cleared by writing 1 to it. **Note 2:** If reset slave's transmission circuit when slave selection signal is active, this flag will be set to 1 after 2 peripheral clock cycles + 3 system clock cycles since the reset operation is done.
[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TXTH.
[17]	**TXFULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.

[16]	**TXEMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** 0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[15]	**SPIENSTS**	**QSPI Enable Status (Read Only)** 0 = QSPI controller Disabled. 1 = QSPI controller Enabled. **Note:** The QSPI peripheral clock is asynchronous with the system clock. In order to make sure the QSPI control logic is disabled, this bit indicates the real status of QSPI controller.
[14:13]	**Reserved**	Reserved.
[12]	**RXTOIF**	**Receive Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 QSPI peripheral clock periods in Master mode or over 576 QSPI peripheral clock periods in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to it.
[11]	**RXOVIF**	**Receive FIFO Overrun Interrupt Flag** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. 0 = No FIFO is overrun. 1 = Receive FIFO is overrun. **Note:** This bit will be cleared by writing 1 to it.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the receive FIFO buffer is smaller than or equal to the setting value of RXTH. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH.
[9]	**RXFULL**	**Receive FIFO Buffer Full Indicator (Read Only)** 0 = Receive FIFO buffer is not full. 1 = Receive FIFO buffer is full.
[8]	**RXEMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[7]	**SLVURIF**	**Slave Mode TX Under Run Interrupt Flag** In Slave mode, if TX underflow event occurs and the slave select line goes to inactive state, this interrupt flag will be set to 1. 0 = No Slave TX under run event. 1 = Slave TX under run event occurred. **Note:** This bit will be cleared by writing 1 to it.
[6]	**SLVBEIF**	**Slave Mode Bit Count Error Interrupt Flag** In Slave mode, when the slave select line goes to inactive state, if bit counter is mismatch with DWIDTH, this interrupt flag will be set to 1. 0 = No Slave mode bit count error event. 1 = Slave mode bit count error event occurred. **Note:** If the slave select active but there is no any bus clock input, the SLVBEIF also active when the slave select goes to inactive state. This bit will be cleared by writing 1 to it.
[5]	**SLVTOIF**	**Slave Time-out Interrupt Flag** When the slave select is active and the value of SLVTOCNT is not 0, if the bus clock is detected, the slave time-out counter in QSPI controller logic will be started. When the

		value of time-out counter is greater than or equal to the value of SLVTOCNT (QSPIx_SSCTL[31:16]) before one transaction is done, the slave time-out interrupt event will be asserted. 0 = Slave time-out is not active. 1 = Slave time-out is active. **Note:** This bit will be cleared by writing 1 to it.
[4]	**SSLINE**	**Slave Select Line Bus Status (Read Only)** 0 = The slave select line status is 0. 1 = The slave select line status is 1. **Note:** This bit is only available in Slave mode. If SSACTPOL (QSPIx_SSCTL[2]) is set 0, and the SSLINE is 1, the QSPI slave select is in inactive status.
[3]	**SSINAIF**	**Slave Select Inactive Interrupt Flag** 0 = Slave select inactive interrupt was cleared or not occurred. 1 = Slave select inactive interrupt event occurred. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[2]	**SSACTIF**	**Slave Select Active Interrupt Flag** 0 = Slave select active interrupt was cleared or not occurred. 1 = Slave select active interrupt event occurred. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[1]	**UNITIF**	**Unit Transfer Interrupt Flag** 0 = No transaction has been finished since this bit was cleared to 0. 1 = QSPI controller has finished one unit transfer. **Note:** This bit will be cleared by writing 1 to it.
[0]	**BUSY**	**Busy Status (Read Only)** 0 = QSPI controller is in idle state. 1 = QSPI controller is in busy state. The following lists the bus busy conditions: j. SPIEN (QSPIx_CTL[0]) = 1 and TXEMPTY = 0. k. For QSPI Master mode, SPIEN (QSPIx_CTL[0]) = 1 and TXEMPTY = 1 but the current transaction is not finished yet. l. For QSPI Master mode, SPIEN (QSPIx_CTL[0]) = 1 and RXONLY = 1. m. For QSPI Slave mode, SPIEN (QSPIx_CTL[0]) = 1 and there is serial clock input into the QSPI core logic when slave select is active. **n.** For QSPI Slave mode, SPIEN (QSPIx_CTL[0]) = 1 and the transmit buffer or transmit shift register is not empty even if the slave select is inactive. **Note:** By applications, this QSPI busy flag should be used with other status registers in QSPIx_STATUS such as TXCNT, RXCNT, TXTHIF, TXFULL, TXEMPTY, RXTHIF, RXFULL, RXEMPTY, and UNITIF. Therefore the QSPI transfer done events of TX/RX operations can be obtained at correct timing point.

QSPI Status2 Register (QSPIx_STATUS2)

Register	Offset	R/W	Description	Reset Value
QSPIx_STATUS2	QSPIx_BA+0x18	R	QSPI Status2 Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		SLVBENUM					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**SLVBENUM**	**Effective Bit Number of Uncompleted RX Data** This status register indicates that effective bit number of uncompleted RX data when SLVBERX (QSPIx_FIFOCTL[10]) is enabled and RX bit count error event happen in QSPI slave mode. This status register will be fixed to 0x0 when SLVBERX (QSPIx_FIFOCTL[10]) is disabled. **Note:** This register will be cleared to 0x0 when user write 0x1 to SLVBEIF (QSPIx_STATUS[6]).
[23:0]	**Reserved**	Reserved.

QSPI Data Transmit Register (QSPIx_TX)

Register	Offset	R/W	Description	Reset Value
QSPIx_TX	QSPIx_BA+0x20	W	QSPI Data Transmit Register	0x0000_0000

31	30	29	28	27	26	25	24
TX							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TX							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TX							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TX							

Bits	Description	
[31:0]	**TX**	**Data Transmit Register** The data transmit registers pass through the transmitted data into the 8-level transmit FIFO buffers. The number of valid bits depends on the setting of DWIDTH (QSPIx_CTL[12:8]) in QSPI mode. In QSPI mode, if DWIDTH is set to 0x08, the bits TX[7:0] will be transmitted. If DWIDTH is set to 0x00, the QSPI controller will perform a 32-bit transfer. **Note:** In Master mode, QSPI controller will start to transfer the QSPI bus clock after 1 APB clock and 6 peripheral clock cycles after user writes to this register.

QSPI Data Receive Register (QSPIx_RX)

Register	Offset	R/W	Description	Reset Value
QSPIx_RX	QSPIx_BA+0x30	R	QSPI Data Receive Register	0x0000_0000

31	30	29	28	27	26	25	24
RX							
23	22	21	20	19	18	17	16
RX							
15	14	13	12	11	10	9	8
RX							
7	6	5	4	3	2	1	0
RX							

Bits	Description	
[31:0]	**RX**	**Data Receive RegisterThere are 8-level FIFO buffers in this controller The data receive register holds the data received from QSPI data input pin If the RXEMPTY (QSPIx_STATUS[8) is not set to 1, the receive FIFO buffers can be accessed through software by reading this register**(Read Only)

6.29 SPI Synchronous Serial Interface Controller (SPI Master Mode)

6.29.1 Overview

The SPI Synchronous serial Interface Controller for SPI master mode performs a serial-to-parallel conversion on data received from the peripheral, and a parallel-to-serial conversion on data received from MCU. This SPI controller can drive one external peripheral (External SPI Flash) and it is seen as the SPI master mode. It can generate an interrupt signal when data transfer is finished and can be cleared by writing 1 to the interrupt flag. The active level of device/slave select signal can be chosen to low active or high active, which depends on the peripheral. Writing a divisor into the SPIM_CTL1 register can program the frequency of serial clock output to the peripheral.

In SPI Flash controller, normal I/O mode contains four 32-bit transmit/receive buffers, and can provide 1 to 4 burst mode operation. The number of bits in each transaction can be 8, 16, 24, or 32; data can be transmitted/received up to four successive transactions in one transfer.

By DMA write mode, user can move data from SRAM to external SPI Flash component. In DMA read mode, user can move data from external SPI Flash component to SRAM. In direct memory mapping mode (DMM mode), this SPI Flash controller will translate the AHB bus commands into SPI Flash operations without MCU setting related SPI Flash command. Therefore users can access external SPI Flash as a ROM module.

In direct memory mapping mode with cache off mode, it will pre-fetch 4-word Flash data after a direct memory mapping access. When using direct memory mapping mode with cache on mode, it will use 16 Kbytes cache memory to reduce the number of accessing external SPI Flash component and the performance of SPI Flash access can be improved. To improve the read operation of SPI Flash without increasing the serial clock frequency, this SPI Flash controller supports DTR (Double Transfer Rate) read command codes that support Standard/Dual/Quad SPI modes. The one byte command code is still latched into the device on the rising edge of the serial clock similar to all other SPI commands. Once a DTR instruction code is accepted by the device, the address input and data output will be latched on both rising and falling edges of the serial clock.

For data protection, this SPI Flash controller supports cipher encryption and decryption circuits to protect data which user places into external SPI Flash when DMA read/write mode and direct memory mapping mode are used.

6.29.2 Features

- Supports maximum 32 Mbytes SPI Flash size
- Supports SPI master mode
- Supports Direct Memory Mapping Mode and Normal I/O Mode
- Supports 8/16/24/32 bits transaction for Normal I/O mode
- Provides burst mode operation in Normal I/O mode, which can transmit/receive data up to four successive transactions in one transfer
- Supports DMA mode read/write
- Supports standard (1-bit), dual (2-bit), and quad (4-bit) I/O transfer mode
- Supports Double Transfer Rate (DTR) mode
- Supports 16 Kbytes cache memory
- Supports Cipher encryption/decryption
- One slave/device select line for external SPI Flash component

6.29.3 Block Diagram

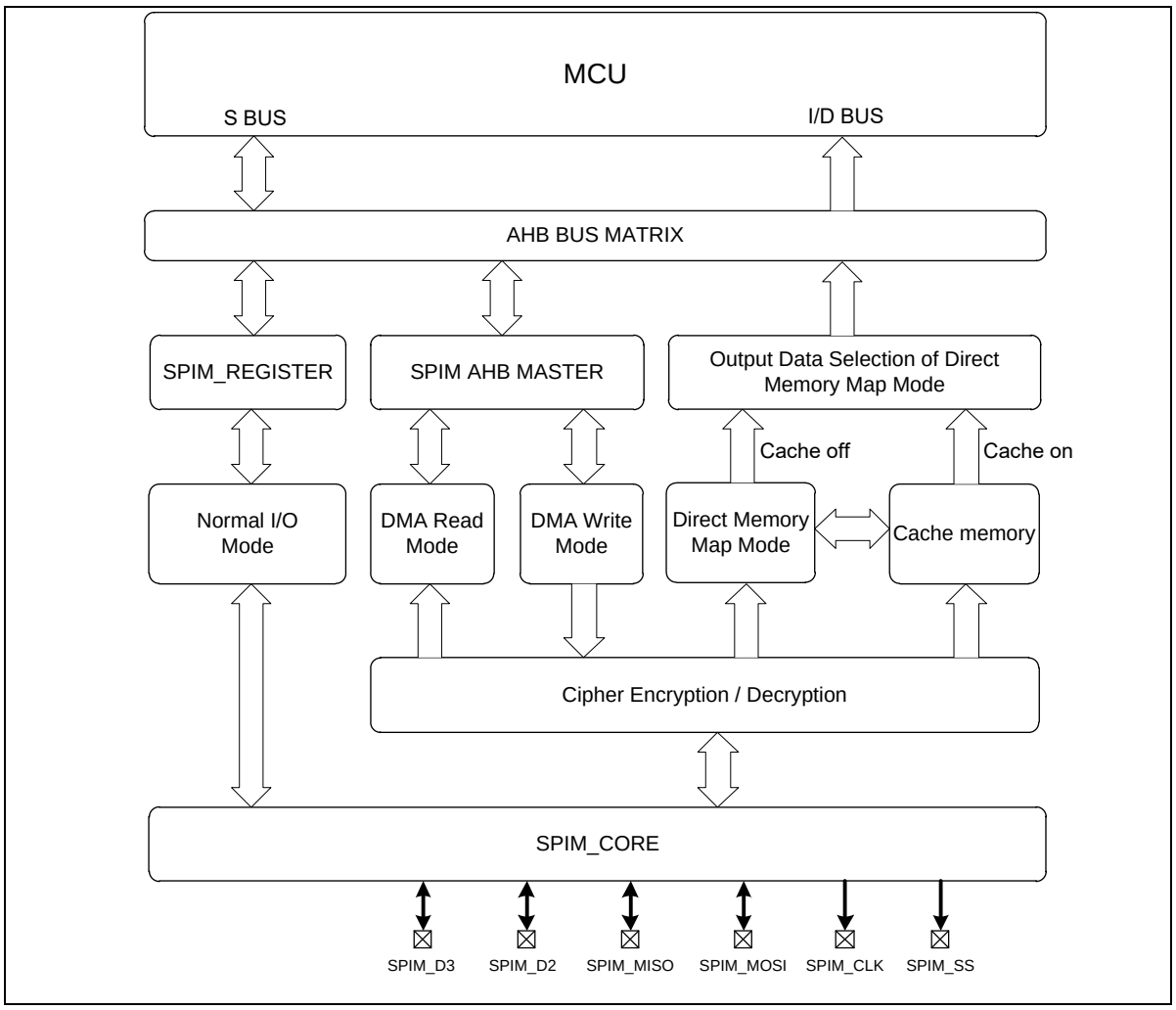


Figure 6.29-1 SPIM Block Diagram

6.29.4 Basic Configuration

- Clock Source Configuration
  - Enable SPIM peripheral clock in SPIMCKEN (CLK_AHBCLK[14]).
- Reset Configuration
  - Reset SPIM controller in SPIMRST (SYS_IPRST0[14]).

6.29.5 Functional Description

6.29.5.1 SPIM Timing Diagram

The timing diagram of SPI transaction is shown in Figure 6.29-2.

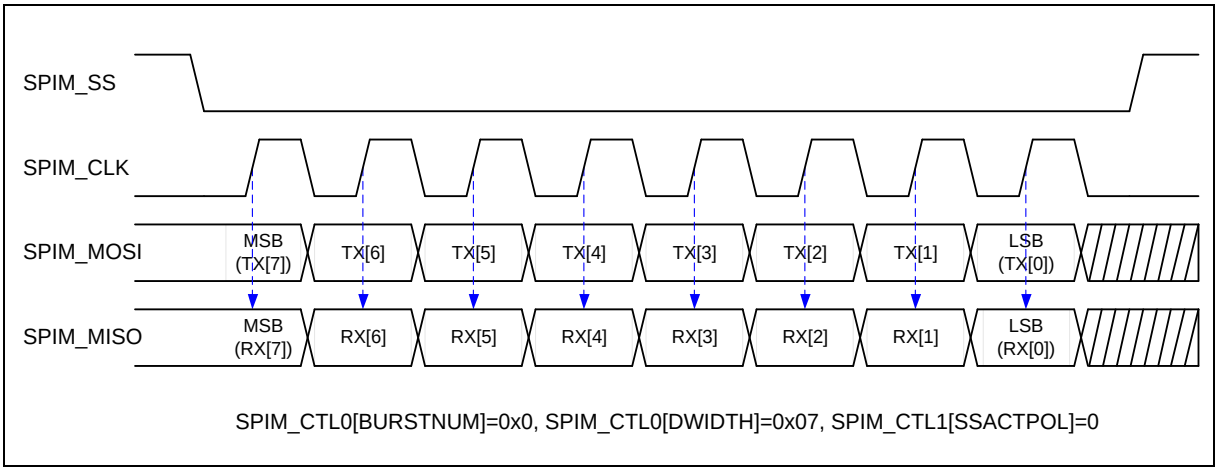


Figure 6.29-2 SPIM Timing Diagram for TX and RX Transfers

6.29.5.2 SPIM Slave Select Control

Table 6.29-1 shows the relationship between slave select active level "SSACTPOL" (SPIM_CTL1[5]) and slave select active enable bit "SS" (SPIM_CTL1[4]).

SSACTPOL	SS	Functional Descriptions
0	0	SPIM_SS is active low, and set SPIM_SS to 0
0	1	SPIM_SS is active low, and set SPIM_SS to 1
1	0	SPIM_SS is active high, and set SPIM_SS to 1
1	1	SPIM_SS is active high, and set SPIM_SS to 0

Table 6.29-1 Functional Description for Slave Select Active Level and Slave Active Enable

In Figure 6.29-3, control register "ACTSCLKT" (SPIM_DMMCTL[31:28]) sets time interval between SPIM SS active edge and the position edge of the first serial SPI output clock, and control register "DESELTIM" (SPIM_DMMCTL[20:16]) sets the minimum time width of SPI Flash deselect time (i.e. Minimum SPIM_SS deselect time).

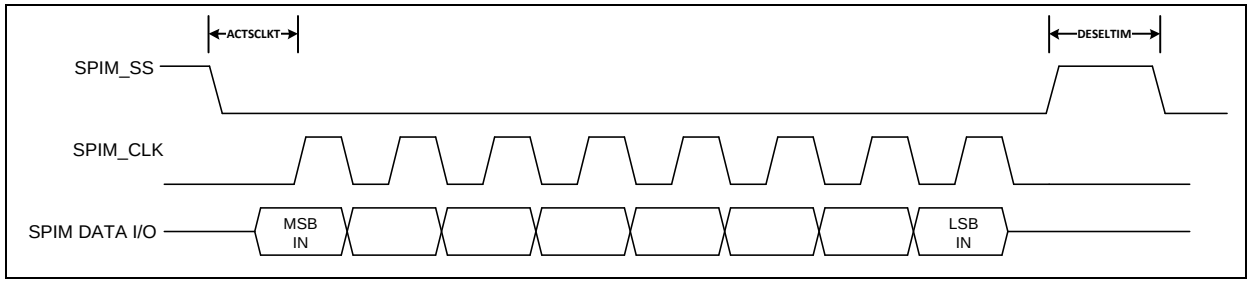


Figure 6.29-3 SPIM_SS Active Setup Time Relative to SPIM_CLK, and SPIM_SS Deselect Time

6.29.5.3 SPIM Programming Example without DMA (Normal I/O Mode)

To access a device with the following requirements, where the timing diagram is shown in Figure 6.29-2:

- Data is transferred with the MSB first.
- Only one byte is transmitted/received in a transfer.
- Chip select signal is active low.
- User should do the following actions basically (refer to the specification of the device for

the detailed steps):

1. Write a divisor into DIVIDER (SPIM_CTL1[31:16]) to determine the frequency of serial clock.
2. Set SSACTPOL (SPIM_CTL1[5]) to 0 to activate the device you want to access.
3. When transmitting (writing) data to device:
   - QDIODIR (SPIM_CTL0[15]) = 1
   - BURSTNUM (SPIM_CTL0[14:13]) = 0x0
   - DWIDTH (SPIM_CTL0[12:8]) = 0x07
   - Write the data you want to transmit into SPIM_TX0[7:0]
4. When receiving (reading) data from device:
   - QDIODIR (SPIM_CTL0[15]) = 0
   - BURSTNUM (SPIM_CTL0[14:13]) = 0x0
   - DWIDTH (SPIM_CTL0[12:8]) = 0x07
5. Set SPIMEN (SPIM_CTL1[0]) to 1 to start the transfer.
6. Wait for interrupt or poll the SPIMEN (SPIM_CTL1[0]) until it turns to 0.
7. When receiving (reading) data from device:
   - Read out the received data from the SPIM_RX0[7:0] register.

6.29.5.4 SPIM Programming Example with DMA

If user wants to access a device with DMA function, three additional registers need to be configured, including SPIM_DMACNT, SPIM_SRAMADDR and SPIM_FADDR. The DMA function can be used to support loading code (reading data from the peripheral into system memory) or reading data from system memory and storing the data in the peripheral. Users must define the source, length and destination, and then hardware will automatically move the desired length of code to the specific target address. It is recommended to send disable commands of continuous read mode and burst wrap mode to external SPI Flash component before using DMA read/write operations to transfer data between SRAM and SPI Flash.

6.29.5.5 Move Data from SPI Flash to System Memory (DMA Read Mode)

Step 1: Check device ID by using normal I/O mode to confirm the device has connected.

Step 2:

1. Set the target memory address into the SPIM_SRAMADDR register.
2. Set the code length into the SPIM_DMACNT register.
3. Set the SPI Flash start address in the SPIM_FADDR register.
4. Set OPMODE (SPIM_CTL0[23:22]) with the DMA Read mode.
5. Set CMDCODE (SPIM_CTL0[31:24]) with the 0x03 command code.
6. Set SPIMEN (SPIM_CTL1[0]) to start. Then SPIM will move the code block from SPIM_FADDR to SPIM_SRAMADDR with the transfer length set in SPIM_DMACNT.
7. Wait for interrupt or poll the SPIMEN (SPIM_CTL1[0]) bit until it turns to 0.

If the used SPI Flash supports other associated read commands, users can also use the following read command codes.

- Fast Read (0Bh), set CMDCODE (SPIM_CTL0[31:24])with 0x0B.
- Fast Read Dual Output (3Bh), set CMDCODE (SPIM_CTL0[31:24]) with 0x3B.

- Fast Read Dual I/O (BBh), set CMDCODE (SPIM_CTL0[31:24]) with 0xBB.
- Fast Read Quad I/O (EBh), set CMDCODE (SPIM_CTL0[31:24]) with 0xEB.
- Word Read Quad I/O (E7h), set CMDCODE (SPIM_CTL0[31:24]) with 0xE7.
- DTR Fast Read (0Dh), set CMDCODE (SPIM_CTL0[31:24]) with 0x0D.
- DTR Fast Read Dual I/O (BDh), set CMDCODE (SPIM_CTL0[31:24]) with 0xBD.
- DTR Fast Read Quad I/O (EDh), set CMDCODE (SPIM_CTL0[31:24]) with 0xED.

6.29.5.6 Move Data from System Memory to SPI Flash (DMA Write Mode)

Step 1: Erase the SPI Flash (using normal I/O mode) before programming it.

Step 2:

1. Send Write Enable command to SPI Flash (using normal I/O mode).
2. Set the source memory address in the SPIM_SRAMADDR register.
3. Set the transfer count into the SPIM_DMACNT register.
4. Set the SPI Flash start address in the SPIM_FADDR register.
5. Set OPMODE (SPIM_CTL0[23:22]) with the DMA Write mode.
6. Set CMDCODE (SPIM_CTL0[31:24]) with the 0x02 command code.
7. Set SPIMEN (SPIM_CTL1[0]) to start.
8. Wait for interrupt or poll the SPIMEN (SPIM_CTL1[0]) until it turns to 0.

If the used SPI Flash supports other associated write commands, users can also use the following command code. (User needs to refer to specification of external SPI Flash component to select suitable quad write mode command.)

- Quad write, set CMDCODE (SPIM_CTL0[31:24]) to 0x32 for TYPE_1 program command (refer to Figure 6.29-4).
- Quad write, set CMDCODE (SPIM_CTL0[31:24]) to 0x38 for TYPE_2 program command (refer to Figure 6.29-5).
- Quad write, set CMDCODE (SPIM_CTL0[31:24]) to 0x40 for TYPE_3 program command (refer to Figure 6.29-6).

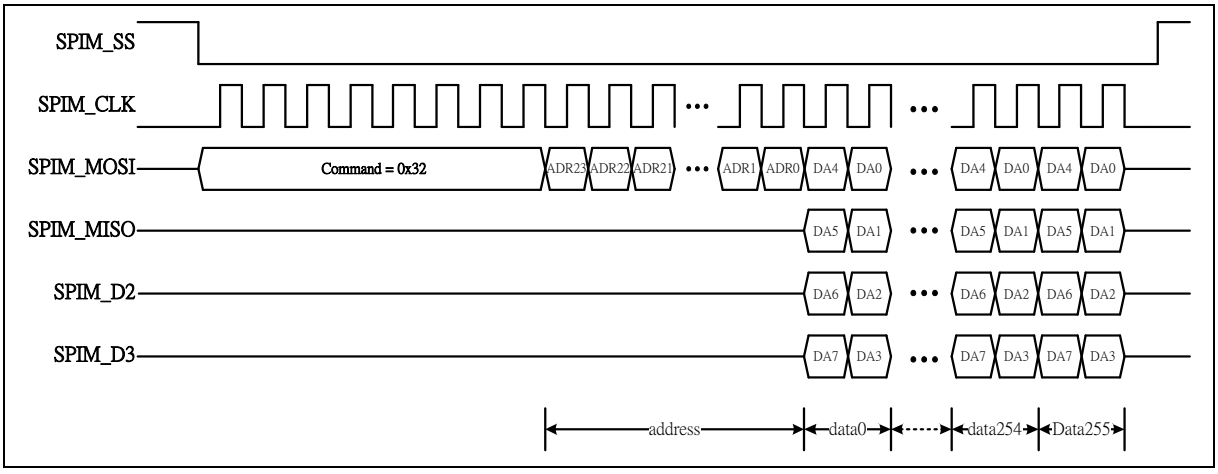


Figure 6.29-4 TYPE_1 Program Command Flow with Quad Mode Write

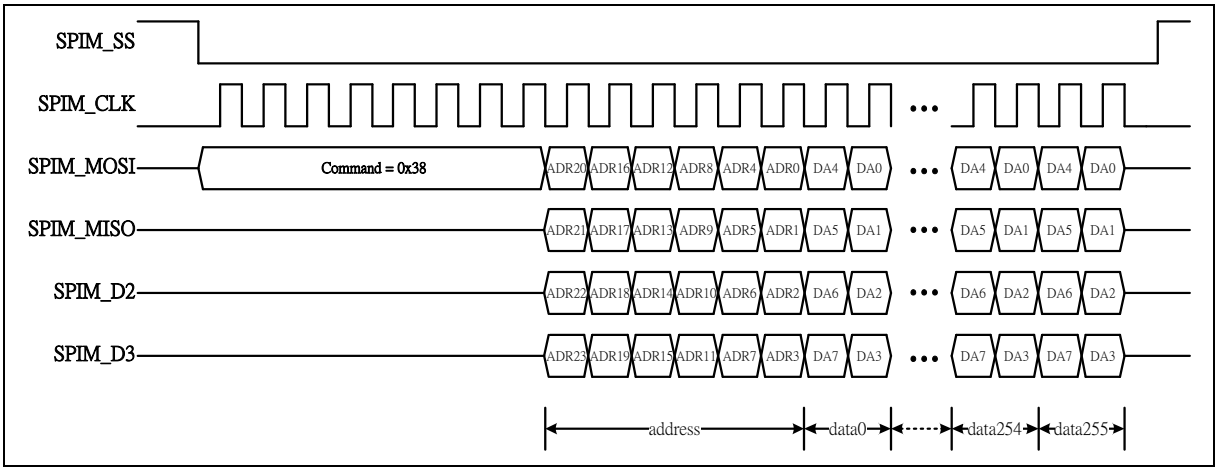


Figure 6.29-5 TYPE_2 Program Command Flow with Quad Mode Write

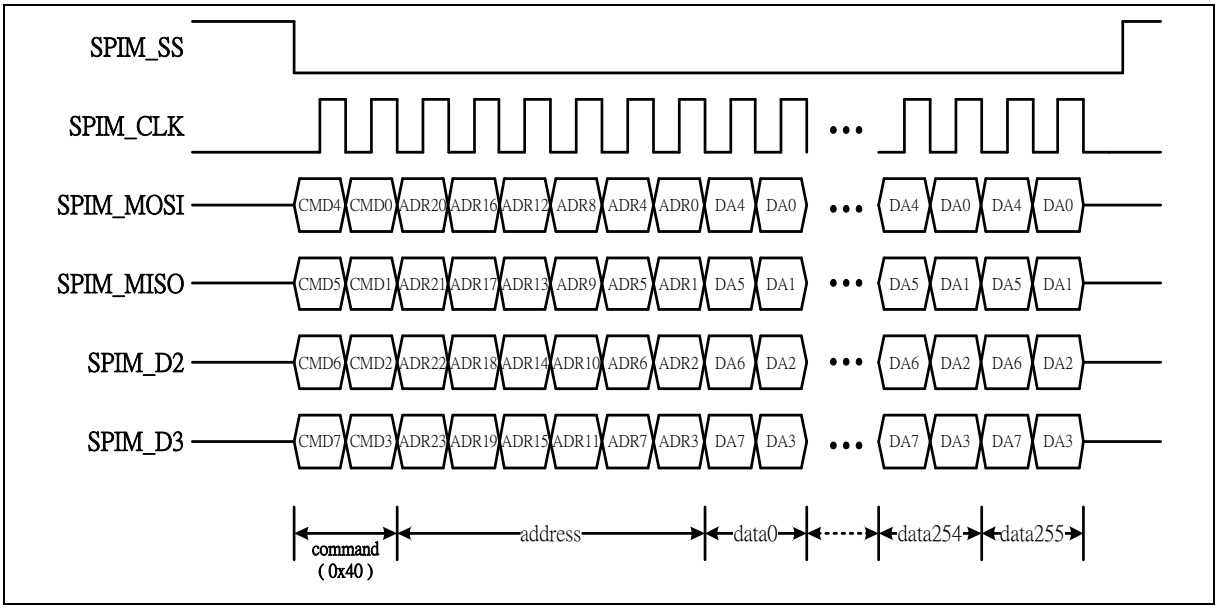


Figure 6.29-6 TYPE_3 Program Command Flow with Quad Mode Write

6.29.5.7 Direct Memory Mapping Mode

SPI Flash can be regarded as a ROM module in direct memory mapping mode. The controller will convert the AHB cycle into SPI Flash without MCU setting related SPI Flash command. The only setting MCU needs to do is to set CMDCODE(SPIM_CTL0[31:24]) with command codes 0x03, 0x0B, 0x3B, 0xBB, 0xEB, 0xE7, 0x0D, 0xBD, or 0xED. Then users can access SPI Flash as a ROM module.

In direct memory mapping mode with cache off mode, it will pre-fetch 4-word Flash data after a direct memory mapping access. In direct memory mapping mode with cache function on, it will use cache memory to reduce the number of accessing external SPI Flash component. Therefore the performance of SPI Flash access can be improved. If users want to modify SPIM hardware registers after the direct memory mapping access, wait for at least 250 peripheral cycles (SPIM bus cycles).

In direct memory mapping mode (DMM mode), quad/dual read commands codes “0xEB, 0xE7, 0xBB” and DTR read command codes “0x0D, 0xBD, 0xED” support continuous read mode (or performance enhance mode) to reduce the command sequence by eight clocks, and allows the read address data to be entered after chip select of SPI Flash component is asserted low (please check the used SPI Flash specification to know support information). Figure 6.29-7 shows the SPI data transfer of quad I/O mode when the continuous read mode is disabled and address mode is 3 bytes.

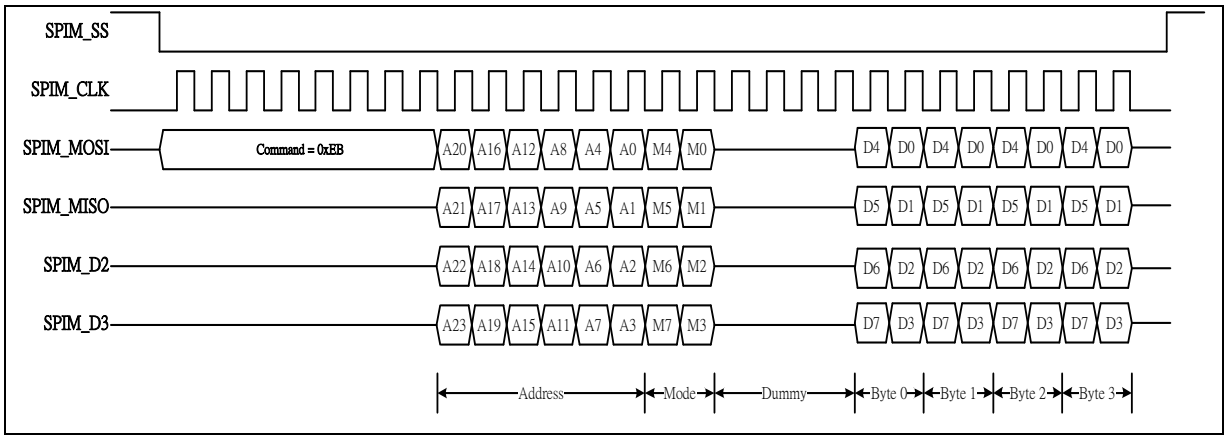


Figure 6.29-7 Fast Read Quad I/O Command 0xEB with Continuous Read Mode Disable

Figure 6.29-8 shows the next SPI data transfer of quad I/O mode when the continuous read mode is enabled and address mode is 3 bytes.

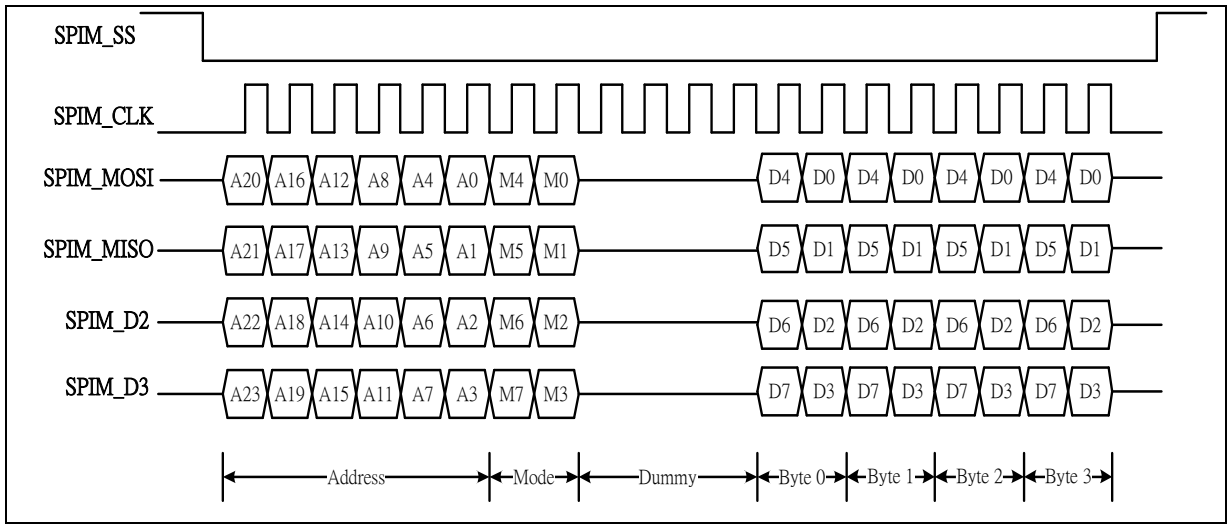


Figure 6.29-8 Fast Read Quad I/O Command 0xEB with Continuous Read Mode Enable

List for supported SPI Flash vendors of continuous read mode for read command 0xBB, 0xEB, 0xE7, 0x0D, 0xBD, and 0xED in direct memory mapping mode:

- Supported SPI Flash vendors for read command 0xBB: Winbond only
- Supported SPI Flash vendors for read command 0xEB: Winbond, Gigadevice, Spansion, EON, and MXIC
- Supported SPI Flash vendors for read command 0xE7: Winbond only
- Supported SPI Flash vendors for DTR read command 0x0D: Check used SPI Flash specification
- Supported SPI Flash vendors for DTR Dual I/O read command 0xBD: Check used SPI Flash specification
- Supported SPI Flash vendors for DTR Qual I/O read command 0xED: Check used SPI Flash specification

After using command codes 0xEB, 0xE7, 0x0D, 0xBD, and 0xED to read the last needed SPI Flash data in DMM mode with the continuous read mode on, please use normal I/O mode to send 0xFF (8 SPI clocks for 3 bytes address mode) or 0x3FF (10 SPI clocks for 4 bytes address mode) to the external SPI Flash on SPI serial interface pin sdo_0 to disable continuous read mode of the external SPI Flash. Check the used SPI Flash specification to know the information of continuous read mode (or performance enhance mode).

When using command codes “0xBB” to read the last needed SPI Flash data in DMM mode with the continuous read mode on, please use normal I/O mode to send 0xFFFF (16 SPI clocks for 3 bytes address mode) or 0xFFFFF (20 SPI clocks for 4 bytes address mode) to the external SPI Flash on SPI serial interface pin sdo_0. Please check the used SPI Flash specification to know the information of continuous read mode (or performance enhance mode).

Use the following settings of normal I/O mode to send continuous read mode reset to the external SPI Flash. Please check the used SPI Flash specification to know the information of continuous read mode (or performance enhance mode).

Steps for continuous read mode reset (disable) for command codes 0xEB, 0xE7, 0x0D, 0xBD, 0xED in direct memory mapping mode:

1. Set OPMODE(SPIM_CTL0[23:22]) to 0x0 (normal I/O mode)

2. Set BITMODE(SPIM_CTL0[21:20]) to 0x2 (quad mode, use 4 serial SPI signals to send data to SPI Flash)
3. Set QDIODIR(SPIM_CTL0[15]) to 0x1 (write mode)
4. Set BURSTNUM(SPIM_CTL0[14:13]) to 0x0 (only one transmit/receive transaction will be executed in one transfer)
5. Set DWIDTH(SPIM_CTL0[12:8]) to 0x1F (send 32 bits data to SPI Flash)
6. Set SPIM_TX0 register to 0xFFFFFFFF
7. Set SPIMEN(SPIM_CTL1[0]) to start normal I/O operations
8. Polling SPIMEN(SPIM_CTL1[0]) to check operation status when normal I/O mode is going. (1 = busy, 0 = finish)
9. Set SS(SPIM_CTL1[4]) to let SPIM_SS in inactive level

Steps for continuous read mode reset for command code 0xBB in direct memory mapping mode:

1. Set OPMODE(SPIM_CTL0[23:22]) to 0x0 (normal I/O mode)
2. Set BITMODE(SPIM_CTL0[21:20]) to 0x1 (dual mode, use 2 serial SPI signals to send data to SPI Flash)
3. Set QDIODIR(SPIM_CTL0[15]) to 0x1 (write mode)
4. Set BURSTNUM(SPIM_CTL0[14:13]) to 0x0 (only one transmit/receive transaction will be executed in one transfer)
5. Set DWIDTH(SPIM_CTL0[12:8]) to 0x1F (send 32 bits data to SPI Flash)
6. Set SPIM_TX0 register to 0xFFFFFFFF
7. Set SPIMEN(SPIM_CTL1[0]) to start normal I/O operations
8. Polling SPIMEN(SPIM_CTL1[0]) to check operation status when normal I/O mode is going. (1 = busy, 0 = finish)
9. Set SS(SPIM_CTL1[4]) to let SPIM_SS in inactive level

In direct memory mapping mode, both of quad read commands “0xEB” and “0xE7” support burst wrap mode for cache application and performance enhance. For cache application, the burst wrap mode can be used to fill the cache line quickly (In this SPI Flash controller, use cache data line with 16 bytes size). For performance enhance with direct memory mapping mode and cache enable, when cache data is miss, the burst wrap mode can let MCU get the required SPI Flash data quickly. When both of the burst wrap mode and continuous read mode are switched from enable to disable, the continuous read mode of the external SPI Flash is disabled firstly and then the burst wrap mode of the external SPI Flash is disabled secondly.

Before DMM mode with read command codes “0xEB or 0xE7” and burst wrap mode is used to access external SPI Flash, please use normal I/O mode of SPI Flash controller to send “0x1101_1101 and 0x2000_0000” (burst wrap enable command of external SPI Flash, it needs 16 SPI clocks. Please check used SPI Flash specification to know information and setting values for burst wrap mode.) to the external SPI Flash on SPI serial interface pin sdo_0 ~ sdo_3. After using command codes “0xEB and 0xE7” to read the last needed SPI Flash data in DMM mode with the burst wrap mode on, please use normal I/O mode of SPI Flash controller to send “0x1101_1101 and 0x0000_0000” (burst wrap disable command of external SPI Flash, it needs 16 SPI clocks. Please check used SPI Flash specification to know information and setting values for burst wrap mode.) to the external SPI Flash on SPI serial interface pin sdo_0 ~ sdo_3.

Use the following settings of normal I/O mode of SPI Flash controller to send SPI command codes to enable or disable the burst wrap mode for the external SPI Flash.

Steps for burst wrap mode enable in direct memory mapping mode:

1. Set OPMODE(SPIM_CTL0[23:22]) to 0x0 (normal I/O mode)
2. Set BITMODE(SPIM_CTL0[21:20]) to 0x2 (quad mode, use 4 serial SPI signals to send data to SPI Flash)
3. Set QDIODIR(SPIM_CTL0[15]) to 0x1 (write mode)
4. Set BURSTNUM(SPIM_CTL0[14:13]) to 0x1 (two successive/receive transaction will be executed in one transfer)
5. Set DWIDTH(SPIM_CTL0[12:8]) to 0x1F (send 32 bits data to SPI Flash)
6. Set SPIM_TX0 register to 0x20000000, SPIM_TX1 register to 0x11011101
7. Set SPIMEN(SPIM_CTL1[0]) to start normal I/O operations
8. Polling SPIMEN(SPIM_CTL1[0]) to check operation status when normal I/O mode is going. (1 = busy, 0 = finish)
9. Set SS(SPIM_CTL1[4]) to let SPIM_SS in inactive level

Steps for burst wrap mode disable in direct memory mapping mode:

1. Set OPMODE(SPIM_CTL0[23:22]) to 0x0 (normal I/O mode)
2. Set BITMODE(SPIM_CTL0[21:20]) to 0x2 (quad mode, use 4 serial SPI signals to send data to SPI Flash)
3. Set QDIODIR(SPIM_CTL0[15]) to 0x1 (write mode)
4. Set BURSTNUM(SPIM_CTL0[14:13]) to 0x1 (two successive/receive transaction will be executed in one transfer)
5. Set DWIDTH(SPIM_CTL0[12:8]) to 0x1F (send 32 bits data to SPI Flash)
6. Set SPIM_TX0 register to 0x00000000, SPIM_TX1 register to 0x11011101
7. Set SPIMEN(SPIM_CTL1[0]) to start normal I/O operations
8. Polling SPIMEN(SPIM_CTL1[0]) to check operation status when normal I/O mode is going. (1 = busy, 0 = finish)
9. Set SS(SPIM_CTL1[4]) to let SPIM_SS in inactive level

To improve the read operation throughput without increasing the serial clock frequency, this SPI Flash controller supports DTR (Double Transfer Rate) Read command codes (i.e. 0x0D, 0xBD, 0xED) that support Standard/Dual/Quad SPI modes for direct memory mapping mode and DMA read mode. The one byte command code is still latched into the device on the 8 rising edges of the serial clock. Once a DTR instruction code is accepted by the device, the SPI Flash read address, mode bits, dummy cycles, and output data, which sends from SPI Flash component will be latched on both rising and falling edges of the serial clock. The timing diagrams for DTR read command 0x0D, 0xBD, and 0xED are shown in Figure 6.29-9, Figure 6.29-10, and Figure 6.29-11.

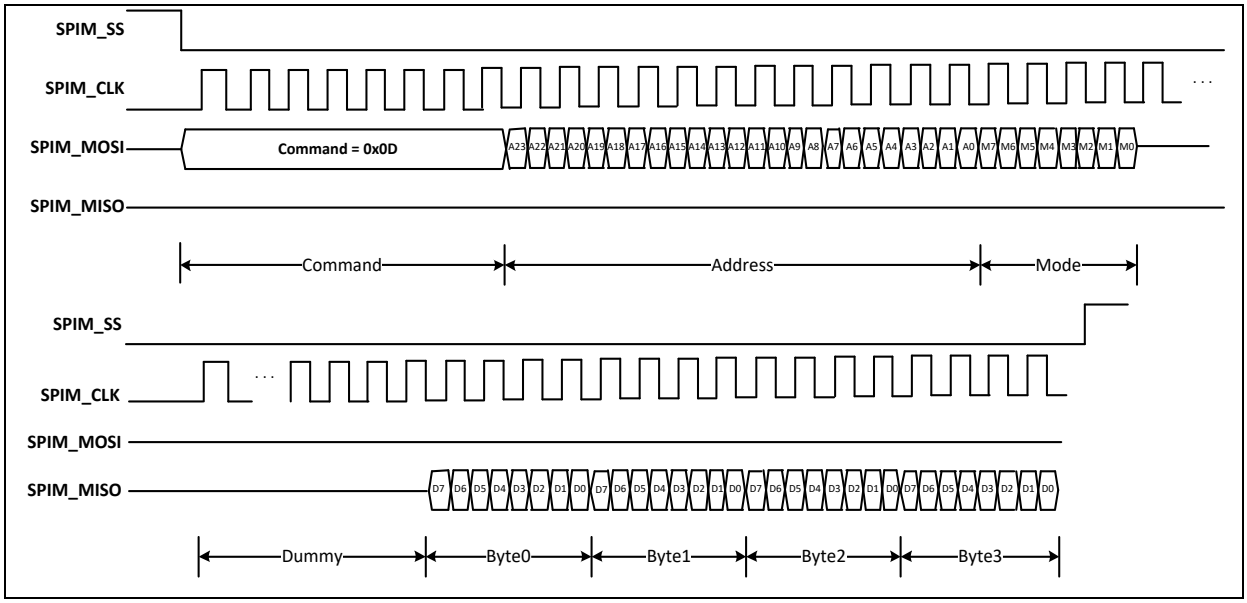


Figure 6.29-9 DTR Fast read command 0x0D

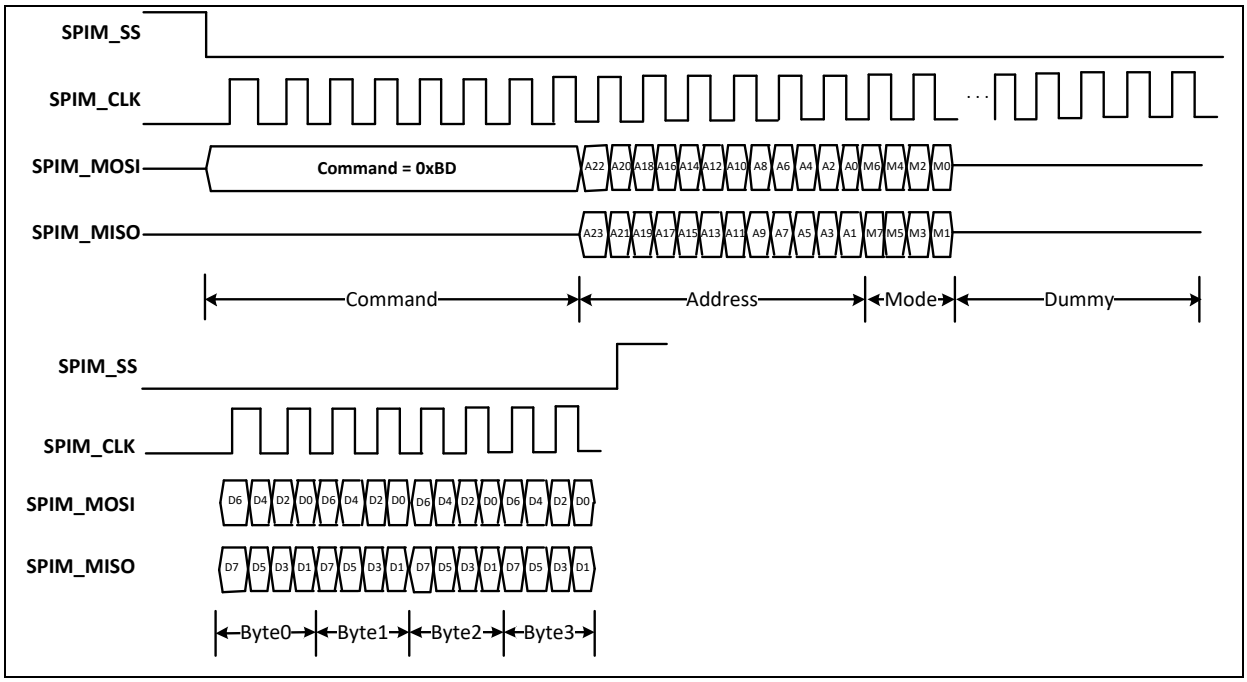


Figure 6.29-10 DTR Fast Read Dual I/O Command 0xBD

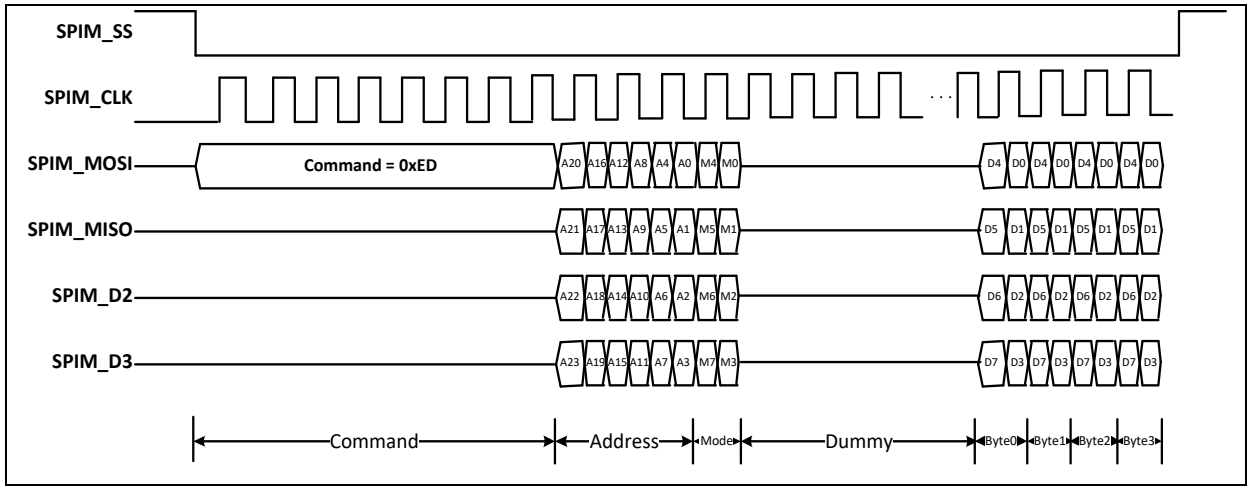


Figure 6.29-11 DTR Fast Read Quad I/O Command 0xED

The dummy cycle number depends on SPI Flash vendors, operation frequencies, and command types, and please check used SPI Flash specification to know dummy cycle number correctly.

6.29.5.8 Cipher Encryption and Decryption

In the SPI Flash controller, cipher encryption and decryption functions are supported to protect data which user places into external SPI Flash. When DMA write mode is used and cipher function is enabled, cipher circuit will encrypt the data which move from SRAM to external SPI Flash. When DMA read mode is used and cipher function is enabled, cipher circuit will decrypt the data which move from external SPI Flash to SRAM. When direct memory mapping mode is used and cipher function is enabled, cipher circuit will decrypt the data which MCU reads from external SPI Flash. In normal I/O mode, the data which send to external SPI Flash or receive from external SPI Flash are not encrypted and decrypted by cipher circuit.

6.29.6 Mapping Address for SPI Flash Controller in AMBA System

Address Space	Controllers
Mapping address for SPIM control register, SPI Flash memory space for DMM mode	
0x4000_7000 – 0x4000_7FFF	SPIM Control Registers
(1) 0x0008_0000 – 0x0207_FFFF (for Note1) (2) 0x0010_0000 – 0x020F_FFFF (for Note2)	Access address range of SPIM Memory Space (32 Mbytes) for Direct Memory Mapping mode (DMM mode). Note 1: The SPIM memory space (32 Mbytes) range is 0x0008_0000 ~ 0x0207_FFFF when internal FLASH size is 512 Kbytes. Note 2: The SPIM memory space (32 Mbytes) range is 0x0010_0000 ~ 0x020F_FFFF when internal FLASH size is 1024 Kbytes.

Table 6.29-2 Mapping Address for SPI Flash Controller in AMBA System

6.29.7 Register Map

R: read only, **W**: write only, **R/W**: both read and write, **C:** Only value 0 can be written

Register	Offset	R/W	Description	Reset Value
SPIM Base Address: **SPIM_BA = 0x4000_7000**				
SPIM_CTL0	SPIM_BA+0x00	R/W	Control and Status Register 0	0x03C0_0003
SPIM_CTL1	SPIM_BA+0x04	R/W	Control Register 1	0x0000_0010
SPIM_RXCLKDLY	SPIM_BA+0x0C	R/W	RX Clock Delay Control Register	0x0000_010F
SPIM_RX0	SPIM_BA+0x10	R	Data Receive Register 0	0x0000_0000
SPIM_RX1	SPIM_BA+0x14	R	Data Receive Register 1	0x0000_0000
SPIM_RX2	SPIM_BA+0x18	R	Data Receive Register 2	0x0000_0000
SPIM_RX3	SPIM_BA+0x1C	R	Data Receive Register 3	0x0000_0000
SPIM_TX0	SPIM_BA+0x20	R/W	Data Transmit Register 0	0x0000_0000
SPIM_TX1	SPIM_BA+0x24	R/W	Data Transmit Register 1	0x0000_0000
SPIM_TX2	SPIM_BA+0x28	R/W	Data Transmit Register 2	0x0000_0000
SPIM_TX3	SPIM_BA+0x2C	R/W	Data Transmit Register 3	0x0000_0000
SPIM_SRAMADDR	SPIM_BA+0x30	R/W	SRAM Memory Address Register	0x0000_0000
SPIM_DMACNT	SPIM_BA+0x34	R/W	DMA Transfer Byte Count Register	0x0000_0000
SPIM_FADDR	SPIM_BA+0x38	R/W	SPI Flash Address Register	0x0000_0000
SPIM_KEY1	SPIM_BA+0x3C	W	Cipher Key1 Register	0x0000_0000
SPIM_KEY2	SPIM_BA+0x40	W	Cipher Key2 Register	0x0000_0000
SPIM_DMMCTL	SPIM_BA+0x44	R/W	Direct Memory Mapping Mode Control Register	0x0008_0000
SPIM_CTL2	SPIM_BA+0x48	R/W	Control Register 2	0x0804_0000

Note: If software wants to write to any register of the peripheral, the SPIMEN bit of the SPIM_CTL1 register should be the status low.

6.29.8 Register Description

Control and Status Register 0 (SPIM_CTL0)

Register	Offset	R/W	Description	Reset Value
SPIM_CTL0	SPIM_BA+0x00	R/W	Control and Status Register 0	0x03C0_0003

31	30	29	28	27	26	25	24
CMDCODE							
23	22	21	20	19	18	17	16
OPMODE		BITMODE		SUSPITV			
15	14	13	12	11	10	9	8
QDIODIR	BURSTNUM		DWIDTH				
7	6	5	4	3	2	1	0
IF	IEN	B4ADDREN	Reserved		BALEN	Reserved	CIPHOFF

Bits	Description	
[31:24]	CMDCODE	**Page Program Command Code** 0x02 = Page program (Used for DMA Write mode). 0x32 = Quad page program with TYPE_1 program flow (Used for DMA Write mode). (Note3) 0x38 = Quad page program with TYPE_2 program flow (Used for DMA Write mode). (Note3) 0x40 = Quad page program with TYPE_3 program flow (Used for DMA Write mode). (Note3) The Others = Reserved. **Read Command Code:** 0x03 = Standard Read (Used for DMA Read/DMM mode). 0x0B = Fast Read (Used for DMA Read/DMM mode). The fast read command code “0x0B” is similar to command code of standard read “0x03” except it can operate at highest possible frequency. (Note2) 0x3B = Fast Read Dual Output (Used for DMA Read/DMM mode). 0xBB = Fast Read Dual I/O (Used for DMA Read/DMM mode). The fast read dual I/O command code “0xBB” is similar to command code of fast read dual output “0x3B” but with capability to input the address bits two bits per clock. (Note2) 0xEB = Fast quad read (Used for DMA Read/DMM mode). 0xE7 = Word quad read (Used for DMA Read/DMM mode). The command code of word quad read “0xE7” is similar to command code of fast quad read “0xEB” except that the lowest address bit must equal to 0 and the number of dummy cycles is less than fast quad read. (Note2) 0x0D = DTR Fast read (Used for DMA Read/DMM mode). 0xBD = DTR dual read (Used for DMA Read/DMM mode). 0xED = DTR quad read (Used for DMA Read/DMM mode). The Others command codes are Reserved. The DTR read commands “0x0D,0xBD,0xED” improves throughput by transferring address and data on both the falling and rising edge of SPI Flash clock (SPIM_CLK). It is similar to those commands “0x0B,

		0xBB, 0xEB" but allows transfer of address and data on rising edge and falling edge of SPI Flash output clock. (Note2) **Note 1:** Quad mode of SPI Flash must be enabled first by normal I/O mode before using quad page program/quad read commands. **Note 2:** See SPI Flash specifications for support items. **Note 3:** For TYPE_1, TYPE_2, and TYPE_3 of page program command code, refer to Figure 6.29-4, Figure 6.29-5, and Figure 6.29-6. **Note 4:** Please disable "continuous read mode" and "burst wrap mode" before DMA write mode of SPI Flash controller is used to program data of external SPI Flash. After using DMA write mode of SPI Flash controller to program the content of external SPI Flash, please set CDINVAL(SPIM_CTL1[3]) to 0x1 (Set all cache data to be invalid).
[23:22]	**OPMODE**	**SPI Function Operation Mode** 0x0 = Normal I/O mode. (Note1) (Note3) 0x1 = DMA write mode. (Note2) (Note3) 0x2 = DMA read mode. (Note3) 0x3 = Direct Memory Mapping mode (DMM mode) (Default). (Note4) **Note 1:** After using Normal I/O mode of SPI Flash controller to program the content of external SPI Flash, please set CDINVAL(SPIM_CTL1[3]) to 0x1 (Set all cache data to be invalid). **Note 2:** In DMA write mode, hardware will send just one page program command per operation. Users must take care of cross-page cases. After using DMA write mode of SPI Flash controller to program the content of external SPI Flash, please set CDINVAL(SPIM_CTL1[3]) to 0x1 (Set all cache data to be invalid). **Note 3:** For external SPI Flash with 32 Mbytes, access address range of external SPI Flash address is from 0x00000000 to 0x01FFFFFF when using Normal I/O mode, DMA write mode, and DMA read mode to write/read external SPI Flash data. Please user check size of used SPI Flash component to know access address range of external SPI Flash. **Note 4:** For external SPI Flash with 32 Mbytes, access address range of external SPI Flash address is from 0x08000000 to 0x09FFFFFF when using Direct Memory mapping mode (DMM mode) to read external SPI Flash data. Please user check size of used SPI Flash component to know access address range of external SPI Flash.
[21:20]	**BITMODE**	**SPI Interface Bit Mode** 0x0 = Standard mode. 0x1 = Dual mode. 0x2 = Quad mode. 0x3 = Reserved. **Note:** Only used for normal I/O mode.
[19:16]	**SUSPITV**	**Suspend Interval** These four bits provide the configuration of suspend interval between two successive transmit/receive transactions in a transfer. The default value is 0x00. When BURSTNUM = 00, setting this field has no effect on transfer. The desired interval is obtained according to the following equation (from the last falling edge of current SPI clock to the first rising edge of next SPI clock): (SUSPITV+2)*period of AHB clock 0x0 = 2 AHB clock cycles. 0x1 = 3 AHB clock cycles. …… 0xE = 16 AHB clock cycles. 0xF = 17 AHB clock cycles. **Note:** Only used for normal I/O mode.
[15]	**QDIODIR**	**SPI Interface Direction Select for Quad and Dual Mode Only** 0 = Interface signals are input. 1 = Interface signals are output. **Note:** Only used for normal I/O mode.
[14:13]	**BURSTNUM**	**Transmit/Receive Burst Number**

		This field specifies how many transmit/receive transactions should be executed continuously in one transfer. 0x0 = Only one transmit/receive transaction will be executed in one transfer. 0x1 = Two successive transmit/receive transactions will be executed in one transfer. 0x2 = Three successive transmit/receive transactions will be executed in one transfer. 0x3 = Four successive transmit/receive transactions will be executed in one transfer. **Note:** Only used for normal I/O Mode.
[12:8]	**DWIDTH**	**Transmit/Receive Bit Length** This specifies how many bits are transmitted/received in one transmit/receive transaction. 0x7 = 8 bits. 0xF = 16 bits. 0x17 = 24 bits. 0x1F = 32 bits. Others = Incorrect transfer result. **Note 1:** Only used for normal I/O mode. **Note 2:** Only 8, 16, 24, and 32 bits are allowed. Other bit length will result in incorrect transfer.
[7]	**IF**	**Interrupt Flag** Write Operation: 0 = No effect. 1 = Write 1 to clear. Read Operation: 0 = The transfer has not finished yet. 1 = The transfer has done.
[6]	**IEN**	**Interrupt Enable Bit** 0 = SPIM Interrupt Disabled. 1 = SPIM Interrupt Enabled.
[5]	**B4ADDREN**	**4-byte Address Mode Enable Bit** 0 = 4-byte address mode Disabled and 3-byte address mode Enabled. 1 = 4-byte address mode Enabled. **Note:** Used for DMA write mode, DMA read mode, and DMM mode.
[4:3]	**Reserved**	Reserved.
[2]	**BALEN**	**Balance the AHB Control Time Between Cipher Enable and Disable Control** When cipher is enabled, the AHB control signal will delay some time caused by the encoding or decoding calculation. Therefore, if setting BALEN to 1, it will make the AHB signal processing time with cipher disabled be equal to that with cipher enabled. **Note:** Only useful when cipher is disabled.
[1]	**Reserved**	Reserved.
[0]	**CIPHOFF**	**Cipher Disable Bit** 0 = Cipher function Enabled. 1 = Cipher function Disabled. **Note 1:** If there is not any KEY1(SPIM_KEY1[31:0]) or KEY2(SPIM_KEY2[31:0]) (KEY1 is 0x0000_0000 or KEY2 is 0x0000_0000), the cipher function will be disabled automatically. **Note 2:** When CIPHOFF(SPIM_CTL0[0]) is 0, both of KEY1(SPIM_KEY1[31:0]) and KEY2(SPIM_KEY2[31:0]) do not equal to 0x0000_0000 (i.e. KEY1≠ 0x0000_0000 and KEY2≠0x0000_0000), cipher encryption/decryption is enabled. **Note 3:** When cipher encryption/decryption is enabled, please set DESELTIM (SPIM_DMMCTL[20:16]) >= 0x10. When cipher encryption/decryption is disabled, please set DESELTIM(SPIM_DMMCTL[20:16]) >= 0x8.

Control Register 1 (SPIM_CTL1)

Register	Offset	R/W	Description	Reset Value
SPIM_CTL1	SPIM_BA+0x04	R/W	Control Register 1	0x0000_0010

31	30	29	28	27	26	25	24
DIVIDER							
23	22	21	20	19	18	17	16
DIVIDER							
15	14	13	12	11	10	9	8
Reserved				IDLETIME			
7	6	5	4	3	2	1	0
Reserved		SSACTPOL	SS	CDINVAL	Reserved	CACHEOFF	SPIMEN

Bits	Description	
[31:16]	**DIVIDER**	**Clock Divider Register** The value in this field is the frequency divider of the AHB clock (HCLK) to generate the serial SPI output clock "SCLK" on the output SPIM_CLK pin. The desired frequency is obtained according to the following equation: $f_{SPIM_CLK} = \frac{f_{HCLK}}{(DIVIDER)*2}$ **Note 1:** When set DIVIDER to zero, the frequency of SPIM_CLK will be equal to the frequency of HCLK. **Note 2:** SCLK is serial SPI output clock. **Note 3:** Each SPI Flash command has the limitation of the maximum operation frequency of SCLK. Please check the specification of the used SPI Flash component to decide the frequency of SPI Flash clock for different command operations of SPI Flash. **Note 4:** For DTR read commands "0x0D, 0xBD, 0xED", the setting values of DIVIDER are only all multiples of 2. For example, 1,2,4,8,16,32,…
[15:12]	**Reserved**	Reserved.
[11:8]	**IDLETIME**	**Idle Time Interval** In DMM mode, IDLETIME is set to control the minimum idle time between two SPI Flash accesses. Minimum idle time = (IDLETIME + 1) * AHB clock cycle time. **Note 1:** Only used for DMM mode. **Note 2:** AHB clock cycle time = 1/AHB clock frequency.
[7:6]	**Reserved**	Reserved.
[5]	**SSACTPOL**	**Slave Select Active Level** It defines the active level of device/slave select signal (SPIM_SS), as shown in Table 6.29-1. 0 = The SPIM_SS slave select signal is active low. 1 = The SPIM_SS slave select signal is active high.
[4]	**SS**	**Slave Select Active Enable Bit** 0 = SPIM_SS is in active level. 1 = SPIM_SS is in inactive level (Default). **Note:** This interface can only drive one device/slave at a given time. Therefore, the slave selects of the

		selected device must be set to its active level before starting any read or write transfer. Functional description of SSACTPOL(SPIM_CTL1[5]) and SS is shown in Table 6.29-1.
[3]	**CDINVAL**	**Cache Data Invalid Enable Bit** Write Operation: 0 = No effect. 1 = Set all cache data to be invalid. This bit is cleared by hardware automatically. Read Operation: No effect **Note:** When SPI Flash memory is page erasing or whole Flash erasing, please set CDINVAL to 0x1. After using normal I/O mode or DMA write mode of SPI Flash controller to program or erase the content of external SPI Flash, please set CDINVAL to 0x1.
[2]	**Reserved**	Reserved.
[1]	**CACHEOFF**	**Cache Memory Function Disable Bit** 0 = Cache memory function Enabled. (Default) 1 = Cache memory function Disabled.
[0]	**SPIMEN**	**Go and Busy Status** Write Operation: 0 = No effect. 1 = Start the transfer. This bit remains set during the transfer and is automatically cleared after transfer finished. Read Operation: 0 = The transfer has done. 1 = The transfer has not finished yet. **Note:** All registers should be set before writing 1 to the SPIMEN bit. When a transfer is in progress, user should not write to any register of this peripheral.

RX Clock Delay Control Register (SPIM_RXCLKDLY)

Register	Offset	R/W	Description	Reset Value
SPIM_RXCLKDLY	SPIM_BA+0x0C	R/W	RX Clock Delay Control Register	0x0000_010F

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			RDEDGE	Reserved	RDDLYSEL		
15	14	13	12	11	10	9	8
PHDELSEL							
7	6	5	4	3	2	1	0
DWDELSEL							

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**RDEDGE**	**Sampling Clock Edge Selection of Received Data for Normal I/O Mode, DMA Read Mode, DMA Write Mode, and Direct Memory Mapping Mode Only** 0 = Use SPI input clock rising edge to sample received data. (Default) 1 = Use SPI input clock falling edge to sample received data.
[19]	**Reserved**	Reserved.
[18:16]	**RDDLYSEL**	**Sampling Clock Delay Selection for Received Data for Normal I/O Mode, DMA Read Mode, DMA Write Mode, and Direct Memory Mapping Mode Only** Determine the number of inserted delay cycles. Used to adjust the sampling clock of received data to latch the correct data. 0x0: No delay. (Default) 0x1: Delay 1 SPI Flash clock. 0x2: Delay 2 SPI Flash clocks. 0x3: Delay 3 SPI Flash clocks. … 0x7: Delay 7 SPI Flash clocks **Note:** The manufacturer or device ID of external SPI Flash component can be used to determine the correct setting value of RDDLYSEL. An example is given as follows. For example, the manufacturer ID and device ID of external SPI Flash for some vendor are 0xEF and 0x1234 separately. First, set RDDLYSEL to 0x0, and use read manufacturer id/device id command to read the manufacturer id of external SPI Flash by using normal I/O mode (the manufacturer id is 0xEF (1110_1111) in this example). If the manufacturer ID which reads from external SPI Flash is 0xF7 (1111_0111), it denotes that manufacturer id is shifted the right by 1 bit and most significant bit (MSB) of manufacturer id is assigned to 1. According to manufacturer id reads from external SPI Flash, RDDLYSEL needs to be set to 0x1 to receive SPI Flash data correctly.
[15:8]	**PHDELSEL**	**SPI Flash Phase Delay Time for DMA Write Mode and DMA Read Mode Only** The bits set phase delay time between command data phase, address data phase, and dummy cycle phase, where SPI Flash controller will send those phase data to external SPI Flash. Phase delay time of DMA write mode and DMA read mode = (PHDELSEL + 1) * AHB clock cycle time **(Note)**.

		Note: AHB clock cycle time = 1/AHB clock frequency.
[7:0]	**DWDELSEL**	**SPI Flash Deselect Time Interval of DMA Write Mode for DMA Write Mode Only** The bits set the deselect time interval of SPI Flash (i.e. time interval of inactive level of SPIM_SS) when SPI Flash controller operates on DMA write mode. **(Note1)** Deselect time interval of DMA write mode = (DWDELSEL + 1) * AHB clock cycle time **(Note2)**. **Note 1:** Please check the used external SPI Flash component to set this register value. In general case, the deselect time interval of SPI Flash is greater than 50 ns when SPI Flash performs the program operation. **Note 2:** AHB clock cycle time = 1/AHB clock frequency.

Data Receive Register 0~3 (SPIM_RX0~3)

Register	Offset	R/W	Description	Reset Value
SPIM_RX0	SPIM_BA+0x10	R	Data Receive Register 0	0x0000_0000
SPIM_RX1	SPIM_BA+0x14	R	Data Receive Register 1	0x0000_0000
SPIM_RX2	SPIM_BA+0x18	R	Data Receive Register 2	0x0000_0000
SPIM_RX3	SPIM_BA+0x1C	R	Data Receive Register 3	0x0000_0000

31	30	29	28	27	26	25	24
RXDAT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RXDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXDAT							

Bits	Description	
[31:0]	**RXDAT**	**Data Receive Register** The Data Receive Registers hold the received data of the last executed transfer. Number of valid RX registers is specified in SPIM_CTL0[BURSTNUM]. If BURSTNUM > 0, received data are held in the most significant RXDAT register first. Number of valid-bit is specified in SPIM_CTL0[DWIDTH]. If DWIDTH is 16, 24, or 32, received data are held in the least significant byte of RXDAT register first. In a byte, received data are held in the most significant bit of RXDAT register first. **Example1**: If SPIM_CTL0[BURSTNUM] = 0x3 and SPIM_CTL1[DWIDTH] = 0x17, received data will be held in the order SPIM_RX3[23:0], SPIM_RX2[23:0], SPIM_RX1[23:0], SPIM_RX0[23:0]. **Example2**: If SPIM_CTL0[BURSTNUM = 0x0 and SPIM_CTL0[DWIDTH] = 0x17, received data will be held in the order SPIM_RX0[7:0], SPIM_RX0[15:8], SPIM_RX0[23:16]. **Example3**: If SPIM_CTL0[BURSTNUM = 0x0 and SPIM_CTL0[DWIDTH] = 0x07, received data will be held in the order SPIM_RX0[7], SPIM_RX0[6], …..,. SPIM_RX0[0].

Data Transmit Register 0~3 (SPIM_TX0~3)

Register	Offset	R/W	Description	Reset Value
SPIM_TX0	SPIM_BA+0x20	R/W	Data Transmit Register 0	0x0000_0000
SPIM_TX1	SPIM_BA+0x24	R/W	Data Transmit Register 1	0x0000_0000
SPIM_TX2	SPIM_BA+0x28	R/W	Data Transmit Register 2	0x0000_0000
SPIM_TX3	SPIM_BA+0x2C	R/W	Data Transmit Register 3	0x0000_0000

31	30	29	28	27	26	25	24
TXDAT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TXDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TXDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXDAT							

Bits	Description	
[31:0]	**TXDAT**	**Data Transmit Register** The Data Transmit Registers hold the data to be transmitted in next transfer. Number of valid TXDAT registers is specified in SPIM_CTL0[BURSTNUM]. If BURSTNUM > 0, data are transmitted in the most significant TXDAT register first. Number of valid-bit is specified in SPIM_CTL0[DWIDTH]. If DWIDTH is 16, 24, or 32, data are transmitted in the least significant byte of TXDAT register first. In a byte, data are transmitted in the most significant bit of TXDAT register first. **Example1**: If SPIM_CTL0[BURSTNUM] = 0x3 and SPIM_CTL1[DWIDTH] = 0x17, data will be transmitted in the order SPIM_TX3[23:0], SPIM_TX2[23:0], SPIM_TX1[23:0], SPIM_TX0[23:0] in next transfer. **Example2**: If SPIM_CTL0[BURSTNUM] = 0x0 and SPIM_CTL0[DWIDTH] = 0x17, data will be transmitted in the order SPIM_TX0[7:0], SPIM_TX0[15:8], SPIM_TX0[23:16] in next transfer. **Example3**: If SPIM_CTL0[BURSTNUM] = 0x0 and SPIM_CTL0[DWIDTH] = 0x07, data will be transmitted in the order SPIM_TX0[7], SPIM_TX0[6], …..,. SPIM_TX0[0] in next transfer.

SRAM Memory Address Register (SPIM_SRAMADDR)

Register	Offset	R/W	Description	Reset Value
SPIM_SRAMADDR	SPIM_BA+0x30	R/W	SRAM Memory Address Register	0x0000_0000

31	30	29	28	27	26	25	24
ADDR							
23	22	21	20	19	18	17	16
ADDR							
15	14	13	12	11	10	9	8
ADDR							
7	6	5	4	3	2	1	0
ADDR							

Bits	Description	
[31:0]	ADDR	**SRAM Memory Address** For DMA Read mode, this is the destination address for DMA transfer. For DMA Write mode, this is the source address for DMA transfer. **Note:** This address must be word-aligned.

DMA Transfer Byte Count Register (SPIM_DMACNT)

Register	Offset	R/W	Description	Reset Value
SPIM_DMACNT	SPIM_BA+0x34	R/W	DMA Transfer Byte Count Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
DMACNT							
15	14	13	12	11	10	9	8
DMACNT							
7	6	5	4	3	2	1	0
DMACNT							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**DMACNT**	**DMA Transfer Byte Count Register** It indicates the transfer length for DMA process. **Note 1:** The unit for counting is byte. **Note 2:** The number must be the multiple of 4. **Note 3:** Please check specification of used SPI Flash to know maximum byte length of page program.

SPI Flash Address Register (SPIM_FADDR)

Register	Offset	R/W	Description	Reset Value
SPIM_FADDR	SPIM_BA+0x38	R/W	SPI Flash Address Register	0x0000_0000

31	30	29	28	27	26	25	24
ADDR							
23	22	21	20	19	18	17	16
ADDR							
15	14	13	12	11	10	9	8
ADDR							
7	6	5	4	3	2	1	0
ADDR							

Bits	Description	
[31:0]	ADDR	**SPI Flash Address Register** For DMA Read mode, this is the source address for DMA transfer. For DMA Write mode, this is the destination address for DMA transfer. **Note 1:** This address must be word-aligned. **Note 2:** For external SPI Flash with 32 Mbytes, the value of this SPI Flash address register “ADDR” is from 0x00000000 to 0x00FFFFFF when using DMA write mode and DMA read mode to write/read external SPI Flash data. Please user check size of used SPI Flash component to know access address range of external SPI Flash.

Cipher Key1 Register (SPIM_KEY1)

Register	Offset	R/W	Description	Reset Value
SPIM_KEY1	SPIM_BA+0x3C	W	Cipher Key1 Register	0x0000_0000

31	30	29	28	27	26	25	24
KEY1							
23	22	21	20	19	18	17	16
KEY1							
15	14	13	12	11	10	9	8
KEY1							
7	6	5	4	3	2	1	0
KEY1							

Bits	Description	
[31:0]	KEY1	**Cipher Key1 Register** This is the KEY1 data for cipher function. **Note 1:** If there is not any KEY1(SPIM_KEY1[31:0]) or KEY2(SPIM_KEY2[31:0]) (KEY1 is 0x0000_0000 or KEY2 is 0x0000_0000), the cipher function will be disabled automatically. **Note 2:** When CIPHOFF(SPIM_CTL0[0]) is 0, both of KEY1(SPIM_KEY1[31:0]) and KEY2(SPIM_KEY2[31:0]) do not equal to 0x0000_0000 (i.e. KEY1≠ 0x0000_0000 and KEY2≠0x0000_0000), cipher encryption/decryption is enabled.

Cipher Key2 Register (SPIM_KEY2)

Register	Offset	R/W	Description	Reset Value
SPIM_KEY2	SPIM_BA+0x40	W	Cipher Key2 Register	0x0000_0000

31	30	29	28	27	26	25	24
KEY2							
23	22	21	20	19	18	17	16
KEY2							
15	14	13	12	11	10	9	8
KEY2							
7	6	5	4	3	2	1	0
KEY2							

Bits	Description	
[31:0]	KEY2	**Cipher Key2 Register** This is the KEY2 data for cipher function. **Note 1:** If there is not any KEY1(SPIM_KEY1[31:0]) or KEY2(SPIM_KEY2[31:0]) (KEY1 is 0x0000_0000 or KEY2 is 0x0000_0000), the cipher function will be disabled automatically. **Note 2:** When CIPHOFF(SPIM_CTL0[0]) is 0, both of KEY1(SPIM_KEY1[31:0]) and KEY2(SPIM_KEY2[31:0]) do not equal to 0x0000_0000 (i.e. KEY1≠ 0x0000_0000 and KEY2≠0x0000_0000), cipher encryption/decryption is enabled.

Direct Memory Mapping Mode Control Register (SPIM_DMMCTL)

Register	Offset	R/W	Description	Reset Value
SPIM_DMMCTL	SPIM_BA+0x44	R/W	Direct Memory Mapping Mode Control Register	0x0008_0000

31	30	29	28	27	26	25	24
ACTSCLKT				Reserved	UACTSCLK	CREN	BWEN
23	22	21	20	19	18	17	16
Reserved			DESELTIM				
15	14	13	12	11	10	9	8
CRMDAT							
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:28]	**ACTSCLKT**	**SPI Flash Active SCLK Time for Direct Memory Mapping Mode, DMA Write Mode, and DMA Read Mode Only** The bits set time interval between SPIM SS active edge and the position edge of the first serial SPI output clock, as shown in Figure 6.29-3. (1) ACTSCLKT = 0 (function disable):. Time interval = 1 AHB clock cycle time. (2) ACTSCLKT ≠ 0 (function enable): Time interval = (ACTSCLKT + 3) * AHB clock cycle time. **Note 1:** AHB clock cycle time = 1/AHB clock frequency. **Note 2:** SCLK is SPI output clock **Note 3:** Please check the used SPI Flash specification to know the setting value of this register, and different SPI Flash vendor may use different setting values.
[27]	**Reserved**	Reserved.
[26]	**UACTSCLK**	**User Sets SPI Flash Active SCLK Time for Direct Memory Mapping Mode, DMA Write Mode, and DMA Read Mode Only** 0 = According to DIVIDER(SPIM_CTL1[31:16]), ACTSCLKT(SPIM_DMMCTL[31:28]) is set by hardware automatically. (Default) 1 = Set ACTSCLKT(SPIM_DMMCTL[31:28]) by user manually. **Note:** When user wants to set ACTSCLKT(SPIM_DMMCTL[31:28]) manually, please set UACTSCLK to 1.
[25]	**CREN**	**Continuous Read Mode Enable Bit for Direct Memory Mapping Mode, Read Command Codes 0xBB, 0xEB, 0xE7, 0x0D, 0xBD, and 0xED Only** 0 = Continuous Read Mode Disabled. (Default) 1 = Continuous Read Mode Enabled. For read operations of SPI Flash, commands of fast read quad I/O (0xEB), word read quad I/O (0xE7), fast read dual I/O (0xBB), DTR fast read (0x0D), DTR fast read dual I/O (0xBD), and DTR fast read quad I/O (0xED) can further reduce command overhead through setting the “continuous read mode” bits (8 bits) after the input address data. **Note:** When using function of continuous read mode and setting USETEN (SPIM_CTL2[16]) to 1, CRMDAT(SPIM_DMMCTL[15:8]) must be set by used SPI Flash specifications. When using function of continuous read mode and setting USETEN(SPIM_CTL2[16]) to 0, CRMDAT(SPIM_DMMCTL[15:8]) is set to value 0x20.

[24]	**BWEN**	**16 Bytes Burst Wrap Mode Enable Bit for Direct Memory Mapping Mode, Cache Enable, and Read Command Code 0xEB, and 0xE7 Only** 0 = Burst Wrap Mode Disabled. (Default) 1 = Burst Wrap Mode Enabled. In direct memory mapping mode, both of quad read commands “0xEB” and “0xE7” support burst wrap mode for cache application and performance enhance. For cache application, the burst wrap mode can be used to fill the cache line quickly (In this SPI Flash controller, use cache data line with 16 bytes size). For performance enhance with direct memory mapping mode and cache enable, when cache data is miss, the burst wrap mode can let MCU get the required SPI Flash data quickly. **Note:** In Direct Memory Mapping Mode, the cache function needs to be enabled when 16 bytes burst wrap mode is enabled.
[23:21]	**Reserved**	Reserved.
[20:16]	**DESELTIM**	**SPI Flash Deselect Time for Direct Memory Mapping Mode Only** Set the minimum time width of SPI Flash deselect time (i.e. Minimum SPIM_SS deselect time), as shown in Figure 6.29-3. (1) Cache function disable: Minimum time width of SPIM_SS deselect time = (DESELTIM + 1) * AHB clock cycle time. (2) Cache function enable: Minimum time width of SPIM_SS deselect time = (DESELTIM + 4) * AHB clock cycle time. **Note 1:** AHB clock cycle time = 1/AHB clock frequency. **Note 2:** When cipher encryption/decryption is enabled, please set this register value >= 0x10. When cipher encryption/decryption is disabled, please set this register value >= 0x8. **Note 3:** Please check the used SPI Flash specification to know the setting value of this register, and different SPI Flash vendor may use different setting values.
[15:8]	**CRMDAT**	**Mode Bits Data for Continuous Read Mode for Direct Memory Mapping Mode Only** Set the mode bits data for continuous read mode (or performance enhance mode). When setting this mode bits currently **(Note1)** and set CREN(SPIM_DMMCTL[25]), this reduces the command phase by eight clocks and allows the read address to be immediately entered after SPIM_SS asserted to active. (Note1) **Note 1:** Please check the used SPI Flash specification to know the setting value of this mode bits data, and different SPI Flash vendor may use different setting values. **Note 2:** CRMDAT needs to be used with CREN(SPIM_DMMCTL[25]).
[7:0]	**Reserved**	Reserved.

Control Register 2 (SPIM_CTL2)

Register	Offset	R/W	Description	Reset Value
SPIM_CTL2	SPIM_BA+0x48	R/W	Control Register 2	0x0804_0000

31	30	29	28	27	26	25	24
Reserved			DCNUM				
23	22	21	20	19	18	17	16
Reserved			DTRMPOFF	Reserved			USETEN
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**DCNUM**	**Dummy Cycle Number for Direct Memory Mapping Mode and DMA Read Mode Only** Set number of dummy cycles **(1) For non-DTR command codes 0x03, 0x0B, 0x3B, 0xBB, 0xEB, and 0xE7:** When read command code do not need any dummy cycles (i.e. dummy cycle number = 0x0), user must set DCNUM to 0x0. For command code 0xBB, if both mode cycle number (or performance enhance cycle number) and dummy cycle number do not equal to 0x0 simultaneously, user must set DCNUM to “mode cycle number + dummy cycle number” by used SPI Flash specification. For command code 0xBB, if there is only dummy cycle number (i.e. dummy cycle number ≠ 0x0 and mode cycle number = 0x0 (or performance enhance cycle number = 0x0)), user set DCNUM to dummy cycle number by used SPI Flash specification. For command codes 0x0B, 0x3B, 0xEB, and 0xE7, user only set DCNUM to dummy cycle number by used SPI Flash specification. **(2) For DTR command codes 0x0D, 0xBD, and 0xED:** user sets DCNUM to dummy cycle number and DTRMPOFF(SPIM_CTL2[20]) by used SPI Flash specification. **Note:** Number of dummy cycles depends on the frequency of SPI output clock, SPI Flash vendor, and read command types. Please check the used SPI Flash specification to know the setting value of this number of dummy cycles.
[23:21]	**Reserved**	Reserved.
[20]	**DTRMPOFF**	**Mode Phase OFF for DTR Command Codes 0x0D, 0xBD, and 0xED for Direct Memory Mapping Mode and DMA Read Mode Only** 0 = Mode cycle number (or performance enhance cycle number) does not equal to 0x0 in DTR read command codes 0x0D, 0xBD, and 0xED. 1 = mode cycle number (or performance enhance cycle number) equals to 0x0 in DTR read command codes 0x0D, 0xBD, and 0xED. **Note:** Please check the used SPI Flash specification to know the mode cycle number (or performance enhance cycle number) for DTR command codes 0x0D, 0xBD, and 0xED.
[19:17]	**Reserved**	Reserved.
[16]	**USETEN**	**User Set Value Enable Bit for Direct Memory Mapping Mode and DMA Read Mode with Read Commands 0x03,0x0B,0x3B,0xBB,0xEB, and 0xE7 Only**

		0 = Hardware circuit of SPI Flash controller will use the following default values of DCNUM(SPIM_CTL2[28:24]) and CRMDAT(SPIM_DMMCTL[15:8]) to configure SPI Flash operations automatically. **Dummy cycle number (DCNUM):** Dummy cycle number for read command 0x03: 0x0 Dummy cycle number for read command 0x0B: 0x8 Dummy cycle number for read command 0x3B: 0x8 Dummy cycle number for read command 0xBB: 0x0 Dummy cycle number for read command 0xEB: 0x4 Dummy cycle number for read command 0xE7: 0x2 Mode bits data for continuous read mode (CRMDAT): 0x20 1 = If DCNUM(SPIM_CTL2[28:24]) and CRMDAT(SPIM_DMMCTL[15:8]) are not set as above default values, user must set USETEN to 0x1, DCNUM(SPIM_CTL2[28:24]) and CRMDAT(SPIM_DMMCTL[15:8]) to configure SPI Flash operations manually. For DTR command codes 0x0D, 0xBD, and 0xED, please set USETEN to 0x1.
[15:0]	**Reserved**	Reserved.

6.30 I²C Serial Interface Controller (I²C)

6.30.1 Overview

I²C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The I²C standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously.

There are five sets of I²C controllers which support Power-down wake-up function.

6.30.2 Features

The I²C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the I²C bus include:

- Supports up to five I²C ports
- Master/Slave mode
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Supports Standard mode (100 kbps), Fast mode (400 kbps) and Fast mode plus (1 Mbps)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization used as a handshake mechanism to suspend and resume serial transfer
- Built-in 14-bit time-out counter requesting the I²C interrupt if the I²C bus hangs up and timer-out counter overflow
- Programmable clocks allow for versatile rate control
- Supports 7-bit addressing and 10-bit addressing mode
- Supports multiple address recognition (four slave address with mask option)
- Supports Power-down wake-up function
- Supports PDMA with one buffer capability (Only I²C0/1 with PDMA function)
- Supports two-level buffer function
- Supports setup/hold time programmable
- Supports Bus Management (SM/PM compatible) function compatible with the SMBUS specification rev 2.0 (http://smbus.org/specs/) and PMBUS specification rev 1.2 (http://pmbus.org/)

6.30.3 Block Diagram

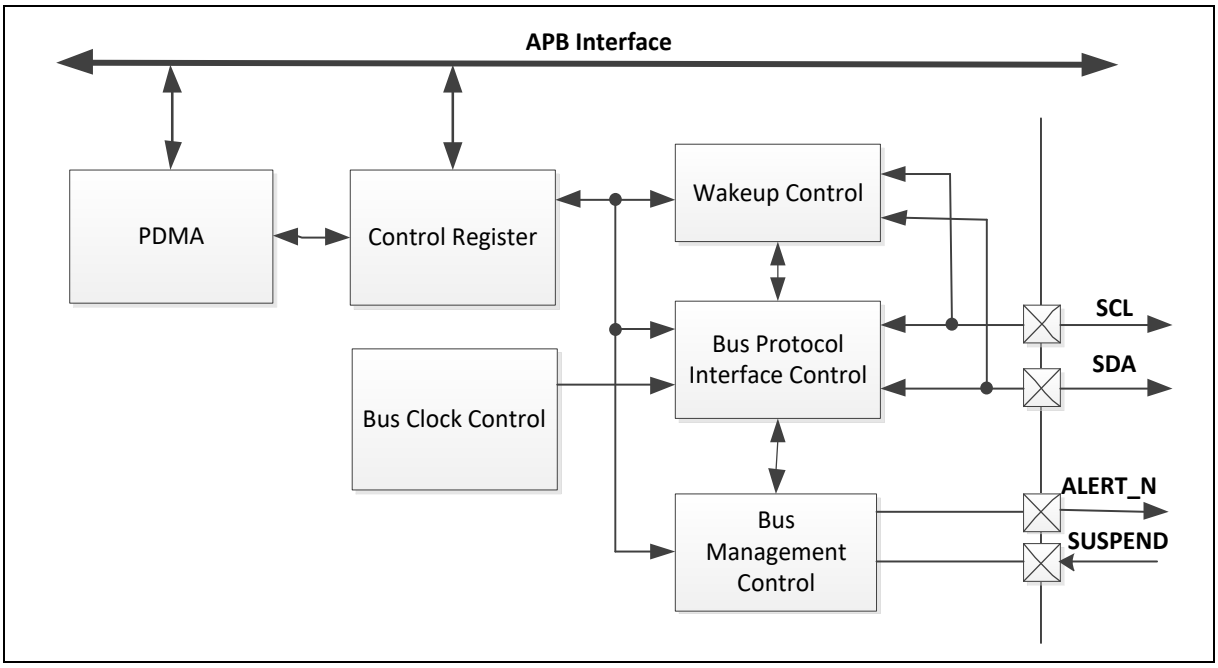


Figure 6.30-1 I²C Controller Block Diagram

6.30.4 Basic Configuration

6.30.4.1 I2C0 Basic Configuration

- Clock Source Configuration
  - Enable I2C0 peripheral clock in I2C0CKEN (CLK_APBCLK0[8]).
- Reset Configuration
  - Reset I2C0 controller in I2C0RST (SYS_IPRST1[8]).
- SCL/SDA Pin Switch
  - Set SWITCHEN (I2C0_CTL1[10] = 1) to switch I2C0 SCL/SDA pin configuration.

6.30.4.2 I2C1 Basic Configuration

- Clock Source Configuration
  - Enable I2C1 peripheral clock in I2C1CKEN (CLK_APBCLK0[9]).
- Reset Configuration
  - Reset I2C1 controller in I2C1RST (SYS_IPRST1[9]).
- SCL/SDA Pin Switch
  - Set SWITCHEN (I2C1_CTL1[10] = 1) to switch I2C1 SCL/SDA pin configuration.

6.30.4.3 I2C2 Basic Configuration

- Clock Source Configuration
  - Enable I2C2 peripheral clock in I2C2CKEN (CLK_APBCLK0[10]).
- Reset Configuration
  - Reset I2C2 controller in I2C2RST (SYS_IPRST1[10]).

- SCL/SDA Pin Switch
  - Set SWITCHEN (I2C2_CTL1[10] = 1) to switch I2C2 SCL/SDA pin configuration.

6.30.4.4 I2C3 Basic Configuration

- Clock Source Configuration
  - Enable I2C3 peripheral clock in I2C3CKEN (CLK_APBCLK0[11]).
- Reset Configuration
  - Reset I2C3 controller in I2C3RST (SYS_IPRST1[11]).
- SCL/SDA Pin Switch
  - Set SWITCHEN (I2C3_CTL1[10] = 1) to switch I2C3 SCL/SDA pin configuration.

6.30.4.5 I2C4 Basic Configuration

- Clock Source Configuration
  - Enable I2C4 peripheral clock in I2C4CKEN (CLK_APBCLK1[3]).
- Reset Configuration
  - Reset I2C4 controller in I2C4RST (SYS_IPRST2[3]).
- SCL/SDA Pin Switch
  - Set SWITCHEN (I2C4_CTL1[10] = 1) to switch I2C4 SCL/SDA pin configuration.

6.30.5 Functional Description

On I²C bus, data is transferred between a Master and a Slave. Data bits transferred on the SCL and SDA lines are synchronously on a byte-by-byte basis. Each data byte is 8-bit long. There is one SCL clock pulse for each data bit with the MSB being transmitted first, and an acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP). Please refer to Figure 6.30-2 for more detailed I²C BUS Timing.

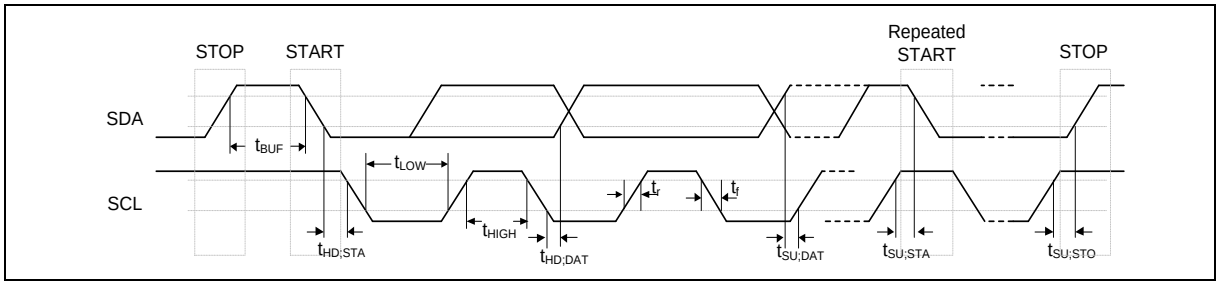


Figure 6.30-2 I²C Bus Timing

The device's on-chip I²C provides the serial interface that meets the I²C bus standard mode specification. The I²C port handles byte transfers autonomously. To enable this port, the bit I2CEN in I2C_CTL0 should be set to '1'. The I²C hardware interfaces to the I²C bus via two pins: SDA and SCL. When I/O pins are used as I²C ports, user must set the pins function to I²C in advance.

There is two-level buffer to improve the performance of I²C bus. In Two-level BUFFER Mode, the next transmitted or the last received data can be active even if the current data is transmitted or the last received isn't read back yet.

Note: Pull-up resistor is needed for I²C operation as the SDA and SCL are open-drain pins.

6.30.5.1 I²C Protocol

Figure 6.30-3 shows the typical I²C protocol. Normally, a standard communication consists of four parts:

- START or Repeated START signal generation
- Slave address and R/W bit transfer
- Data transfer
- STOP signal generation

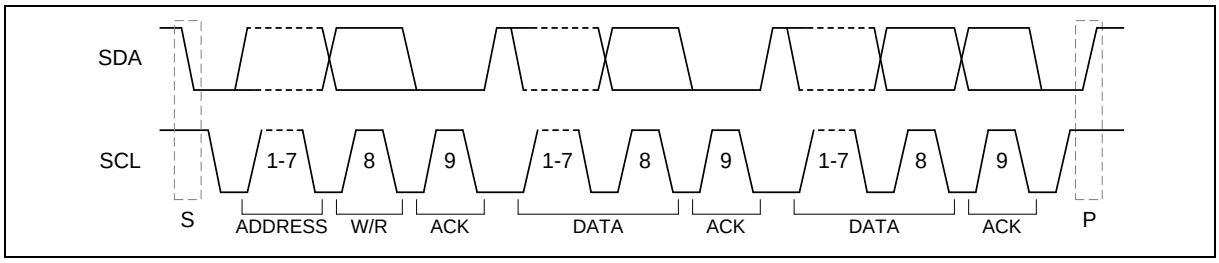


Figure 6.30-3 I²C Protocol

- START or Repeated START signal

  When the bus is free/idle, which means no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the “S” bit, is defined as a HIGH to LOW transition on the SDA line while SCL is HIGH. The START signal denotes the beginning of a new data transmission.

  After having sent the address byte (address and read/write bit), the master may send any number of bytes followed by a stop condition. Instead of sending the stop condition it is also allowed to send another start condition again followed by an address (and of course including a read/write bit) and more data. The start condition is called as Repeat START (Sr). This is defined recursively allowing any number of start conditions to be sent. The purpose of this is to allow combined write/read operations to one or more devices without releasing the bus and thus with the guarantee that the operation is not interrupted. The controller uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus.

- STOP signal

  The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the “P” bit, is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH.

  Figure 6.30-4 shows the waveform of START, Repeat START and STOP.

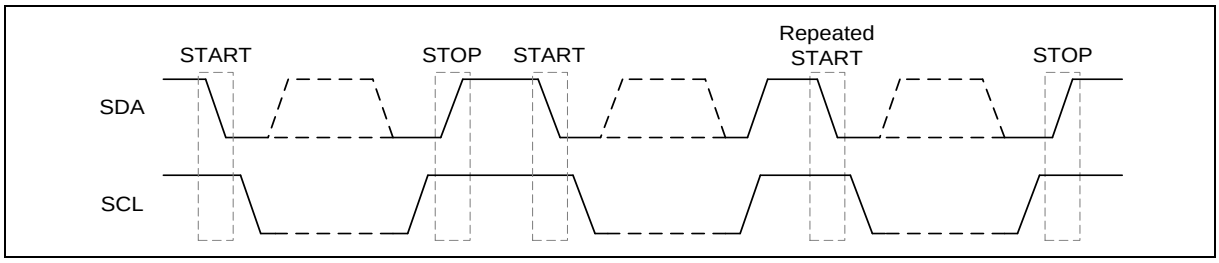


Figure 6.30-4 START and STOP Conditions

- Slave Address Transfer

After a (Repeated) START condition, the master sends a slave address to identify the target device of the communication. The start address can comprise one or two address bytes (for 7-bit or for 10-bit addressing schemes). After an address byte, a slave sensitive to the transmitted address has to acknowledge the reception.

Therefore, the slave's address can be programmed in the device, where it is compared to the received address. In case of a match, the slave answers with an acknowledge (SDA = 0). Slaves that are not targeted answer with a non-acknowledge (SDA = 1). In addition to the match of the programmed address, another address byte value has to be answered with an acknowledge if the slave is capable to handle the corresponding requests.

- Data Transfer

  When a slave receives a correct address with an R/W bit, the data will follow R/W bit specified to transfer. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a Not Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

  If the master, as a receiving device, does Not Acknowledge (NACK) the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal. The Figure 6.30-5 and Figure 6.30-6 shows the waveform of bit transfer and acknowledge.

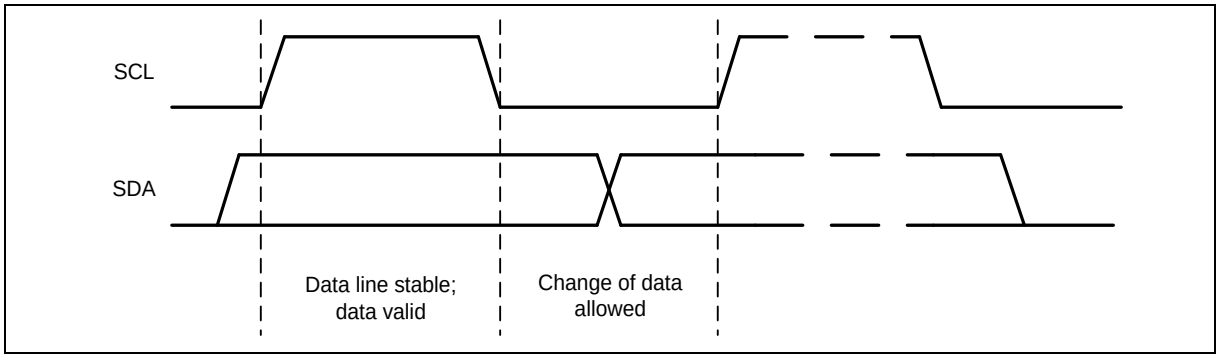


Figure 6.30-5 Bit Transfer on the I²C Bus

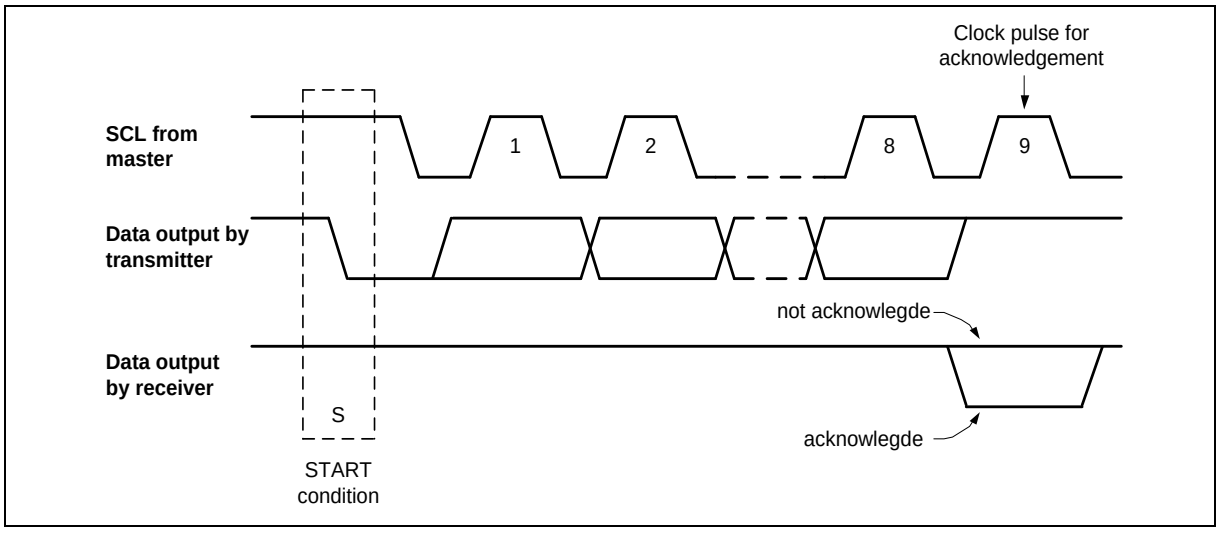


Figure 6.30-6 Acknowledge on the I²C Bus

- Data transfer on I²C bus

  Figure 6.30-7 shows a master transmits data to slave by 7-bit. A master addresses a slave

with a 7-bit address and 1-bit write index to denote that the master wants to transmit data to the slave. The master keeps transmitting data after the slave returns acknowledge to the master.

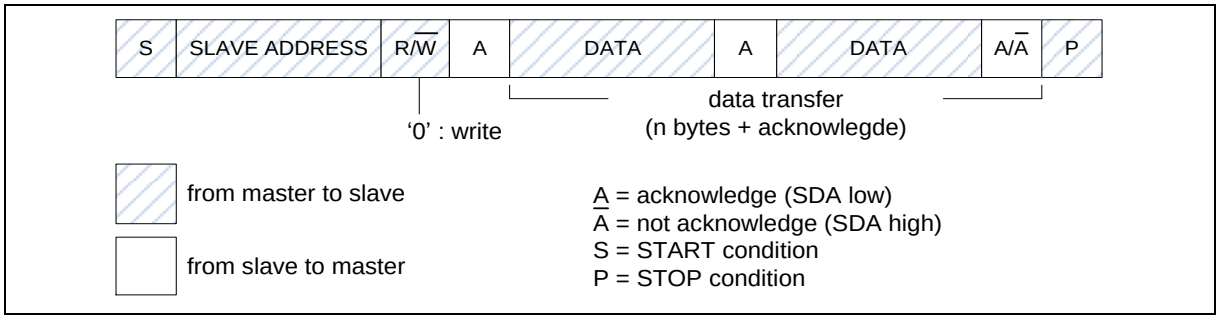


Figure 6.30-7 Master Transmits Data to Slave by 7-bit

Figure 6.30-8 shows a master read data from slave by 7-bit. A master addresses a slave with a 7-bit address and 1-bit read index to denote that the master wants to read data from the slave. The slave will start transmitting data after the slave returns acknowledge to the master.

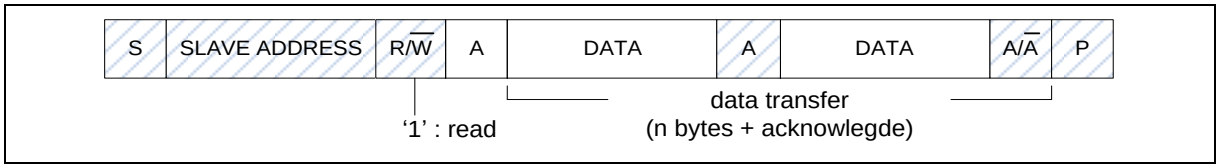


Figure 6.30-8 Master Reads Data from Slave by 7-bit

Figure 6.30-9 shows a master transmits data to slave by 10-bit. A master addresses a slave with a 10-bit address. First byte contains 10-bit address indicator (5'b11110) and 2-bit address with write index, second byte contains 8-bit address. The master keeps transmitting data after the second byte end. Note that 7-bit and 10-bit address device can work on the same bus.

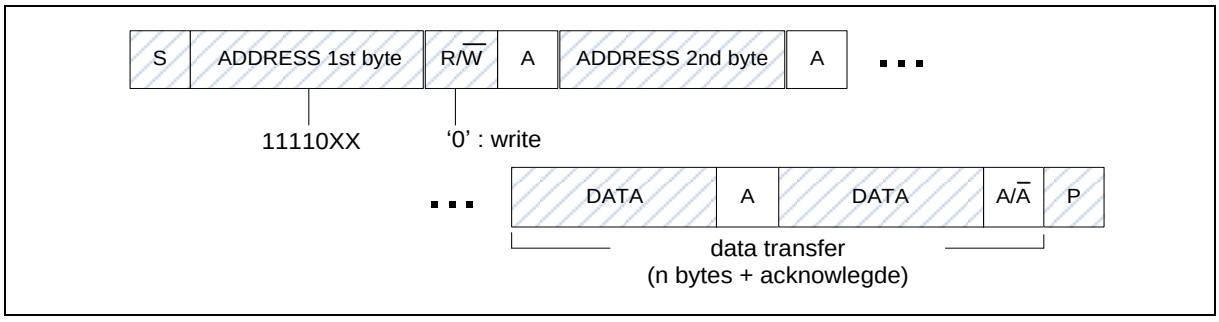


Figure 6.30-9 Master Transmits Data to Slave by 10-bit

Figure 6.30-10 shows a master read data from slave by 10-bit. A master addresses a slave with a 10-bit address. First mater transmits 10-bit address to slave, after that master transmits first byte with read index. The slave will start transmitting data after the first byte with read index.

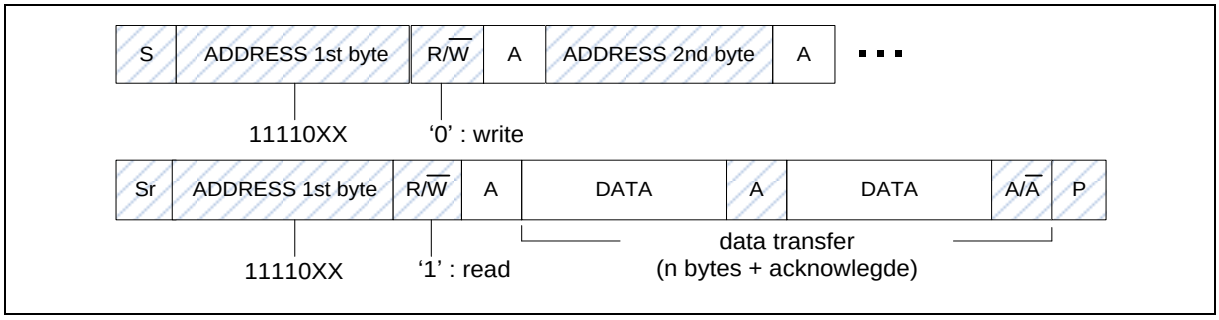


Figure 6.30-10 Master Reads Data from Slave by 10-bit

6.30.5.2 Operation Modes

The on-chip I²C ports support three operation modes, Master, Slave, and General Call Mode.

In a given application, I²C port may operate as a master or as a slave. In Slave mode, the I²C port hardware looks for its own slave address and the general call address. If one of these addresses is detected, and if the slave is willing to receive or transmit data from/to master(by setting the AA bit), acknowledge pulse will be transmitted out on the 9th clock, hence an interrupt is requested on both master and slave devices if interrupt is enabled. When the microcontroller wishes to become the bus master, hardware waits until the bus is free before entering Master mode so that a possible slave action is not be interrupted. If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer.

To control the I²C bus transfer in each mode, user needs to set I2C_CTL0, I2C_DAT registers according to current status code of I2C_STATUS0 register. In other words, for each I²C bus action, user needs to check current status by I2C_STATUS0 register, and then set I2C_CTL0, I2C_DAT registers to take bus action. Finally, check the response status by I2C_STATUS0.

The bits, STA, STO and AA in I2C_CTL0 register are used to control the next state of the I²C hardware after SI flag of I2C_CTL0 [3] register is cleared. Upon completion of the new action, a new status code will be updated in I2C_STATUS0 register and the SI flag of I2C_CTL0 register will be set. But the SI flag will not be set when I²C STOP. If the I²C interrupt control bit INTEN (I2C_CTL0 [7]) is set, appropriate action or software branch of the new status code can be performed in the Interrupt service routine.

Figure 6.30-11 shows the current I²C status code is 0x08, and then set I2C_DATA=SLA+W and (STA,STO,SI,AA) = (0,0,1,x) to send the address to I²C bus. If a slave on the bus matches the address and response ACK, the I2C_STATUS0 will be updated by status code 0x18.

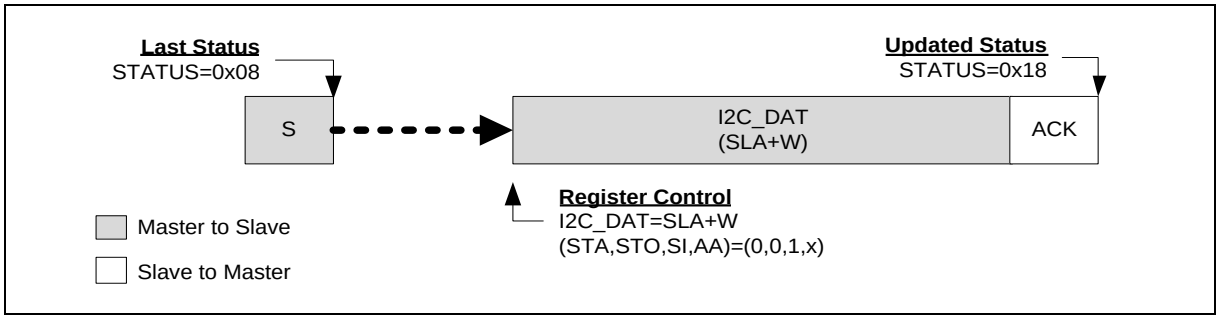


Figure 6.30-11 Control I²C Bus according to the Current I²C Status

Master Mode

In Figure 6.30-12 and Figure 6.30-13, all possible protocols for I²C master are shown. User needs to follow proper path of the flow to implement required I²C protocol.

In other words, user can send a START signal to bus and I²C will be in Master Transmitter (MT) mode (Figure 6.30-12) or Master receiver (MR) mode (Figure 6.30-13) after START signal has been sent

successfully and new status code would be 0x08. Followed by START signal, user can send slave address, read/write bit, data and Repeated START, STOP to perform I²C protocol.

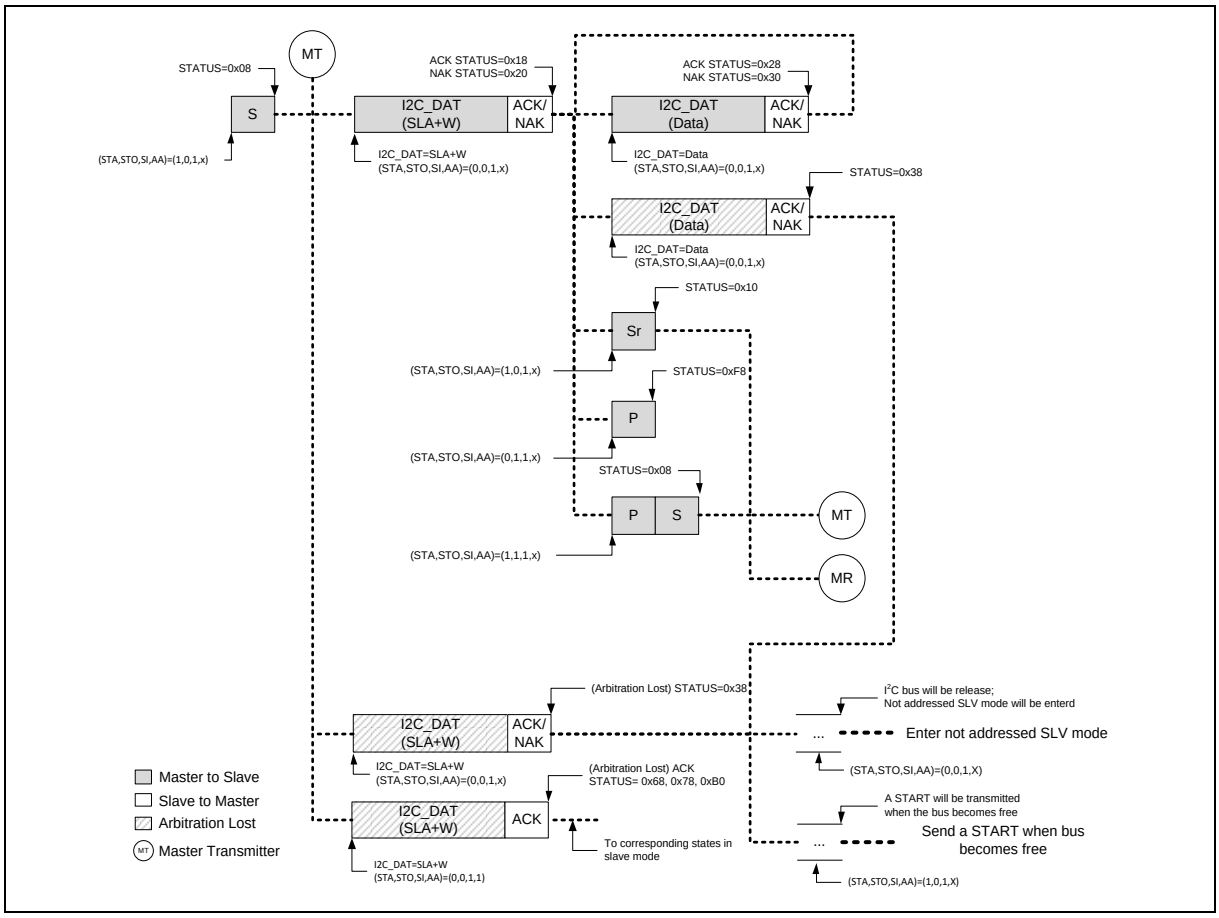


Figure 6.30-12 Master Transmitter Mode Control Flow

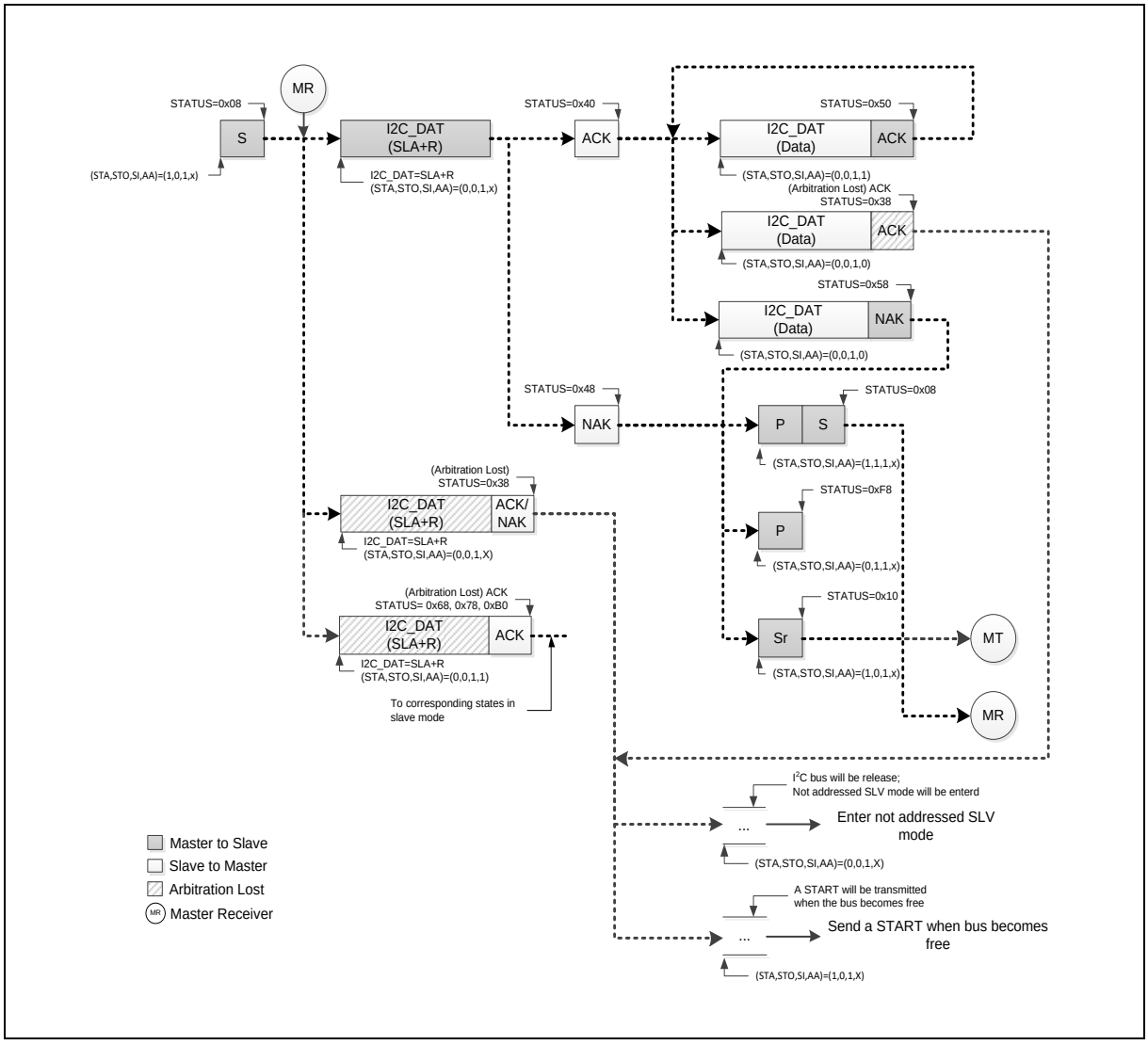


Figure 6.30-13 Master Receiver Mode Control Flow

If the I²C is in Master mode and gets arbitration lost, the status code will be 0x38. In status 0x38, user may set (STA, STO, SI, AA) = (1, 0, 1, X) to send START to re-start Master operation when bus become free. Otherwise, user may set (STA, STO, SI, AA) = (0, 0, 1, X) to release I²C bus and enter not addressed Slave mode.

Slave Mode

When reset default, I²C is not addressed and will not recognize the address on I²C bus. User can set slave address by I2C_ADDRn (n=0~3) and set (STA, STO, SI, AA) = (0, 0, 1, 1) to let I²C recognize the address sent by master. Figure 6.30-14 shows all the possible flow for I²C in Slave mode. Users need to follow a proper flow (as shown in Figure 6.30-14 to implement their own I²C protocol.

If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer. If the detected address is SLA+W (Master want to write data to Slave) after arbitration lost, the status code is 0x68. If the detected address is SLA+R (Master want to read data from Slave) after arbitration lost, the status code is 0xB0.

Note: During I²C communication, the SCL clock will be released when writing '1' to clear SI flag in Slave mode.

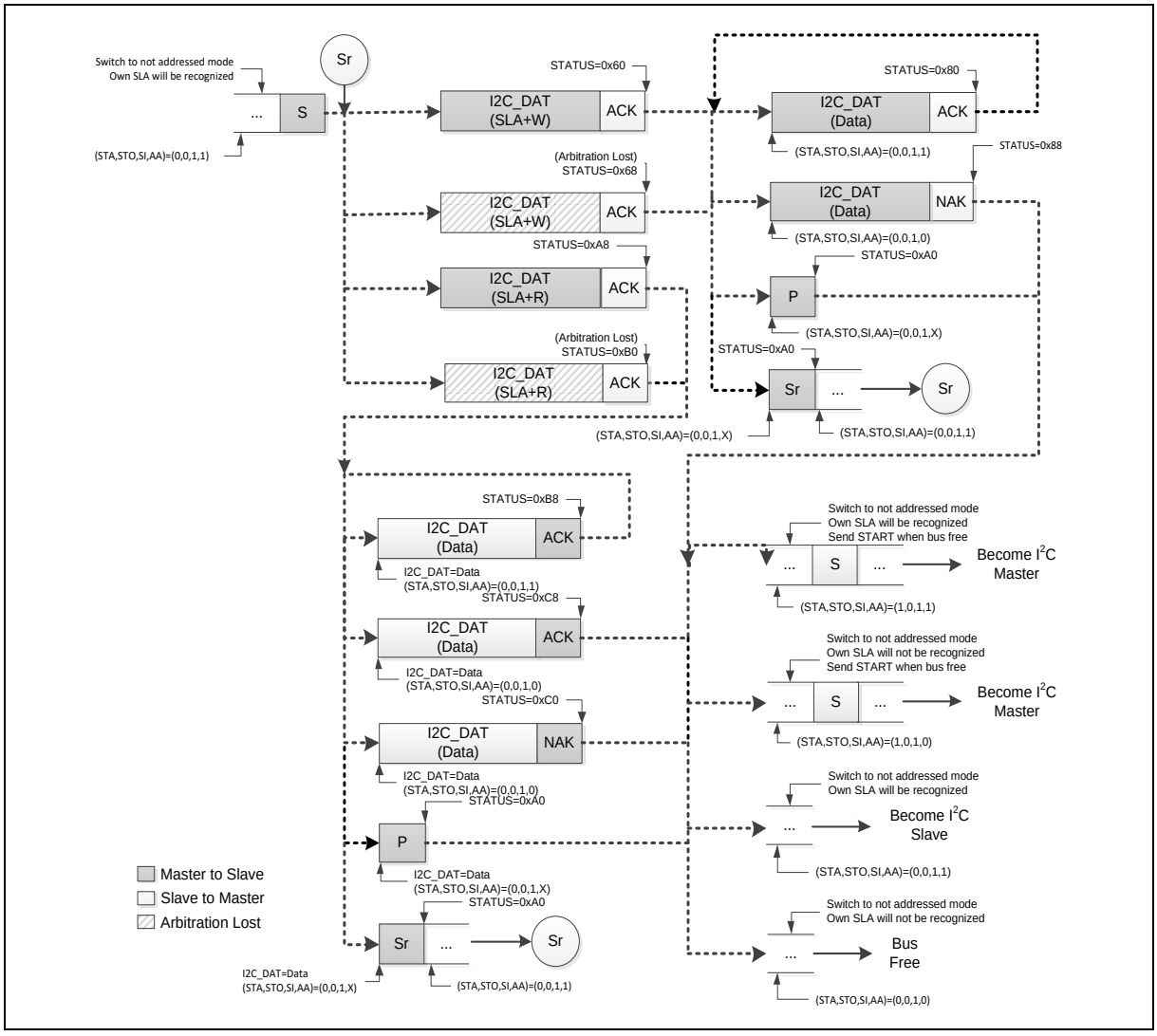


Figure 6.30-14 Slave Mode Control Flow

If I²C is still receiving data in addressed Slave mode but got a STOP or Repeated START, the status code will be 0xA0. User could follow the action for status code 0x88 as shown in the above figure when getting 0xA0 status.

If I²C is still transmitting data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0xC8 as shown in the above figure when getting 0xA0 status.

Note: After slave gets status of 0x88, 0xC8, 0xC0 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I²C signal or address from master. At this status, I²C should enter idle mode.

General Call (GC) Mode

If the GC bit (I2C_ADDRn [0]) is set, the I²C port hardware will respond to General Call address (00H). User can clear GC bit to disable general call function. When the GC bit is set and the I²C in Slave mode, it can receive the general call address by 0x00 after master send general call address to I²C bus, then it will follow status of GC mode.

The GC mode can wake up when address matched. Note that the default address is 0x00, but user must set an address except for 0x00.

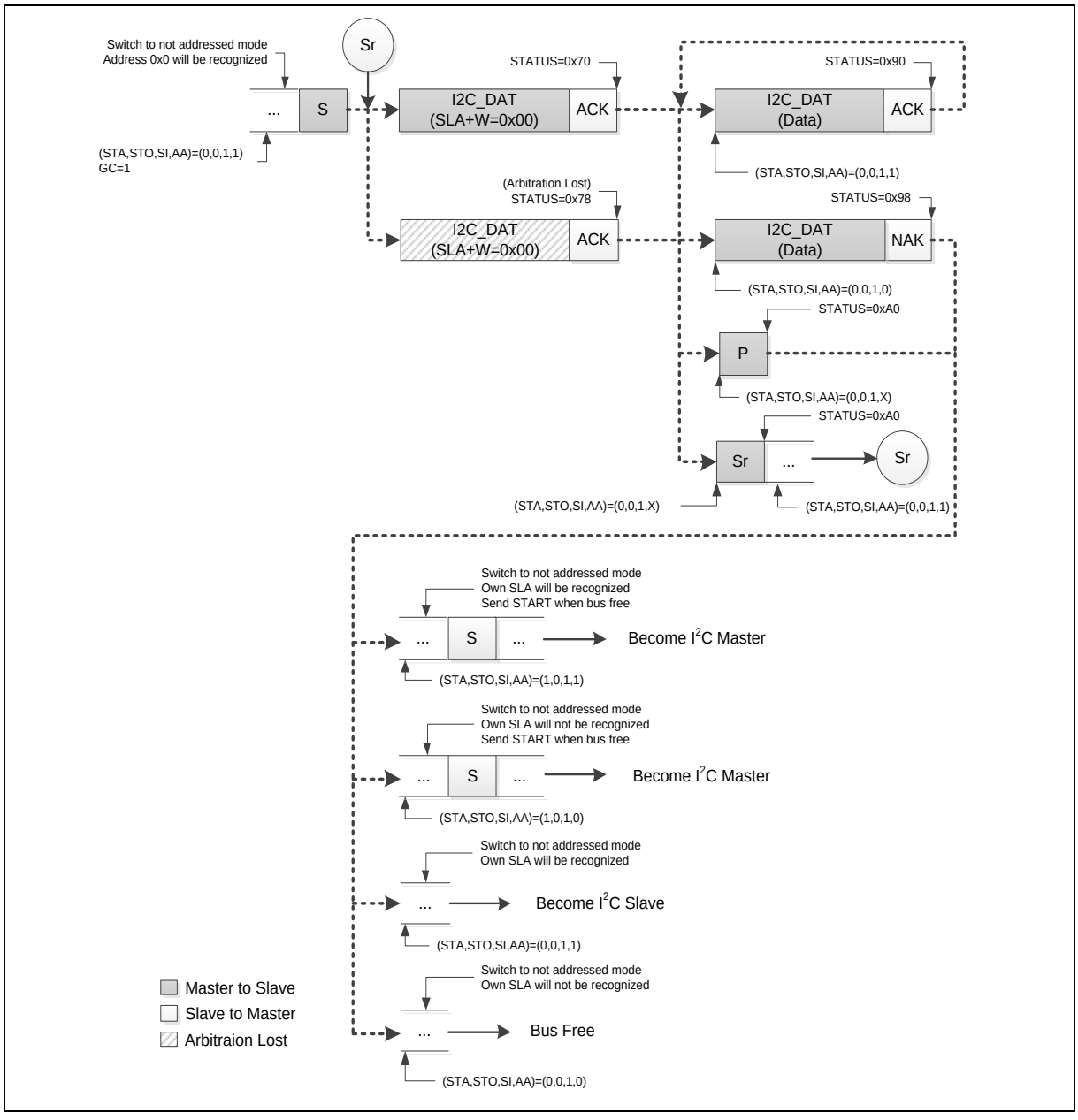


Figure 6.30-15 GC Mode

If I²C is still receiving data in GC mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0x98 in above figure when getting 0xA0 status.

Note: After slave gets status of 0x98 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I²C signal or address from master. At this time, the I²C controller should enter idle mode.

Multi-Master

In some applications, there are two or more masters on the same I²C bus to access slaves, and the masters may transmit data simultaneously. The I²C supports multi-master by including collision detection and arbitration to prevent data corruption.

If for some reason two masters initiate command at the same time, the arbitration procedure determines which master wins and can continue with the command. Arbitration is performed on the SDA signal while the SCL signal is high. Each master checks if the SDA signal on the bus corresponds to the generated SDA signal. If the SDA signal on the bus is low but it should be high, then this master has lost arbitration. The device that has lost arbitration can generate SCL pulses until the byte ends and must then release the bus and go into slave mode. The arbitration procedure can continue until all the data is transferred. This means that in multi-master system each master must monitor the bus for collisions and act accordingly.

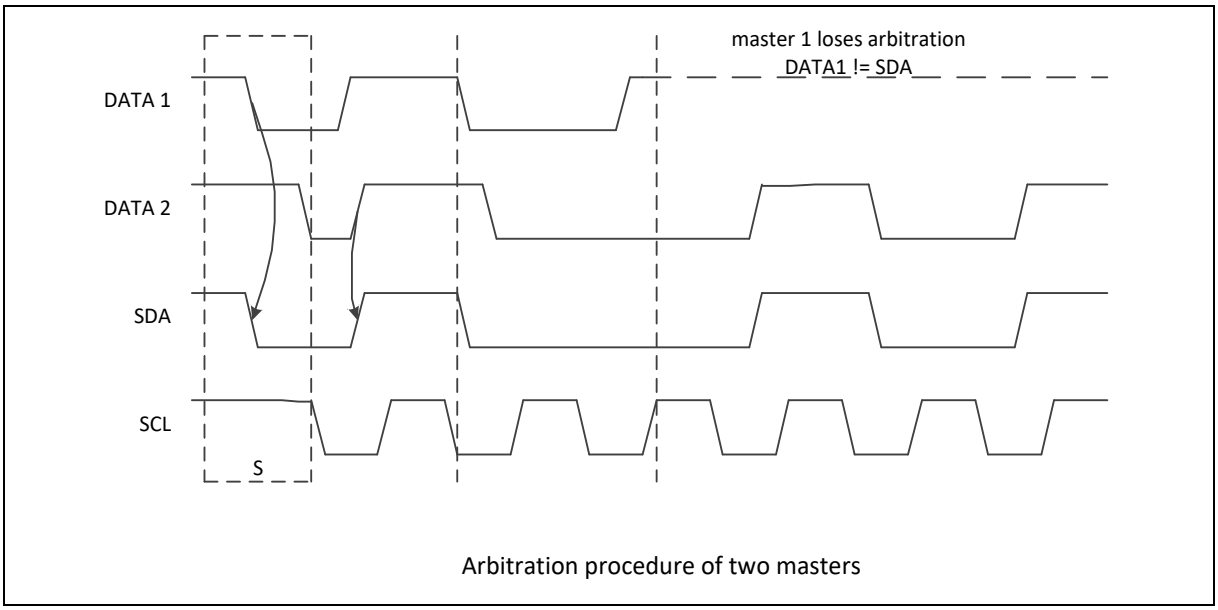


Figure 6.30-16 Arbitration Lost

- When I2C_STATUS0 = 0x38, an “Arbitration Lost” is received. Arbitration lost event maybe occur during the send START bit, data bits or STOP bit. User could set (STA, STO, SI, AA) = (1, 0, 1, X) to send START again when bus free, or set (STA, STO, SI, AA) = (0, 0, 1, X) to not addressed Slave mode. User can detect bus free by ONBUSY (I2C_STATUS1 [8]).
- When I2C_STATUS0 = 0x00, a “Bus Error” is received. To recover I^2C bus from a bus error, STO should be set and SI should be cleared, and then STO is cleared to release bus.
  - Set (STA, STO, SI, AA) = (0, 1, 1, X) to stop current transfer
  - Set (STA, STO, SI, AA) = (0, 0, 1, X) to release bus

Bus Management (SMBus/PMBus Compatible)

This section is relevant only when Bus Management feature is supported.

Introduction

The Bus Management is an I^2C interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. The Bus Management provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification rev 2.0 (http://smbus.org/specs/) and PMBUS specification rev 1.2 (http://pmbus.org/).

The System Management Bus Specification refers to three types of devices.

- A slave is a device that receives or responds to a command.

- A master is a device that issues commands, generates the clocks and terminates the transfer.
- A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This Bus Management peripheral is based on I^2C specification Rev 2.1.

Device Identification – Slave Address

Any device that exists on the Bus Management as a slave has a unique address called the Slave Address. For reference, the following addresses are reserved and must not be used by or assign to any Bus Management device. (Refer to SMBus specification for detail information)

Slave Address Bits 7-1	R/W Bit Bit 0	Comment
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Address reserved for different bus format
0000 011	X	Reserved for future use
0000 1XX	X	Reserved for future use
0101 000	X	Reserved for ACCESS.bus host
0110 111	X	Reserved for ACCESS.bus default address
1111 0XX	X	10-bit slave addressing
1111 1XX	X	Reserved for future use
0001 000	X	SMBus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

Table 6.30-1 Reserved SMBus Address

Bus Protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are Quick Command, Send Byte, Receive Byte, Write Byte, Write Word, Read Byte, Read Word, Process Call, Block Read, Block Write and Block Write-Block Read Process Call. These protocols should be implemented by the user software. (For more details of these protocols, refer to SMBus specification ver. 2.0)

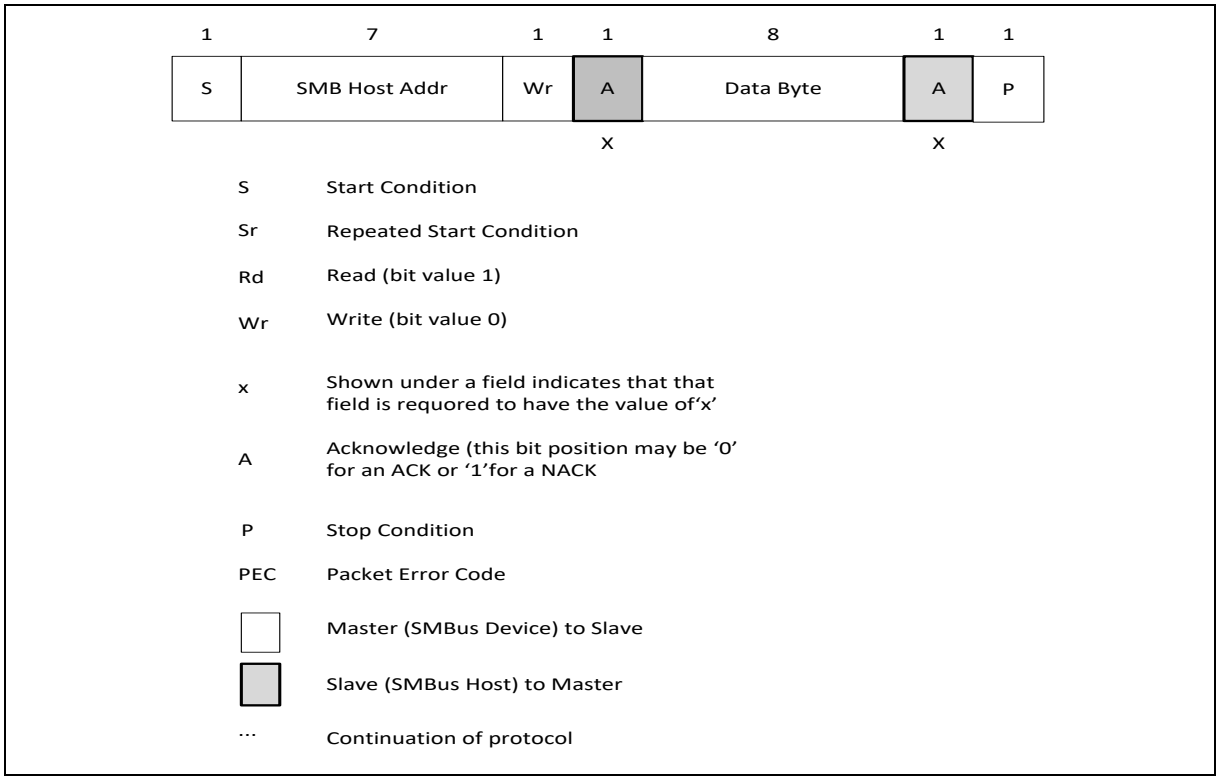


Figure 6.30-17 Bus Management Packet Protocol Diagram Element Key

Address Resolution Protocol (ARP)

Bus Management slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The Bus Management Device Default Address (0b1100 001) is enabled by setting BUSEN (I2C_BUSCTL[7]), BMDEN (I2C_BUSCTL[2]) and ALERTEN (I2C_BUSCTL[4]) bits. The ARP commands should be implemented by the user software. Arbitration is also performed in slave mode for ARP support.

Received Command and Data acknowledge control

A Bus Management receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting ACKMEN bit (I2C_BUSCTL[0]).

Host Notify Protocol

To prevent message coming to the Bus Management host controller from unknown devices in unknown formats only one method of communication is allowed, a modified form of the Write Word protocol. The standard Write Word protocol is modified by replacing the command code with the alerting device's address.

This peripheral supports the Host Notify protocol by setting the BUSEN (I2C_BUSCTL[7]), BMHEN (I2C_BUSCTL[3]) and ALERTEN (I2C_BUSCTL[4]). In this case the host will acknowledge the Bus Management Host address (0b0001000). This protocol is used when the device acts as a master and the host as a slave.

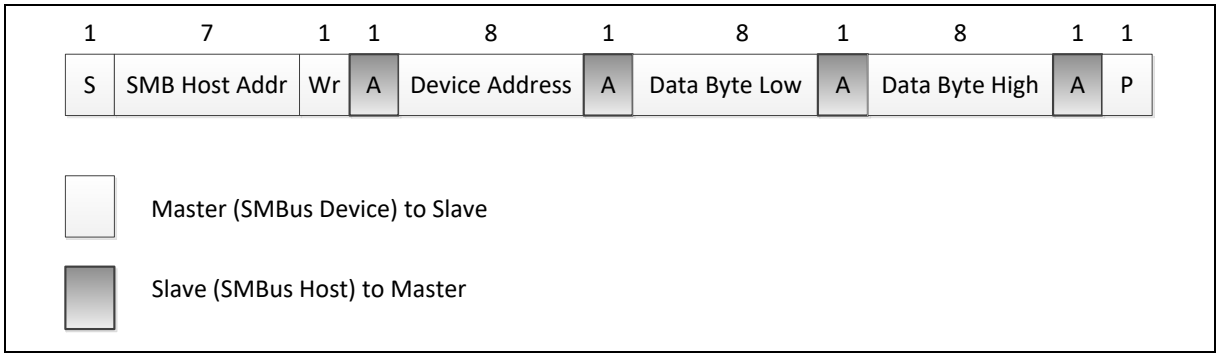


Figure 6.30-187-bit Addressable Device to Host Communication

Bus Management Alert

The Bus Management ALERT optional signal is supported. A slave-only device can signal the host through the Bus Management ALERT pin (GPA[14]/GPE[10]) that it wants to talk. The host processes the interrupt and simultaneously accesses all Bus Management ALERT pin's devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled Bus Management ALERT pin low will acknowledge the Alert Response Address.

When configured as a slave device(BMHEN=0), the Bus Management ALERT pin is pulled low by setting the ALERTEN bit (I2C_BUSCTL[4]). The Alert Response Address (ARA) is enabled at the same time.

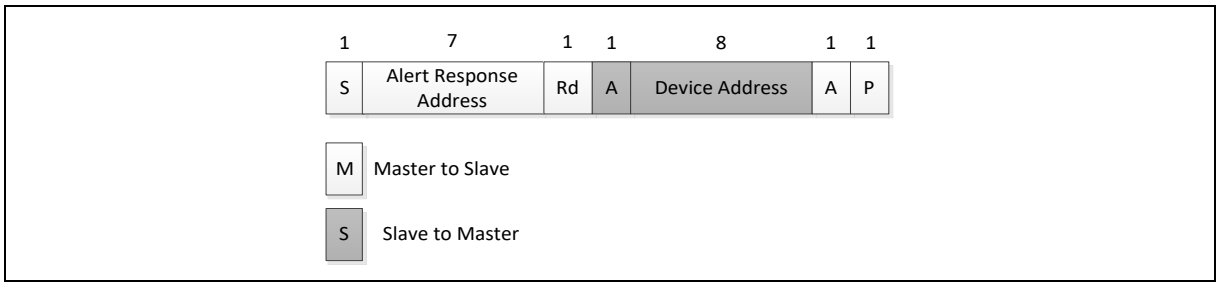


Figure 6.30-197-bit Addressable Device Responds to an ARA

When configured as a host (BMHEN=1), the ALERT flag (I2C_BUSSTS[3]) is set when a falling edge is detected on the Bus Management ALERT pin and ALERTEN=1. When ALERTEN=0, the ALERT line is considered high even if the external Bus Management ALERT pin is low. If the Bus Management ALERT pin is not needed, the Bus Management ALERT pin can be used as a standard GPIO if ALERTEN = 0;

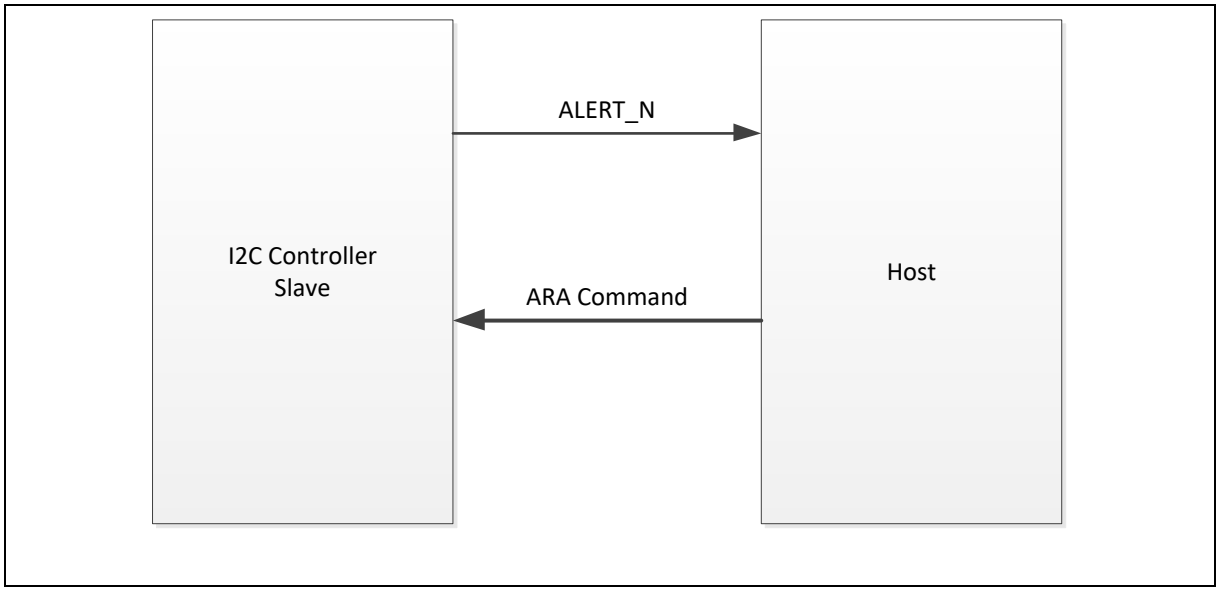


Figure 6.30-20 Bus Management ALERT Function

Packet Error Checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the C(x) = x8 + x2 + x + 1 CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator when the PECEN bit (I2C_BUSCTL[1]) is set and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC. The calculated value of PEC also can be read back on I2C_PKTCRC.

Time-out

This peripheral embeds hardware timers in order to be compliant with the 3 time-outs defined in SMBus specification ver. 2.0.

Bus Management Time-out:

The SCL low time-out condition when bus no IDLE

$T_{Time\text{-}out}$ = (BUSTO(I2C_BUSTOUT[7:0]) +1) x 16x1024 (14-bit) x T_{PCLK} (if TOCDIV4 = 0).

= (BUSTO(I2C_BUSTOUT[7:0])+1) x 16x1024 (14-bit) x 4 x T_{PCLK} (if TOCDIV4 = 1)

The bus idle condition (both SCL and SDA high) when bus IDLE

$T_{Time\text{-}out}$ = (BUSTO(I2C_BUSTOUT[7:0])+ 1) x 4 x T_{PCLK}.

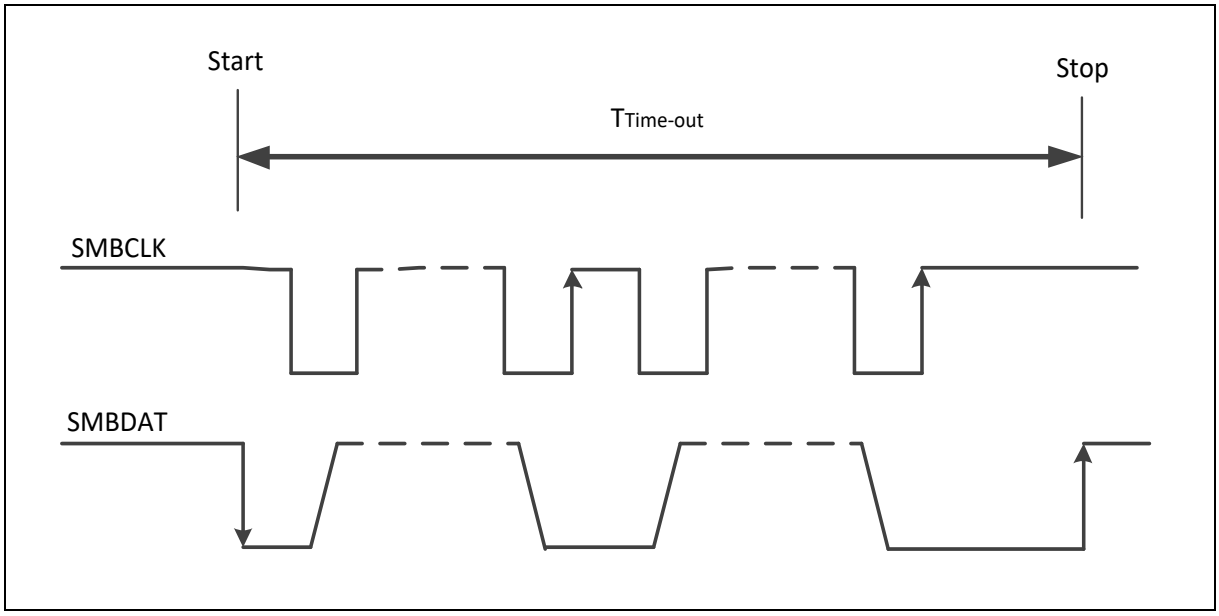


Figure 6.30-21 Bus Management Time Out Timing

Bus Clock Low Time-out:

In Master mode, the Master cumulative clock low extend time ($T_{LOW:MEXT}$) is detected

In Slave mode, the slave cumulative clock low extend time ($T_{LOW:SEXT}$) is detected

$T_{TLOW:EXT}$ = (CLKTO (I2C_CLKTOUT[7:0])+1) x 16x1024 (14-bit) x T_{PCLK} (if TOCDIV4= 0).

= (CLKTO (I2C_CLKTOUT[7:0])+1) x 16x1024 (14-bit) x 4 x T_{PCLK} (if TOCDIV4= 1)

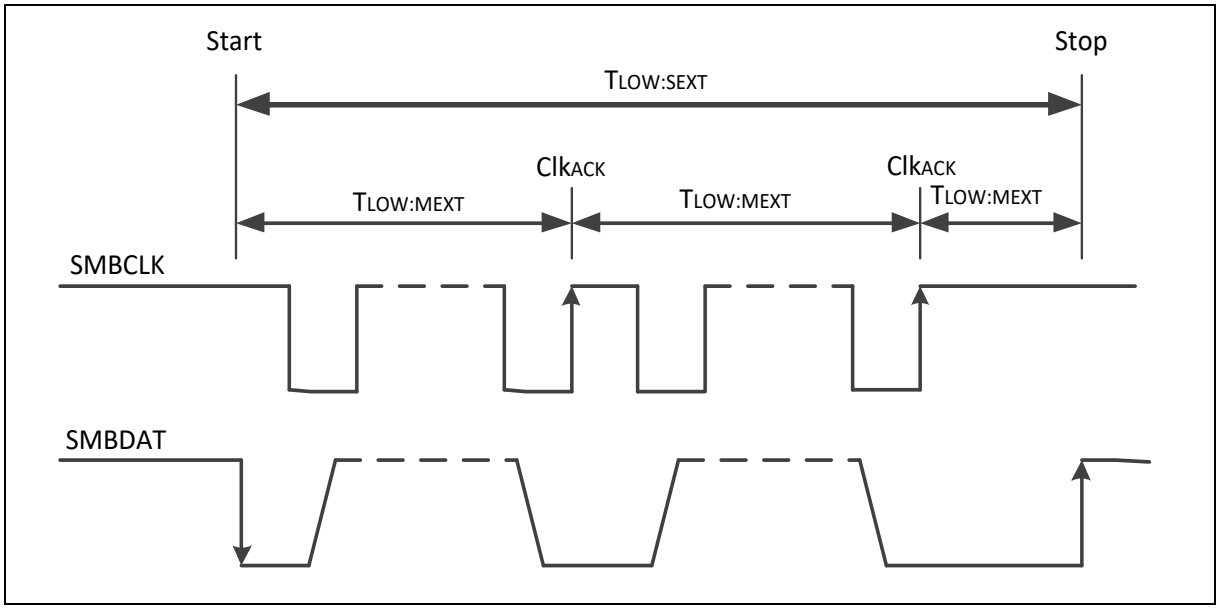


Figure 6.30-22 Bus Clock Low Time Out Timing

Bus Idle Detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for T_{IDLE} greater than $T_{HIGH,MAX}$.

This timing parameter covers the condition where a master has been dynamically added to the bus and

may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

6.30.5.3 PDMA Transfer Function

The I²C controller supports PDMA transfer function. When TXPDMAEN (I2C_CTL1 [0]) is set to 1, the I²C controller will issue request to PDMA controller to start the PDMA transmission process automatically.

When RXPDMAEN (I2C_CTL1 [1]) is set to 1, the I²C controller will start the receive PDMA process. The I²C controller will issue the request to PDMA controller automatically when there is data written into the received BUFFER.

When I²C enters PDMA mode, the mostly status interrupt will be masked. Let the interrupt not occur besides the bus error or NACK or STOP interrupt (0x20, 0x30, 0x38, 0x48, 0x58, 0x00, 0xA0, 0xC0, 0x88 and 0x98).

Set the PDMASTR (I2C_CTL1 [8]) only the I²C controller in master TX mode. If PDMASTR is cleared to 0, I²C will send STOP automatically after PDMA transfer done and buffer empty. If PDMASTR is set to 1, SI will be set to 1 and I²C bus will be stretched by hardware after PDMA transfer done and buffer empty.

Note: Only I²C0, I²C1 support PDMA transfer function in the NuMicro M463/M467 series.

6.30.5.4 Two-level BUFFER Mode on I²C bus

Set TWOBUFEN = 1 (I2C_CTL1[5]) to enable the two-level BUFFER for I²C transmitted or received buffer. It only support 7-bit slave mode and not support wakeup function. The function is used to improve transfer/receive performance.

Two-level BUFFER Mode I²C will not stretch SCL, so user must enable SRCINTEN (I2C_CTL0[13]), DPCINTEN (I2C_CTL0[12]) and set DPBITSEL (I2C_CTL0[9:8]) for avoid data transfer errors.

SARCIF (I2C_CTL0[15]) will be set 1 by hardware when I²C slave detect address read command and SRCINTEN is enable. SARCIF (I2C_CTL0[15]) is used to send interrupt in advance to let software has enough time to prepare data.

After software enable DPCINTEN(I2C_CTL0[12]) and set DPBITSEL(I2C_CTL0[9:8]), DPCIF (I2C_CTL0[14]) will be set by hardware when slave transfer/receive data bit counts equal to the set up value. User can modify DPBITSEL(I2C_CTL0[9:8]) to adjust interrupt timing (DPCIF). The method is to use early entry handler to avoid missing interrupt events.

For example, one of circumstance is master send repeat start after write data ack, the two event will send interrupt very closely. Software may miss the write data ack status then slave will lose the last data transferred by master.

In order to ensure transfer/receive data correctly in Two-level BUFFER Mode (non-stretch mode), user must follow bellow control flow in two buffer mode.

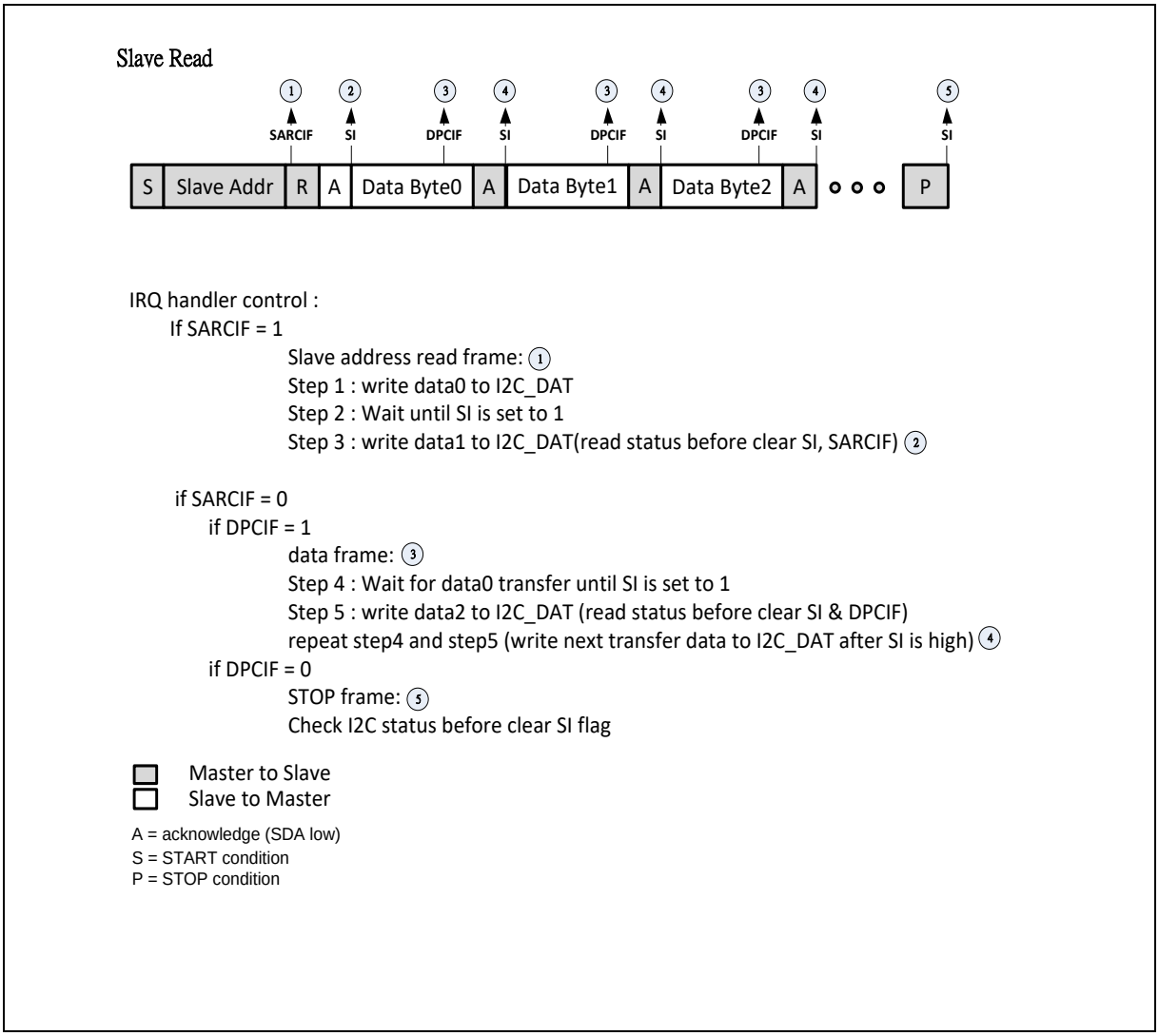
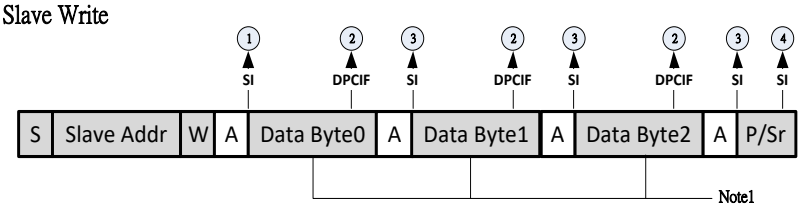


Figure 6.30-23 Two-level BUFFER Mode Slave Read Control

Slave Write



IRQ handler control :

If DPCIF = 0

slave write command interrupt, STOP or Repeat START frame : ① & ④

Step 1 : check status I2C status before clear SI flag

If DPCIF = 1 ②

Data frame:

Step 2 : Wait until SI is set to 1

Step 3 : If status is 0x80 or 0x88, need to read the data(I2C_DAT) transmit from master ③

(read status before clear SI and DPCIF)

Repeat step2 and step3

Note 1 : DPCIF is used to prevent missing interrupt event. In data frame DPCIF will cause hardware assert interrupt in advance.(before SI)

Master to Slave

Slave to Master

S = START condition

Sr = Repeat START condition

P = STOP condition

A = acknowledge (SDA low)

Figure 6.30-24 Two-level BUFFER Mode Slave Write Control

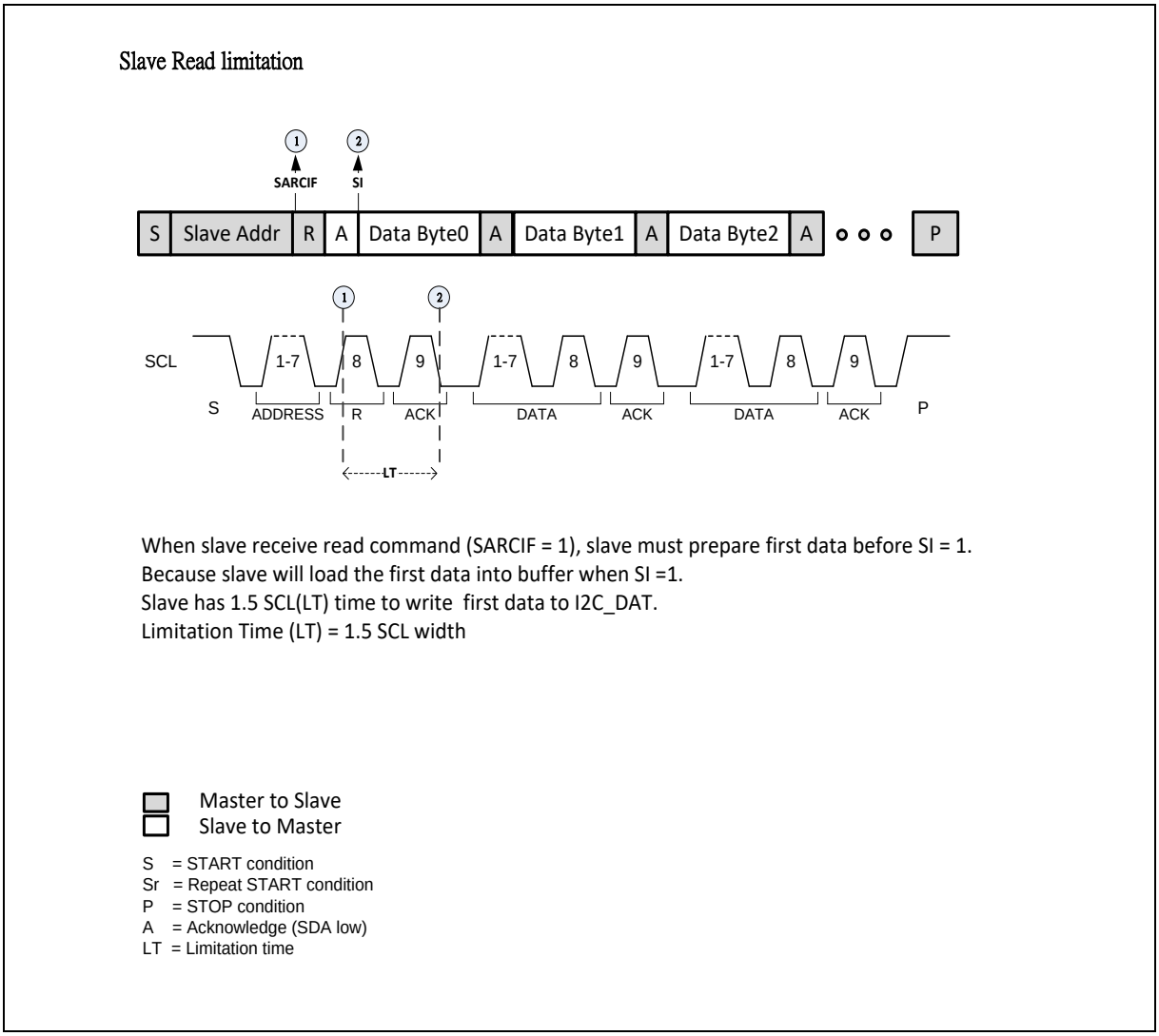


Figure 6.30-25 Two-level BUFFER Mode Slave Read Limitation

Note: When start up two-level buffer, PDMA cannot operate at the same time. PDMA and two-level buffer only use independently.

6.30.5.5 Programmable setup and hold times

To guarantee a correct data setup and hold time, the timing must be configured. By programming HTCTL (I2C_TMCTL[24:16]) to configure hold time and STCTL (I2C_TMCTL[8:0]) to configure setup time.

The delay timing refer peripheral clock (PCLK). When device stretch master clock, the setup and hold time configuration value will not affected by stretched.

User should focus the limitation of setup and hold time configuration, the timing setting must follow I^2C protocol. Once setup time configuration greater than design limitation, that means if setup time setting make SCL output less than three PCLKs, the I^2C controller cannot work normally due to SCL must sample three times. And once hold time configuration greater than I^2C clock limitation, I^2C will occur bus error. It is suggested that user calculate suitable timing with baud rate and protocol before setting timing. Table 6.30-2 shows the relationship between I^2C baud rate and PCLK, the number of table represent one clock duty contain how many PCLKs. Setup and hold time configuration even can program some extreme values in the design, but user should follow I^2C protocol standard.

I²C Baud Rate PCLK	100k	200k	400k	800k	1200k
12 MHz	120	60	30	15	10
24 MHz	240	120	60	30	20
48 MHz	480	240	120	60	40
72 MHz	720	360	180	90	60

Table 6.30-2 Relationship between I²C Baud Rate and PCLK

For setup time wrong adjustment example, assuming one SCL cycle contains 5 PCLKs and set STCTL (I2C_TMCTL[8:0]) to 3 that stretch three PCLKs for setup time setting. The setup time maximum setting value: ST_{limit} = (I2C_CLKDIV[7:0]+1) X 2 - 6.

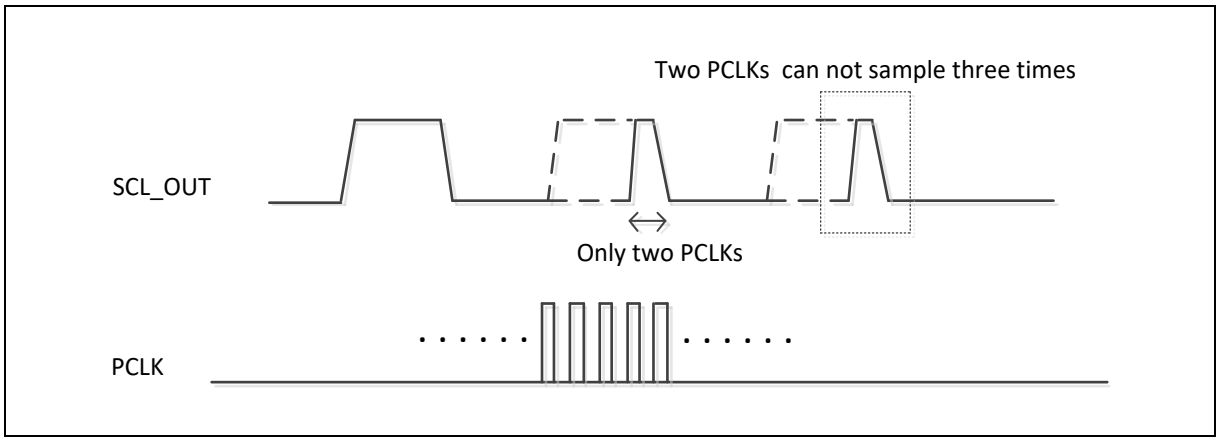


Figure 6.30-26 Setup Time Wrong Adjustment

For hold time wrong adjustment example, use I²C Baud Rate = 1200k and PCLK = 72 MHz, the SCL high/low duty = 60 PCLK. When HTCTL (I2C_TMCTL[24:16]) is set to 61 and STCTL (I2C_TMCTL[8:0]) is set to 0, then SDA output delay will over SCL high duty and cause bus error. The hold time maximum setting value: HT_{limit} = (I2C_CLKDIV[7:0]+1) X 2 - 9.

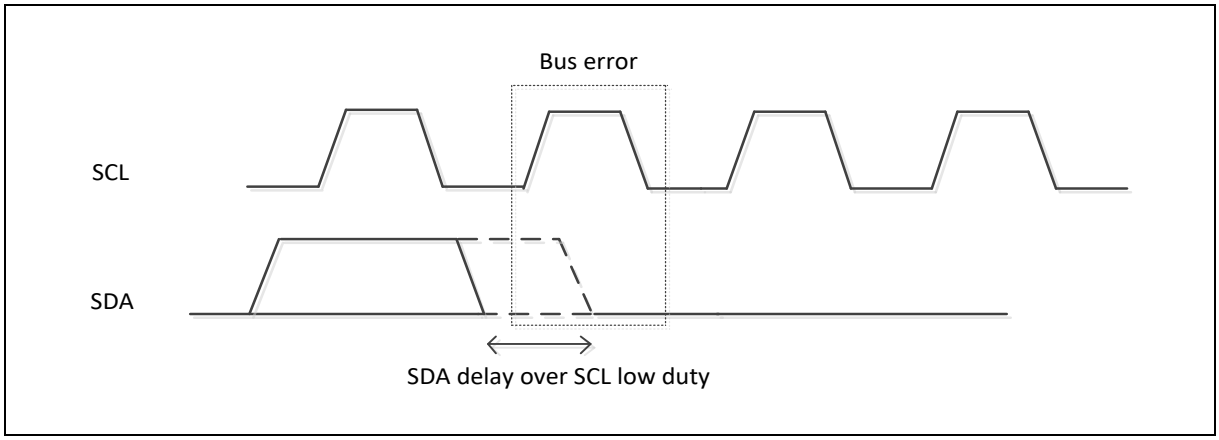


Figure 6.30-27 Hold Time Wrong Adjustment

6.30.5.6 I²C Protocol Registers

To control I²C port through the following fifteen special function registers: I2C_CTL0 (control register), I2C_STATUS0 (status register), I2C_DAT (data register), I2C_ADDRn (address registers, n=0~3),

I2C_ADDRMSKn (address mask registers, n=0~3), I2C_CLKDIV (clock rate register), I2C_TOCTL (Time-out control register), I2C_WKCTL(wake up control register) and I2C_WKSTS(wake up status register).

Address Registers (I2C_ADDR)

The I²C port is equipped with four slave address registers, I2C_ADDRn (n=0~3). The contents of the register are irrelevant when I²C is in Master mode. In Slave mode, the bit field ADDR(I2C_ADDRn[7:1]) must be loaded with the chip's own slave address. The I²C hardware will react if the contents of I2C_ADDRn are matched with the received slave address.

The I²C ports support the "General Call" function. If the GC bit (I2C_ADDRn [0]) is set the I²C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.

When the GC bit is set and the I²C is in Slave mode, it can receive the general call address by 00H after Master send general call address to I²C bus, then it will follow status of GC mode.

Slave Address Mask Registers (I2C_ADDRMSK)

The I²C bus controller supports multiple address recognition with four address mask registers I2C_ADDRMSKn (n=0~3). When the bit in the address mask register is set to 1, it means the received corresponding address bit is "Don't care". If the bit is set to 0, it means the received corresponding register bit should be exactly the same as address register.

Data Register (I2C_DAT)

This register contains a byte of serial data to be transmitted or a byte which just has been received. The CPU can be read from or written to the 8-bit (I2C_DAT [7:0]) directly while it is not in the process of shifting a byte. When I²C is in a defined state and the serial interrupt flag (SI) is set, data in I2C_DAT [7:0] remains stable. While data is being shifted out, data on the bus is simultaneously being shifted in; I2C_DAT [7:0] always contains the last data byte presented on the bus.

The acknowledge bit is controlled by the I²C hardware and cannot be accessed by the CPU. Serial data is shifted into I2C_DAT [7:0] on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into I2C_DAT [7:0], the serial data is available in I2C_DAT [7:0], and the acknowledge bit (ACK or NACK) is returned by the control logic during the ninth clock pulse. In order to monitor bus status while sending data, the bus data will be shifted to I2C_DAT[7:0] when sending I2C_DAT[7:0] to bus. In the case of sending data, serial data bits are shifted out from I2C_DAT [7:0] on the falling edge of SCL clocks, and is shifted to I2C_DAT [7:0] on the rising edge of SCL clocks. Figure 6.30-28 shows I²C Data Shifting Direction.

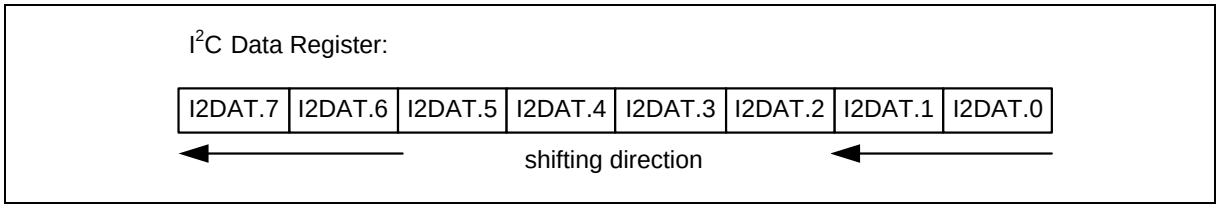


Figure 6.30-28 I²C Data Shifting Direction

Control Register (I2C_CTL0)

The CPU can be read from and written to I2C_CTL0 [7:0] directly. When the I²C port is enabled by setting I2CEN (I2C_CTL0 [6]) to high, the internal states will be controlled by I2C_CTL0 and I²C logic hardware.

There are two bits are affected by hardware: the SI bit is set when the I²C hardware requests a serial interrupt, and the STO bit is cleared when a STOP condition is present on the bus. The STO bit is also cleared when I2CEN = 0.

Once a new status code is generated and stored in I2C_STATUS0, the I²C Interrupt Flag bit SI (I2C_CTL0 [3]) will be set automatically. If the Enable Interrupt bit INTEN (I2C_CTL0 [7]) is set at this

time, the I²C interrupt will be generated. The bit field I2C_STATUS0[7:0] stores the internal state code, the content keeps stable until SI is cleared by software.

Status Register (I2C_STATUS0)

I2C_STATUS0 [7:0] is an 8-bit read-only register. The bit field I2C_STATUS0 [7:0] contains the status code and there are 26 possible status codes. All states are listed in Table 6.30-3. When I2C_STATUS0 [7:0] is F8H, no serial interrupt is requested. All other I2C_STATUS0 [7:0] values correspond to the defined I²C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2C_STATUS0[7:0] one cycle PCLK after SI set by hardware and is still present one cycle PCLK after SI reset by software.

In addition, the state 00H stands for a Bus Error, which occurs when a START or STOP condition is present at an incorrect position in the I²C format frame. A Bus Error may occur during the serial transfer of an address byte, a data byte or an acknowledge bit. To recover I²C from bus error, STO should be set and SI should be cleared to enter Not Addressed Slave mode. Then STO is cleared to release bus and to wait for a new communication. The I²C bus cannot recognize stop condition during this action when a bus error occurs.

Master Mode		Slave Mode	
STATUS	Description	STATUS	Description
0x08[1]	Start	0xA0	Slave Transmit Repeat Start or Stop
0x10[1]	Master Repeat Start	0xA8[1]	Slave Transmit Address ACK
0x18[1]	Master Transmit Address ACK	0xB8[1]	Slave Transmit Data ACK
0x20	Master Transmit Address NACK	0xC0	Slave Transmit Data NACK
0x28[1]	Master Transmit Data ACK	0xC8[1]	Slave Transmit Last Data ACK
0x30	Master Transmit Data NACK	0x60[1]	Slave Receive Address ACK
0x38	Master Arbitration Lost	0x68[1]	Slave Receive Arbitration Lost
0x40[1]	Master Receive Address ACK	0x80[1]	Slave Receive Data ACK
0x48	Master Receive Address NACK	0x88	Slave Receive Data NACK
0x50[1]	Master Receive Data ACK	0x70[1]	GC mode Address ACK
0x58	Master Receive Data NACK	0x78[1]	GC mode Arbitration Lost
0x00	Bus error	0x90[1]	GC mode Data ACK
		0x98	GC mode Data NACK
		0xB0[1]	Address Transmit Arbitration Lost
0xF0	If the BMDEN =1 and the ACKMEN bit is enabled, the information of I2C_STATUS0 will be fixed as 0xF0 in slave receive condition.		
0xF8	Bus Released **Note:** Status “0xF8” exists in both master/slave modes, and it won’t raise interrupt.		
Note [1]: No interrupt in PDMA mode			

Table 6.30-3 I²C Status Code Description

Clock Baud Rate Bits (I2C_CLKDIV)

The data baud rate of I²C is determines by DIVIDER(I2C_CLKDIV [9:0]) register when I²C is in Master mode, and it is not necessary in a Slave mode. In the Slave mode, I²C will automatically synchronize it with any clock frequency from master I²C device. In the slave mode, system clock frequency should

greater than I²C bus maximum clock 20 times.

The data baud rate of I²C setting is Data Baud Rate of I²C = (system clock) / (4x (I2C_CLKDIV [9:0] +1)). If system clock = 16 MHz, the I2C_CLKDIV [9:0] = 40 (28H), the data baud rate of I²C = 16 MHz/ (4x (40 +1)) = 97.5 kbit/s.

Time-out Control Register (I2C_TOCTL)

There is a 14-bit time-out counter which can be used to deal with the I²C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it overflows (TOIF=1) and generates I²C interrupt to CPU or stops counting by clearing TOCEN to 0. When time-out counter is enabled, writing 1 to the SI flag will reset counter and re-start up counting after SI is cleared. If I²C bus hangs up, it causes the I2C_STATUS0 and flag SI are not updated for a period, the 14-bit time-out counter may overflow and acknowledge CPU the I²C interrupt. Refer to Figure 6.30-29 for the 14-bit time-out counter. User may write 1 to clear TOIF to 0.

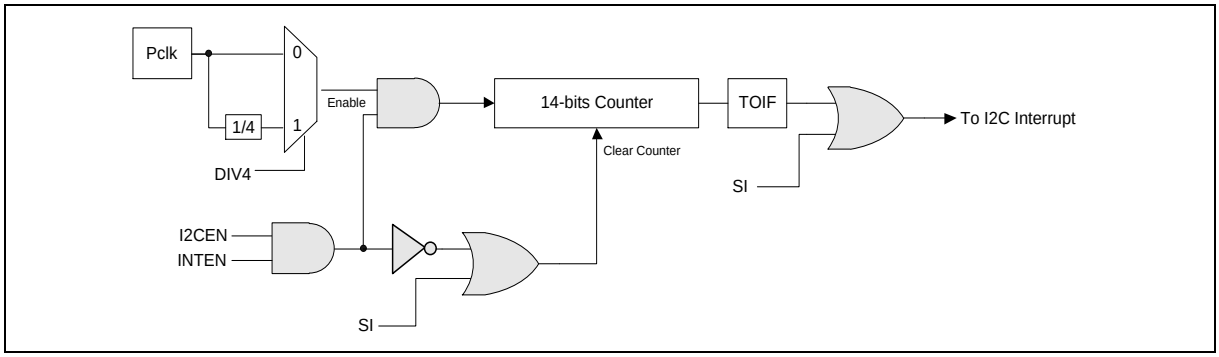


Figure 6.30-29 I²C Time-out Count Block Diagram

Wake-up Control Register (I2C_WKCTL)

When chip enters Power-down mode and set WKEN (I2C_WKCTL [0]) to 1, other I²C master can wake up the chip by addressing the I²C device, user must configure the related setting before entering sleep mode. The ACK bit cycle of address match frame is done in power-down. The controller will stretch the SCL to low when the address is matched the device's address and the ACK cycle done, then the I²C controller will go ahead. If NHDBUSEN (I2C_WKCTL [7]) is set, the controller will not stretch the SCL to low. Note that when the controller does not stretch the SCL to low, transmit or receive data will perform immediately. If data transmitted or received when SI event is not clear, user must reset the I²C controller and execute the original operation again.

Wake-up Status Register (I2C_WKSTS)

When system is woken up by other I²C master device, WKIF is set to indicate this event. User needs write "1" to clear this bit.

When the chip is woken-up by address match with one of the device address register (I2C_ADDRn), the user shall check the WKAKDONE (I2C_WKSTS [1]) bit is set to 1 to confirm the address byte has done. The WKAKDONE bit indicates that the ACK bit cycle of address byte is done in power-down. The controller will stretch the SCL to low when the address is matched the device's slave address and the ACK cycle done. The SCL is stretched until WKAKDONE is clear by user. If the frequency of SCL is low speed and the system has wakeup from address match frame, the user shall check WKAKDONE to confirm this frame has transaction done and then to do the wakeup procedure. Note that user cannot release WKIF through clearing the WKAKDONE bit to 0.

The WRSTSWK (I2C_WKSTS [2]) bit records the Read/Write command before the I²C controller sends address. The user can read this bit's status to prepare the next transmitted data (WRSTSWK = 0) or to wait the incoming data (WRSTSWK = 1) can be stored in time after the system is wake-up by the address match frame. Note that the WRSTSWK (I2C_WKSTS [2]) bit is cleared when writing one to the WKAKDONE (I2C_WKSTS [1]) bit.

When system is woken up by other I²C master device, WKIF is set to indicate this event. User needs to write "1" to clear this bit.

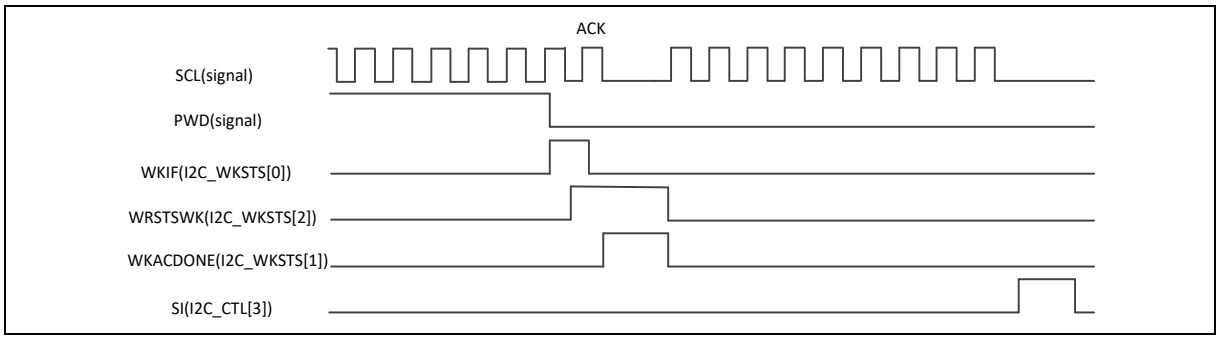


Figure 6.30-30 I²C Wake-Up Related Signals Waveform

I²C Control Register 1 (I2C_CTL1)

For the TWOBUFEN (I2C_CTL1[5]) bit, it is used to enable the two-level buffer for I²C transmitted or received buffer. It is used to improve the performance of the I²C bus.

If enable 10-bit addressing mode ADDR10EN (I2C_CTL1 [9]) is set, the I²C will run in 10-bit mode.

For PDMA function, set TXPDMAEN (I2C_CTL1 [0]) and RXPDMAEN (I2C_CTL1 [1]) can be set to operate. And set PDMARST (I2C_CTL1 [2]) to reset the PDMA control logic.

I²C Status Register 1 (I2C_STATUS1)

The I²C controller supports four slave address flag registers, ADMAT0, ADMAT1, ADMAT2 and ADMAT3 (I2C_STATUS1[3:0]). Every control register represent which address is used and set 1 to inform software.

I²C Timing Configure Control Register (I2C_TMCTL)

In order to configure setup/hold time, the HTCTL (I2C_TMCTL[24:16]) and STCTL (I2C_TMCTL[8:0]) are set based on actual demand.

Bus Management Control Register (I2C_BUSCTL)

The SMBus management control events are defined in this register. It includes the Acknowledge Control by Manual (ACKMEN (I2C_BUSCTL[0])), Packet Error Checking Enable (PECEN (I2C_BUSCTL[1])), device (BMDEN(I2C_BUSCTL[2])) or host (BMHEN (I2C_BUSCTL[3])) enable in this peripheral device. Both the alert and the suspend function can be set in ALERTEN (I2C_BUSCTL[4]), SCTLOSTS (I2C_BUSCTL[5])) and SCTLOEN (I2C_BUSCTL[6]).

The system bus management enables control by BUSEN(I2C_BUSCTL[7]) bit. The BUSTOCHK (I2C_BUSCTL[9]) is used to calculate the time-out of clock low in bus active and the idle period in bus Idle.

The calculated PEC (when the PECEN is set) value is transmitted or received can be controlled by PECTXEN bit (I2C_BUSCTL[8]).

There is a special bit of ACKM9SI (I2C_BUSCTL[11]). When the ACKMEN is set, there is SI interrupt in the 8th clock input and the user can read the data and status register. If the 8th clock bus is released when the SI interrupt is cleared, there is another SI interrupt event in the 9th clock cycle when this bit is set to 1 to know the bus status in this transaction frame done.

Set the PECDIEN (I2C_BUSCTL [13]), BCDIEN (I2C_BUSCTL[12]) or PECCLR (I2C_BUSCTL[10]) for PEC control flow.

I²C Bus Management Timer Control Register (I2C_BUSTCTL)

Set TORSTEN (I2C_BUSTCTL [4]), CLKTOIEN (I2C_BUSTCTL [3]), BUSTOIEN (I2C_BUSTCTL[2]), CLKTOEN (I2C_BUSTCTL [1]) and BUSTOEN (I2C_BUSTCTL [0]) for bus time-out or clock low time-out control flow.

I²C Bus Management Status Register (I2C_BUSSTS)

Monitor the PECDONE (I2C_BUSSTS [7]), BCDONE (I2C_BUSSTS[1]) or PECERR (I2C_BUSSTS[2]) for PEC control flow.

Monitor the SCTLDIN (I2C_BUSSTS[4]) for SUSCON input status.

I²C Byte Number Register (I2C_PKTSIZE)

When the PECEN bit (I2C_BUSCTL [1]) is set. The I²C controller will calculate the PEC value of the data on the bus. The PLDSIZE (I2C_PKTSIZE[8:0]) is used to define the data number in the bus. When the counter reach the value of PLDSIZE, the final PEC value will be transmitted or received automatically when the PECTXEN bit (I2C_BUSCTL[8]) is set.

I²C PEC VALUR Register (I2C_PKTCRC)

The register indicates the calculated PECCRC (I2C_PKTCRC[7:0]) value of data on the I²C bus. The detail of information is defined in the Bus Management (SMBus/PMBus Compatible).

I²C Bus Management Timer and I²C Clock Low Timer Register (I2C_BUSTOUT/ I2C_CLKTOUT)

Both of the definitions of these registers are described in the Bus Management (SMBus/PMBus Compatible).

6.30.5.7 Example for Random Read on EEPROM

The following steps are used to configure the I2C0 related registers when using I²C to read data from EEPROM.

1. Set I2C0 the multi-function pin as SCL and SDA pins. Refer to the Basic Configuration section for the multi-function configuration.
2. Enable I2C0 APB clock. Refer to the Basic Configuration section for the clock configuration.
3. Set I2C0RST=1 to reset I2C0 controller then set I2C0 controller to normal operation. Refer to the Basic Configuration section for the reset controller configuration.
4. Set I2CEN=1 to enable I2C0 controller in the “I2C_CTL0” register.
5. Give I2C0 clock a divided register value for I²C clock rate in the “I2C_CLKDIV”.
6. Enable system I2C0 IRQ in system “NVIC” control register.
7. Set INTEN=1 to enable I2C0 Interrupt in the “I2C_CTL0” register.
8. Set I2C0 address registers “I2C_ADDR0 ~ I2C_ADDR3”.

Random read operation is one of the methods of access EEPROM. The method allows the master to access any address of EEPROM space. Figure 6.30-31 shows the EEPROM random read operation.

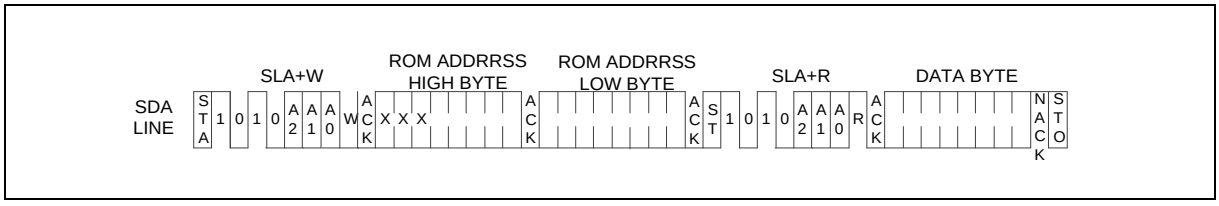


Figure 6.30-31 EEPROM Random Read

Figure 6.30-32 shows how to use the I²C controller to implement the protocol of EEPROM random read.

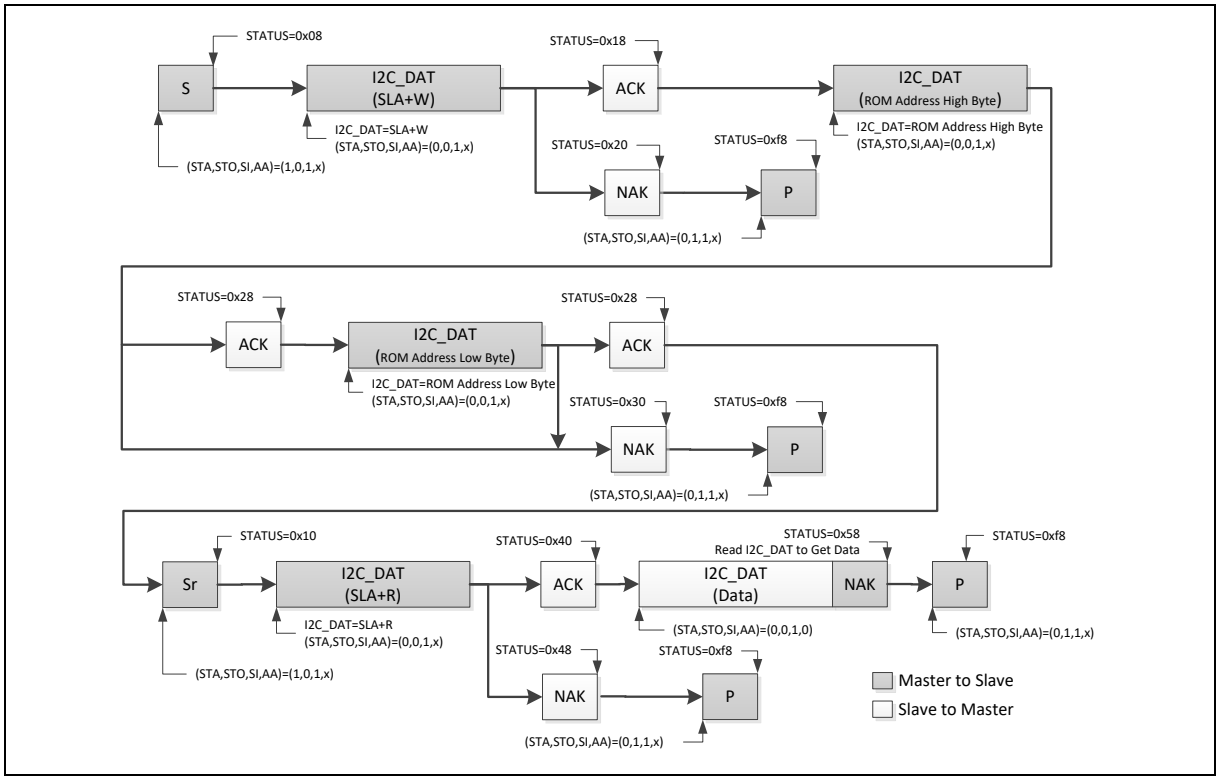


Figure 6.30-32 Protocol of EEPROM Random Read

The I²C controller, which is a master, sends START to bus. Then, it sends a SLA+W (Slave address + Write bit) to EERPOM followed by two bytes data address to set the EEPROM address to read. Finally, a Repeat START followed by SLA+R is sent to read the data from EEPROM.

6.30.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
I²C Base Address: **I2Cn_BA = 0x4008_0000 + (0x1000 *n)** **n= 0,1,2,3,4**				
I2C_CTL0	I2Cn_BA+0x00	R/W	I²C Control Register 0	0x0000_0000
I2C_ADDR0	I2Cn_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2C_DAT	I2Cn_BA+0x08	R/W	I²C Data Register	0x0000_0000
I2C_STATUS0	I2Cn_BA+0x0C	R	I²C Status Register 0	0x0000_00F8
I2C_CLKDIV	I2Cn_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000
I2C_TOCTL	I2Cn_BA+0x14	R/W	I²C Time-out Control Register	0x0000_0000
I2C_ADDR1	I2Cn_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2C_ADDR2	I2Cn_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2C_ADDR3	I2Cn_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000
I2C_ADDRMSK0	I2Cn_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2C_ADDRMSK1	I2Cn_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2C_ADDRMSK2	I2Cn_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2C_ADDRMSK3	I2Cn_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000
I2C_WKCTL	I2Cn_BA+0x3C	R/W	I²C Wake-up Control Register	0x0000_0000
I2C_WKSTS	I2Cn_BA+0x40	R/W	I²C Wake-up Status Register	0x0000_0000
I2C_CTL1	I2Cn_BA+0x44	R/W	I²C Control Register 1	0x0000_0000
I2C_STATUS1	I2Cn_BA+0x48	R/W	I²C Status Register 1	0x0000_0000
I2C_TMCTL	I2Cn_BA+0x4C	R/W	I²C Timing Configure Control Register	0x0000_0000
I2C_BUSCTL	I2Cn_BA+0x50	R/W	I²C Bus Management Control Register	0x0000_0000
I2C_BUSTCTL	I2Cn_BA+0x54	R/W	I²C Bus Management Timer Control Register	0x0000_0000
I2C_BUSSTS	I2Cn_BA+0x58	R/W	I²C Bus Management Status Register	0x0000_0000
I2C_PKTSIZE	I2Cn_BA+0x5C	R/W	I²C Packet Error Checking Byte Number Register	0x0000_0000
I2C_PKTCRC	I2Cn_BA+0x60	R	I²C Packet Error Checking Byte Value Register	0x0000_0000
I2C_BUSTOUT	I2Cn_BA+0x64	R/W	I²C Bus Management Timer Register	0x0000_0005
I2C_CLKTOUT	I2Cn_BA+0x68	R/W	I²C Bus Management Clock Low Timer Register	0x0000_0005

6.30.7 Register Description

I²C Control Register (I2C_CTL0)

Register	Offset	R/W	Description	Reset Value
I2C_CTL0	I2Cn_BA+0x00	R/W	I²C Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SARCIF	**DPCIF**	**SRCINTEN**	**DPCINTEN**	**Reserved**		**DPBITSEL**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTEN	**I2CEN**	**STA**	**STO**	**SI**	**AA**	**Reserved**	

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**SARCIF**	**Slave Address Read Command Interrupt Flag** This bit is set by hardware when I²C receive address match read command. **Note:** This bit is cleared by writing 1 to it.
[14]	**DPCIF**	**Data Phase Count Interrupt Flag** This bit is set by hardware when I²C transfer bit count equal to DPBITSEL setting **Note:** This bit is cleared by writing 1 to it.
[13]	**SRCINTEN**	**Slave Read Command Interrupt Enable Bit** 0 = Slave Read Command Interrupt Disabled. 1 = Slave Read Command Interrupt Enabled.
[12]	**DPCINTEN**	**Data Phase Count Interrupt Enable Bit** 0 = Data Phase Count Interrupt Disabled. 1 = Data Phase Count Interrupt Enabled.
[11:10]	**Reserved**	Reserved.
[9:8]	**DPBITSEL**	**Data Phase Bit Count Select** 00 = DPCIF never set by hardware. 01 = When I²C is transfer data and bit count equal to 6, DPCIF will be set by hardware. 10 = When I²C is transfer data and bit count equal to 7, DPCIF will be set by hardware. 11 = When I²C is transfer data and bit count equal to 8, DPCIF will be set by hardware.
[7]	**INTEN**	**Enable Interrupt** 0 = I²C interrupt Disabled. 1 = I²C interrupt Enabled.
[6]	**I2CEN**	**I²C Controller Enable Bit** Set to enable I²C serial function controller. When I2CEN=1 the I²C serial function enable. The multi-function pin function must set to SDA, and SCL of I²C function first. 0 = I²C controller Disabled.

		1 = I²C controller Enabled.
[5]	**STA**	**I²C START Control** Setting STA to logic 1 to enter Master mode, the I²C hardware sends a START or repeat START condition to bus when the bus is free.
[4]	**STO**	**I²C STOP Control** In Master mode, setting STO to transmit a STOP condition to bus then I²C controller will check the bus condition if a STOP condition is detected. This bit will be cleared by hardware automatically.
[3]	**SI**	**I²C Interrupt Flag** When a new I²C state is present in the I2C_STATUS0 register, the SI flag is set by hardware. If bit INTEN (I2C_CTL0 [7]) is set, the I²C interrupt is requested. SI must be cleared by software. Clear SI by writing 1 to this bit. For ACKMEN is set in slave read mode, the SI flag is set in 8th clock period for user to confirm the acknowledge bit and 9th clock period for user to read the data in the data buffer.
[2]	**AA**	**Assert Acknowledge Control** When AA =1 prior to address or data is received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when 1.) A slave is acknowledging the address sent from master, 2.) The receiver devices are acknowledging the data sent by transmitter. When AA=0 prior to address or data received, a Not acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line.
[1:0]	**Reserved**	Reserved.

I²C Data Register (I2C_DAT)

Register	Offset	R/W	Description	Reset Value
I2C_DAT	I2Cn_BA+0x08	R/W	I²C Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DAT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DAT**	**I²C Data** Bit [7:0] is located with the 8-bit transferred/received data of I²C serial port.

I²C Status Register (I2C_STATUS0)

Register	Offset	R/W	Description	Reset Value
I2C_STATUS0	I2Cn_BA+0x0C	R	I²C Status Register 0	0x0000_00F8

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STATUS							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**STATUS**	**I²C Status** The three least significant bits are always 0. The five most significant bits contain the status code. There are 28 possible status codes. When the content of I2C_STATUS0 is F8H, no serial interrupt is requested. Others I2C_STATUS0 values correspond to defined I²C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2C_STATUS0 one cycle after SI is set by hardware and is still present one cycle after SI has been reset by software. In addition, states 00H stands for a Bus Error. A Bus Error occurs when a START or STOP condition is present at an illegal position in the formation frame. Example of illegal position are during the serial transfer of an address byte, a data byte or an acknowledge bit.

I²C Clock Divided Register (I2C_CLKDIV)

Register	Offset	R/W	Description	Reset Value
I2C_CLKDIV	I2Cn_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NFCNT				Reserved		DIVIDER	
7	6	5	4	3	2	1	0
DIVIDER							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:12]	**NFCNT**	**Noise Filter Count** The register bits control the input filter width. 0 = Filter width 3*PCLK. 1 = Filter width 4*PCLK. *N* = Filter width (3+*N*)*PCKL. **Note:** Filter width Min :3*PCLK, Max : 18*PCLK
[11:10]	**Reserved**	Reserved.
[9:0]	**DIVIDER**	**I²C Clock Divided** Indicates the I²C clock rate: Data Baud Rate of I²C = (system clock) / (4x (I2C_CLKDIV+1)). **Note:** The minimum value of I2C_CLKDIV is 4.

I²C Time-out Control Register (I2C_TOCTL)

Register	Offset	R/W	Description	Reset Value
I2C_TOCTL	I2Cn_BA+0x14	R/W	I²C Time-out Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					TOCEN	TOCDIV4	TOIF

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**TOCEN**	**Time-out Counter Enable Bit** When enabled, the 14-bit time-out counter will start counting when SI is cleared. Setting flag SI to '1' will reset counter and re-start up counting after SI is cleared. 0 = Time-out counter Disabled. 1 = Time-out counter Enabled.
[1]	**TOCDIV4**	**Time-out Counter Input Clock Divided by 4** When enabled, the time-out period is extended 4 times. 0 = Time-out period is extend 4 times Disabled. 1 = Time-out period is extend 4 times Enabled.
[0]	**TOIF**	**Time-out Flag** This bit is set by hardware when I²C time-out happened and it can interrupt CPU if I²C interrupt enable bit (INTEN) is set to 1. **Note:** This bit will be cleared by software writing 1.

I²C Slave Address Register (ADDRx)

Register	Offset	R/W	Description	Reset Value
I2C_ADDR0	I2Cn_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2C_ADDR1	I2Cn_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2C_ADDR2	I2Cn_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2C_ADDR3	I2Cn_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					ADDR		
7	6	5	4	3	2	1	0
ADDR							GC

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:1]	**ADDR**	**I²C Address** The content of this register is irrelevant when I²C is in Master mode. In the slave mode, the seven most significant bits must be loaded with the chip's own address. The I²C hardware will react if either of the address is matched. **Note:** When software set 10'h000, the address cannot be used.
[0]	**GC**	**General Call Function** 0 = General Call Function Disabled. 1 = General Call Function Enabled.

I²C Slave Address Mask Register (ADDRMSKx)

Register	Offset	R/W	Description	Reset Value
I2C_ADDRMSK0	I2Cn_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2C_ADDRMSK1	I2Cn_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2C_ADDRMSK2	I2Cn_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2C_ADDRMSK3	I2Cn_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**ADDRMSK**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRMSK							**Reserved**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:1]	**ADDRMSK**	**I²C Address Mask** 0 = Mask Disabled (the received corresponding register bit should be exact the same as address register.). 1 = Mask Enabled (the received corresponding address bit is don't care.). I²C bus controllers support multiple address recognition with four address mask register. When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register. **Note:** The wake-up function cannot use address mask.
[0]	**Reserved**	Reserved.

I²C Wake-up Control Register (I2C_WKCTL)

Register	Offset	R/W	Description	Reset Value
I2C_WKCTL	I2Cn_BA+0x3C	R/W	I²C Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NHDBUSEN	**Reserved**						**WKEN**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**NHDBUSEN**	**I²C No Hold BUS Enable Bit** 0 = I²C hold bus after wake-up. 1= I²C don't hold bus after wake-up. **Note:** The I²C controller could respond when WKIF event is not clear, it may cause error data transmitted or received. If data transmitted or received when WKIF event is not clear, user must reset I²C controller and execute the original operation again.
[6:1]	**Reserved**	Reserved.
[0]	**WKEN**	**I²C Wake-up Enable Bit** 0 = I²C wake-up function Disabled. 1= I²C wake-up function Enabled.

I²C Wake-up Status Register (I2C_WKSTS)

Register	Offset	R/W	Description	Reset Value
I2C_WKSTS	I2Cn_BA+0x40	R/W	I²C Wake-up Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**WRSTSWK**	**WKAKDONE**	**WKIF**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**WRSTSWK**	**Read/Write Status Bit in Address Wakeup Frame** 0 = Write command be record on the address match wakeup frame. 1 = Read command be record on the address match wakeup frame. **Note:** This bit will be cleared when software can write 1 to WKAKDONE bit.
[1]	**WKAKDONE**	**Wakeup Address Frame Acknowledge Bit Done** 0 = The ACK bit cycle of address match frame is not done. 1 = The ACK bit cycle of address match frame is done in power-down. **Note:** This bit cannot release WKIF. This bit will be cleared by software writing 1.
[0]	**WKIF**	**I²C Wake-up Flag** When chip is woken up from Power-down mode by I²C, this bit is set to 1. **Note:** This bit will be cleared by software writing 1.

I²C Control Register 1 (I2C_CTL1)

Register	Offset	R/W	Description	Reset Value
I2C_CTL1	I2Cn_BA+0x44	R/W	I²C Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					SWITCHEN	ADDR10EN	PDMASTR
7	6	5	4	3	2	1	0
Reserved		TWOBUFEN	UDRIEN	OVRIEN	PDMARST	RXPDMAEN	TXPDMAEN

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**SWITCHEN**	**SCL And SDA Pin Switch Enable Bit** 0 = I²C use original pin configuration. 1 = I²C switch SCL and SDA pin configuration. **Note:** Original pin configuration table is shown in Basic Configuration section.
[9]	**ADDR10EN**	**Address 10-bit Function Enable Bit** 0 = Address match 10-bit function Disabled. 1 = Address match 10-bit function Enabled.
[8]	**PDMASTR**	**PDMA Stretch Bit** 0 = I²C send STOP automatically after PDMA transfer done. (only master TX) 1 = I²C SCL bus is stretched by hardware after PDMA transfer done if the SI is not cleared. (only master TX)
[7:6]	**Reserved**	Reserved.
[5]	**TWOBUFEN**	**Two-level BUFFER Enable Bit** 0 = Two-level buffer Disabled. 1 = Two-level buffer Enabled. Set to enable the two-level buffer for I²C transmitted or received buffer. It is used to improve the performance of the I²C bus.
[4]	**UDRIEN**	**I²C Under Run Interrupt Control Bit** Setting UDRIEN to logic 1 will send a interrupt to system when the TWOFF bit is enabled and there is under run event happened in transmitted buffer.
[3]	**OVRIEN**	**I²C over Run Interrupt Control Bit** Setting OVRIEN to logic 1 will send a interrupt to system when the TWOFF bit is enabled and there is over run event in received buffer.
[2]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the I²C request to PDMA.

[1]	**RXPDMAEN**	**PDMA Receive Channel Available** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[0]	**TXPDMAEN**	**PDMA Transmit Channel Available** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled.

I²C Status Register 1 (I2C_STATUS1)

Register	Offset	R/W	Description	Reset Value
I2C_STATUS1	I2Cn_BA+0x48	R/W	I²C Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							ONBUSY
7	6	5	4	3	2	1	0
UDR	OVR	EMPTY	FULL	ADMAT3	ADMAT2	ADMAT1	ADMAT0

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**ONBUSY**	**On Bus Busy (Read Only)** Indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected. 0 = The bus is IDLE (both SCL and SDA High). 1 = The bus is busy.
[7]	**UDR**	**I²C Under Run Status Bit (Read Only)** This bit indicates the transmitted two-level buffer TX or RX is under run when the TWOBUFEN = 1.
[6]	**OVR**	**I²C over Run Status Bit (Read Only)** This bit indicates the received two-level buffer TX or RX is over run when the TWOBUFEN = 1.
[5]	**EMPTY**	**TWO-LEVEL BUFFER EMPTY (Read Only)** This bit indicates two-level buffer TX or RX empty or not when the TWOBUFEN = 1. This bit is set when POINTER is equal to 0.
[4]	**FULL**	**TWO-LEVEL BUFFER FULL (Read Only)** This bit indicates two-level buffer TX or RX full or not when the TWOBUFEN = 1. This bit is set when POINTER is equal to 2.
[3]	**ADMAT3**	**I²C Address 3 Match Status** When address 3 is matched, hardware will inform which address used. This bit will be set to 1, and will be cleared by software writing 1.
[2]	**ADMAT2**	**I²C Address 2 Match Status** When address 2 is matched, hardware will inform which address used. This bit will be set to 1 and will be cleared by software writing 1.
[1]	**ADMAT1**	**I²C Address 1 Match Status** When address 1 is matched, hardware will inform which address used. This bit will be set to 1 and cleared by software writing 1.

[0]	**ADMAT0**	**I²C Address 0 Match Status** When address 0 is matched, hardware will inform which address used. This bit will be set to 1 and cleared by software writing 1.

I²C Timing Configure Control Register (I2C_TMCTL)

Register	Offset	R/W	Description	Reset Value
I2C_TMCTL	I2Cn_BA+0x4C	R/W	I²C Timing Configure Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							**HTCTL**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
HTCTL							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**STCTL**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STCTL							

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24:16]	**HTCTL**	**Hold Time Configure Control** This field is used to generate the delay timing between SCL falling edge and SDA rising edge in transmission mode. The delay hold time is numbers of peripheral clock = HTCTL x PCLK.
[15:9]	**Reserved**	Reserved.
[8:0]	**STCTL**	**Setup Time Configure Control** This field is used to generate a delay timing between SDA falling edge and SCL rising edge in transmission mode. The delay setup time is numbers of peripheral clock = STCTL x PCLK. **Note:** Setup time setting should not make SCL output less than three PCLKs.

I²C Bus Management Control Register (I2C_BUSCTL)

Register	Offset	R/W	Description	Reset Value
I2C_BUSCTL	I2Cn_BA+0x50	R/W	I²C Bus Management Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		PECDIEN	BCDIEN	ACKM9SI	PECCLR	BUSTOCHK	PECTXEN
7	6	5	4	3	2	1	0
BUSEN	SCTLOEN	SCTLOSTS	ALERTEN	BMHEN	BMDEN	PECEN	ACKMEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**PECDIEN**	**Packet Error Checking Byte Transfer Done Interrupt Enable Bit** 0 = PEC transfer done interrupt Disabled. 1 = PEC transfer done interrupt Enabled. **Note:** This bit is used in PECEN =1.
[12]	**BCDIEN**	**Packet Error Checking Byte Count Done Interrupt Enable Bit** 0 = Byte count done interrupt Disabled. 1 = Byte count done interrupt Enabled. **Note:** This bit is used in PECEN =1.
[11]	**ACKM9SI**	**Acknowledge Manual Enable Extra SI Interrupt** 0 = There is no SI interrupt in the 9th clock cycle when the BUSEN =1 and ACKMEN =1. 1 = There is SI interrupt in the 9th clock cycle when the BUSEN =1 and ACKMEN =1.
[10]	**PECCLR**	**PEC Clear at Repeat Start** The calculation of PEC starts when PECEN is set to 1 and it is cleared when the STA or STO bit is detected. This PECCLR bit is used to enable the condition of REPEAT START can clear the PEC calculation. 0 = PEC calculation is cleared by “Repeat Start” function Disabled. 1 = PEC calculation is cleared by “Repeat Start” function Enabled.
[9]	**BUSTOCHK**	**Timer Check in Idle State** The BUSTOCHK is used to calculate the time-out of clock low in bus active and the idle period in bus Idle. This bit is used to define which condition is enabled. 0 = BUSTOCHK is used to calculate the clock low period in bus active. 1 = BUSTOCHK is used to calculate the IDLE period in bus Idle. **Note:** The BUSY (I2C_BUSSTS[0]) indicates the current bus state.
[8]	**PECTXEN**	**Packet Error Checking Byte Transmission/Reception** 0 = No PEC transfer. 1 = PEC transmission is requested.

		Note: 1.This bit has no effect in slave mode when ACKMEN =0.
[7]	**BUSEN**	**BUS Enable Bit** 0 = The system management function Disabled. 1 = The system management function Enabled. **Note:** When the bit is enabled, the internal 14-bit counter is used to calculate the time out event of clock low condition.
[6]	**SCTLOEN**	**Suspend or Control Pin Output Enable Bit** 0 = The SUSCON pin in input. 1 = The output enable is active on the SUSCON pin.
[5]	**SCTLOSTS**	**Suspend/Control Data Output Status** 0 = The output of SUSCON pin is low. 1 = The output of SUSCON pin is high.
[4]	**ALERTEN**	**Bus Management Alert Enable Bit** Device Mode (BMHEN =0). 0 = Release the BM_ALERT pin high and Alert Response Header disabled: 0001100x followed by NACK if both of BMDEN and ACKMEN are enabled. 1 = Drive BM_ALERT pin low and Alert Response Address Header enables: 0001100x followed by ACK if both of BMDEN and ACKMEN are enabled. Host Mode (BMHEN =1). 0 = BM_ALERT pin not supported. 1 = BM_ALERT pin supported.
[3]	**BMHEN**	**Bus Management Host Enable Bit** 0 = Host function Disabled. 1 = Host function Enabled.
[2]	**BMDEN**	**Bus Management Device Default Address Enable Bit** 0 = Device default address Disabled. When the address 0'b1100001x comes and either BMDEN or ACKMEN is disabled, the device responses NACK. 1 = Device default address Enabled. When the address 0'b1100001x comes and both BMDEN and ACKMEN are enabled, the device responses ACK.
[1]	**PECEN**	**Packet Error Checking Calculation Enable Bit** 0 = Packet Error Checking Calculation Disabled. 1 = Packet Error Checking Calculation Enabled. **Note:** When I^2C enter power-down mode, the bit should be enabled after wake-up if needed PEC calculation.
[0]	**ACKMEN**	**Acknowledge Control by Manual** In order to allow ACK control in slave reception including the command and data, slave byte control mode must be enabled by setting the ACKMEN bit. 0 = Slave byte control Disabled. 1 = Slave byte control Enabled. The 9th bit can response the ACK or NACK according the received data by user. When the byte is received, stretching the SCL signal low between the 8th and 9th SCL pulse. **Note:** If the BMDEN =1 and this bit is enabled, the information of I2C_STATUS0 will be fixed as 0xF0 in slave receive condition.

I²C Bus Management Timer Control Register (I2C_BUSTCTL)

Register	Offset	R/W	Description	Reset Value
I2C_BUSTCTL	I2Cn_BA+0x54	R/W	I²C Bus Management Timer Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			TORSTEN	CLKTOIEN	BUSTOIEN	CLKTOEN	BUSTOEN

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**TORSTEN**	**Time Out Reset Enable Bit** 0 = I²C state machine reset Disabled. 1 = I²C state machine reset Enabled. (The clock and data bus will be released to high)
[3]	**CLKTOIEN**	**Extended Clock Time Out Interrupt Enable Bit** 0 = Clock time out interrupt Disabled. 1 = Clock time out interrupt Enabled.
[2]	**BUSTOIEN**	**Time-out Interrupt Enable Bit** BUSY =1. 0 = SCL low time-out interrupt Disabled. 1 = SCL low time-out interrupt Enabled. BUSY =0. 0 = Bus IDLE time-out interrupt Disabled. 1 = Bus IDLE time-out interrupt Enabled.
[1]	**CLKTOEN**	**Cumulative Clock Low Time Out Enable Bit** 0 = Cumulative clock low time-out detection Disabled. 1 = Cumulative clock low time-out detection Enabled. For Master, it calculates the period from START to ACK For Slave, it calculates the period from START to STOP
[0]	**BUSTOEN**	**Bus Time Out Enable Bit** 0 = Bus clock low time-out detection Disabled. 1 = Bus clock low time-out detection Enabled (bus clock is low for more than BUSTO (I2C_BUSTOUT[7:0]) (in BIDLE=0) or higher than BUSTO (in BIDLE =1).

I²C Bus Management Status Register (I2C_BUSSTS)

Register	Offset	R/W	Description	Reset Value
I2C_BUSSTS	I2Cn_BA+0x58	R/W	I²C Bus Management Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PECDONE	CLKTO	BUSTO	SCTLDIN	ALERT	PECERR	BCDONE	BUSY

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**PECDONE**	**PEC Byte Transmission/Receive Done** 0 = PEC transmission/ receive is not finished when the PECEN is set. 1 = PEC transmission/ receive is finished when the PECEN is set. **Note:** This bit will be cleared by software writing 1.
[6]	**CLKTO**	**Clock Low Cumulate Time-out Status** 0 = Cumulative clock low is no any time-out. 1 = Cumulative clock low time-out occurred. **Note:** This bit will be cleared by software writing 1.
[5]	**BUSTO**	**Bus Time-out Status** 0 = There is no any time-out or external clock time-out. 1 = A time-out or external clock time-out occurred. In bus busy, the bit indicates the total clock low time-out event occurred; otherwise, it indicates the bus idle time-out event occurred. **Note:** This bit will be cleared by software writing 1.
[4]	**SCTLDIN**	**Bus Suspend or Control Signal Input Status** 0 = The input status of SUSCON pin is 0. 1 = The input status of SUSCON pin is 1.
[3]	**ALERT**	**SMBus Alert Status** Device Mode (BMHEN =0). 0 = SMBALERT pin state is low. 1 = SMBALERT pin state is high. Host Mode (BMHEN =1). 0 = No SMBALERT event. 1 = There is SMBALERT event (falling edge) is detected in SMALERT pin when the BMHEN = 1 (SMBus host configuration) and the ALERTEN = 1. **Note:** 1. The SMBALERT pin is an open-drain pin, the pull-high resistor is must in the system. 2. This bit will be cleared by software writing 1.

[2]	**PECERR**	**PEC Error in Reception** 0 = PEC value equal to the received PEC data packet. 1 = PEC value doesn't match the receive PEC data packet. **Note:** This bit will be cleared by software writing 1.
[1]	**BCDONE**	**Byte Count Transmission/Receive Done** 0 = Byte count transmission/ receive is not finished when the PECEN is set. 1 = Byte count transmission/ receive is finished when the PECEN is set. **Note:** This bit will be cleared by software writing 1.
[0]	**BUSY**	**Bus Busy** Indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected 0 = Bus is IDLE (both SCL and SDA High). 1 = Bus is busy.

I²C Byte Number Register (I2C_PKTSIZE)

Register	Offset	R/W	Description	Reset Value
I2C_PKTSIZE	I2Cn_BA+0x5C	R/W	I²C Packet Error Checking Byte Number Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							PLDSIZE
7	6	5	4	3	2	1	0
PLDSIZE							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**PLDSIZE**	**Transfer Byte Number** The transmission or receive byte number in one transaction when the PECEN is set. The maximum transaction or receive byte is 256 Bytes. **Note:** The byte number counting includes address, command code, and data frame.

I²C PEC Value Register (I2C_PKTCRC)

Register	Offset	R/W	Description	Reset Value
I2C_PKTCRC	I2Cn_BA+0x60	R	I²C Packet Error Checking Byte Value Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PECCRC							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**PECCRC**	**Packet Error Checking Byte Value (Read Only)** This byte indicates the packet error checking content after transmission or receive byte count by using the C(x) = X8 + X2 + X + 1.

I²C Bus Management Timer Register (I2C_BUSTOUT)

Register	Offset	R/W	Description	Reset Value
I2C_BUSTOUT	I2Cn_BA+0x64	R/W	I²C Bus Management Timer Register	0x0000_0005

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BUSTO							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**BUSTO**	**Bus Management Time-out Value** Indicates the bus time-out value in bus is IDLE or SCL low. **Note:** If the user wants to revise the value of BUSTOUT, the TORSTEN (I2C_BUSTCTL[4]) bit shall be set to 1 and cleared to 0 first when the BUSEN(I2C_BUSCTL[7]) is set.

I²C Clock Low Timer Register (I2C_CLKTOUT)

Register	Offset	R/W	Description	Reset Value
I2C_CLKTOUT	I2Cn_BA+0x68	R/W	I²C Bus Management Clock Low Timer Register	0x0000_0005

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLKTO							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**CLKTO**	**Bus Clock Low Timer** The field is used to configure the cumulative clock extension time-out. **Note:** If the user wants to revise the value of CLKLTOUT, the TORSTEN bit shall be set to 1 and cleared to 0 first when the BUSEN is set.

6.31 Secure Digital Host Controller (SDH)

6.31.1 Overview

The Secure Digital Host Controller (SD Host) has DMAC unit and SD unit. The DMAC unit provides a DMA (Direct Memory Access) function for SD to exchange data between system memory and shared buffer (128 bytes), and the SD unit controls the interface of SD/SDHC. The SDHOST controller can support SD/SDHC and cooperated with DMAC to provide a fast data transfer between system memory and cards.

6.31.2 Features

- AMBA AHB master/slave interface compatible, for data transfer and register read/write.
- Supports single DMA channel.
- Supports hardware Scatter-Gather function.
- Using single 128 Bytes shared buffer for data exchange between system memory and cards.
- Synchronous design for DMA with single clock domain, AHB bus clock (HCLK).
- Interface with DMAC for register read/write and data transfer.
- Supports SD/SDHC card.
- Completely asynchronous design for Secure Digital with two clock domains, HCLK and Engine clock, note that frequency of HCLK should be higher than the frequency of peripheral clock.

6.31.3 Block Diagram

The block diagram and Card Pad Assignment of SDHOST Controller is shown as follows.

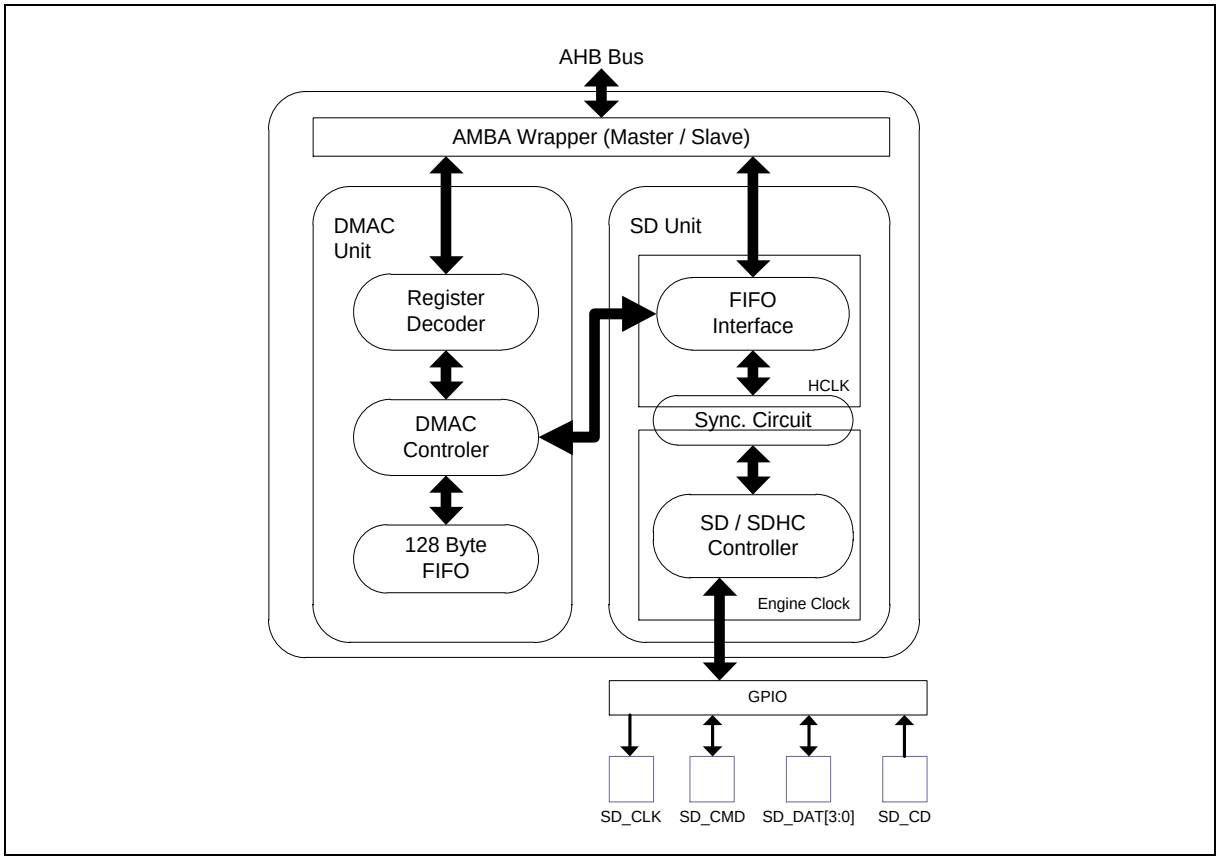


Figure 6.31-1 SD Host Controller Block Diagram

6.31.4 Basic Configuration

6.31.4.1 SD0 Basic Configuration

- Clock source Configuration
  - Select the source of SD0 engine clock on SDH0SEL (CLK_CLKSEL0[21:20]).
  - Select the clock divider number of SD0 engine clock on SDH0DIV (CLK_CLKDIV0[31:24]).
  - Enable SD0 engine clock in SDH0CKEN (CLK_AHBCLK[6]).
- Reset Configuration
  - Reset SD0 controller in SDH0RST (SYS_IPRST0[6]).

6.31.4.2 SD1 Basic Configuration

- Clock source Configuration
  - Select the source of SD1 engine clock on SDH1SEL (CLK_CLKSEL0[23:22]).
  - Select the clock divider number of SD1 engine clock on SDH1DIV (CLK_CLKDIV3[31:24]).
  - Enable SD1 engine clock in SDH1CKEN (CLK_AHBCLK[17]).
- Reset Configuration

- Reset SD1 controller in SDH1RST (SYS_IPRST0[17]).

6.31.5 Functional Description

The SD host provides an interface for SD/SDHC card access with 1-bit/4-bit data bus width.

The SD controller uses an independent clock source named SDCLK as engine clock. SDCLK can be completely asynchronous with system clock HCLK, software can change SD clock arbitrary. However, HCLK should be faster than SDCLK.

6.31.5.1 Basic Operation

This SD controller can generate all types of 48-bit command to the SD card and retrieve all types of response from SD card. After response in, the content of response will be stored at SDH_SDRSP0 and SDH_SDRSP1. SD controller will calculate CRC7 and check its correctness for response. If CRC7 is error, CRCIF (SDH_INTSTS[1]) will be set and CRC7 (SDH_INTSTS[2]) will be '0'. For response R1b, software should notice that after response in, SD card will put busy signal on data line DAT0; software should check DAT0STS (SDH_INTSTS[7]) with clock polling until it became high. For response R3, CRC7 is invalid; but SD controller will still calculate CRC7 and get an error result, software should ignore this error and clear CRCIF flag (SDH_INTSTS[1]).

The SD controller is composed of two state machines – command/response part and data part. For command/response part, the trigger bits are COEN (SDH_CTL[0]), RIEN (SDH_CTL[1]), R2EN (SDH_CTL[4]), CLK74OEN (SDH_CTL[5]) and CLK8OEN (SDH_CTL[6]) in SDH_CTL register. If software enables all of these bits, the execution priority will be CLK74OEN (SDH_CTL[5]) > COEN (SDH_CTL[0]) > RIEN (SDH_CTL[1])/R2EN (SDH_CTL[14) > CLK8OEN (SDH_CTL[6]), note that RIEN and R2EN can't be triggered at the same time. For data part, there are DIEN (SDH_CTL[2]) and DOEN (SDH_CTL[3]) for selection. Software can only trigger one of them at one time. If DIEN (SDH_CTL[2]) is triggered, the SD controller waits start bit from data line DAT0 immediately, and then get the specified amount data from SD card. After data-in, the SD controller will check the correctness of CRC16; if it is incorrect, CRCIF (SDH_INTSTS[1]) will be set and CRC16 (SDH_INTSTS[3]) will be '0'. If DOEN (SDH_CTL[3]) is triggered, the SD controller will wait until response in is finished, and then send specified amount data to the SD card. After data-out, the SD controller will get CRC status from SD card and check its correctness; CRCSTS (SDH_INTSTS[6:4]) should be '010', otherwise CRCIF (SDH_INTSTS[1]) will be set and CRCSTS (SDH_INTSTS[6:4]) will be the value it received.

If R2EN (SDH_CTL[4]) is triggered, the SD controller will receive response R2 (136 bits) from SD card, CRC7 and end bit will be dropped. The receiving data will be placed at DMA's buffer, starting from address offset 0x0.

6.31.5.2 Multiple Block Transfer

The SD controller also provides multiple block transfer function (change BLKLEN (SDH_BLEN[10:0]) to change the block length). Software can use this function to accelerate data transfer throughput. If CRC7, CRC16 or CRC status is error, SD controller will stop transfer and set CRCIF (SDH_INTSTS[1]), software should do engine reset when this situation occurred.

There is a hardware time-out mechanism for response in and data in inside SD engine. Software can specify a 24-bit time-out value at TOUT (SDH_TOUT[23:0]), and then SD controller will decide when to time-out according to this value.

6.31.5.3 DMA Controller

The SD host DMA controller provides a DMA (Direct Memory Access) function for SD host controller to exchange data between system memory (SRAM) and shared buffer (128 bytes). Arbitration of DMA request between SD host is done by DMA's bus master. Software just simply fills in the starting address and enables DMA, and then let DMA to handle the data transfer automatically.

There is a 128 bytes shared buffer inside DMA, it can provide multi-block transfers for SD host. Software can access these shared buffers directly when SD host is not in busy.

6.31.5.4 Programming Flow

Here is a simple example programming flow without DMA Scatter-Gather enable.

1. Set DMAEN (SDH_DMACTL[0]) to enable DMA.
2. Fill corresponding starting address in SDH_DMASA for SD host.
3. Trigger SD host to start DMA transfer.
4. Wait until transfer is finished.

Here is a simple example programming flow with DMA Scatter-Gather enable.

1. Set DMAEN (SDH_DMACTL[0]) to enable DMA (SDH_DMACTL[0]) and SGEN (SDH_DMACTL[3]) to enable Scatter-Gather function.
2. Fill corresponding starting address of Physical Address Descriptor (PAD) table in SDH_DMASA for SD host.
3. When bit-0 of SDH_DMASA is 1, the PAD will fetch in out of order, otherwise, it's fetched in order from PAD. The first time of writing bit-0 with 1 or not is not available for this function. The bits will be available in PAD table.
4. Trigger SD host to start DMA transfer.
5. Wait until transfer is finished.

The format of PAD table is shown in Figure 6.31-2. Note that the total byte count of all Pads must be equal to the byte count filled in SD host. EOT should be set to 1 in the last descriptor.

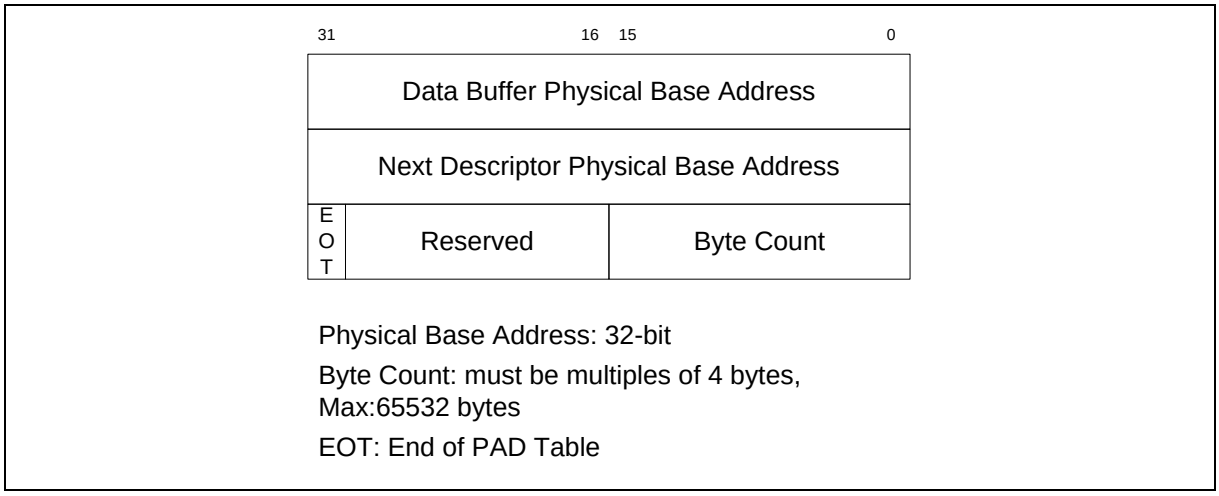


Figure 6.31-2 PAD (Physical Address Descriptor) Table Format

6.31.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
SDH Base address: **SDHx_BA = 0x4000_D000 + (0x1000 * x)** **x=0,1**				
SDH_FB_n **n=0,1..31**	SDHx_BA+0x000 + 0x4 * n	R/W	Shared Buffer	0x0000_0000
SDH_DMACTL	SDHx_BA+0x400	R/W	DMA Control and Status Register	0x0000_0000
SDH_DMASA	SDHx_BA+0x408	R/W	DMA Transfer Starting Address Register	0x0000_0000
SDH_DMABCNT	SDHx_BA+0x40C	R	DMA Transfer Byte Count Register	0x0000_0000
SDH_DMAINTEN	SDHx_BA+0x410	R/W	DMA Interrupt Enable Control Register	0x0000_0001
SDH_DMAINTSTS	SDHx_BA+0x414	R/W	DMA Interrupt Status Register	0x0000_0000
SDH_GCTL	SDHx_BA+0x800	R/W	Global Control and Status Register	0x0000_0000
SDH_GINTEN	SDHx_BA+0x804	R/W	Global Interrupt Control Register	0x0000_0001
SDH_GINTSTS	SDHx_BA+0x808	R/W	Global Interrupt Status Register	0x0000_0000
SDH_CTL	SDHx_BA+0x820	R/W	SD Control and Status Register	0x0101_0000
SDH_CMDARG	SDHx_BA+0x824	R/W	SD Command Argument Register	0x0000_0000
SDH_INTEN	SDHx_BA+0x828	R/W	SD Interrupt Control Register	0x0000_0A00
SDH_INTSTS	SDHx_BA+0x82C	R/W	SD Interrupt Status Register	0x000X_008C
SDH_RESP0	SDHx_BA+0x830	R	SD Receiving Response Token Register 0	0x0000_0000
SDH_RESP1	SDHx_BA+0x834	R	SD Receiving Response Token Register 1	0x0000_0000
SDH_BLEN	SDHx_BA+0x838	R/W	SD Block Length Register	0x0000_01FF
SDH_TOUT	SDHx_BA+0x83C	R/W	SD Response/Data-in Time-out Register	0x0000_0000

6.31.7 Register Description

Shared Buffer (SDH_FB_n)

Register	Offset	R/W	Description	Reset Value
SDH_FB_n **n=0,1…31**	SDHx_BA+0x000 + 0x4 * n	R/W	Shared Buffer	0x0000_0000

31	30	29	28	27	26	25	24
SB							
23	22	21	20	19	18	17	16
SB							
15	14	13	12	11	10	9	8
SB							
7	6	5	4	3	2	1	0
SB							

Bits	Description	
[31:0]	**SB**	**SDH Shared Buffer** Single 128 Bytes shared buffer for data exchange between system memory and cards.

DMA Control and Status Register (SDH_DMACTL)

Register	Offset	R/W	Description	Reset Value
SDH_DMACTL	SDHx_BA+0x400	R/W	DMA Control and Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						DMABUSY	Reserved
7	6	5	4	3	2	1	0
Reserved				SGEN	Reserved	DMARST	DMAEN

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**DMABUSY**	**DMA Transfer Is in Progress** This bit indicates if SD Host is granted and doing DMA transfer or not. 0 = DMA transfer is not in progress. 1 = DMA transfer is in progress.
[8:4]	**Reserved**	Reserved.
[3]	**SGEN**	**Scatter-gather Function Enable Bit** 0 = Scatter-gather function Disabled (DMA will treat the starting address in SDH_DMASA as starting pointer of a single block memory). 1 = Scatter-gather function Enabled (DMA will treat the starting address in SDH_DMASA as a starting address of Physical Address Descriptor (PAD) table. The format of these Pads' will be described later).
[2]	**Reserved**	Reserved.
[1]	**DMARST**	**Software Engine Reset** 0 = No effect. 1 = Reset internal state machine and pointers. The contents of control register will not be cleared. This bit will auto be cleared after few clock cycles. **Note:** The software reset DMA related registers.
[0]	**DMAEN**	**DMA Engine Enable Bit** 0 = DMA Disabled. 1 = DMA Enabled. **Note 1:** If this bit is cleared, DMA will ignore all requests from SD host and force bus master into IDLE state. **Note 2:** If target abort occurred, DMAEN will be cleared.

DMA Transfer Starting Address Register (SDH_DMASA)

Register	Offset	R/W	Description	Reset Value
SDH_DMASA	SDHx_BA+0x408	R/W	DMA Transfer Starting Address Register	0x0000_0000

31	30	29	28	27	26	25	24
DMASA							
23	22	21	20	19	18	17	16
DMASA							
15	14	13	12	11	10	9	8
DMASA							
7	6	5	4	3	2	1	0
DMASA							ORDER

Bits	Description	
[31:1]	DMASA	**DMA Transfer Starting Address** This field pads 0 as least significant bit indicates a 32-bit starting address of system memory (SRAM) for DMA to retrieve or fill in data. If DMA is not in normal mode, this field will be interpreted as a starting address of Physical Address Descriptor (PAD) table. **Note:** Starting address of the SRAM must be word aligned, for example, 0x0000_0000, 0x0000_0004.
[0]	ORDER	**Determined to the PAD Table Fetching Is in Order or Out of Order** 0 = PAD table is fetched in order. 1 = PAD table is fetched out of order. **Note:** The bit0 is valid in scatter-gather mode when SGEN = 1.

DMA Transfer Byte Count Register (SDH_DMABCNT)

Register	Offset	R/W	Description	Reset Value
SDH_DMABCNT	SDHx_BA+0x40C	R	DMA Transfer Byte Count Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						BCNT	
23	22	21	20	19	18	17	16
BCNT							
15	14	13	12	11	10	9	8
BCNT							
7	6	5	4	3	2	1	0
BCNT							

Bits	Description	
[31:26]	Reserved	Reserved.
[25:0]	BCNT	**DMA Transfer Byte Count (Read Only)** This field indicates the remained byte count of DMA transfer. The value of this field is valid only when DMA is busy; otherwise, it is 0.

DMA Interrupt Enable Control Register (SDH_DMAINTEN)

Register	Offset	R/W	Description	Reset Value
SDH_DMAINTEN	SDHx_BA+0x410	R/W	DMA Interrupt Enable Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						WEOTIEN	ABORTIEN

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WEOTIEN**	**Wrong EOT Encountered Interrupt Enable Bit** 0 = Interrupt generation Disabled when wrong EOT is encountered. 1 = Interrupt generation Enabled when wrong EOT is encountered.
[0]	**ABORTIEN**	**DMA Read/Write Target Abort Interrupt Enable Bit** 0 = Target abort interrupt generation Disabled during DMA transfer. 1 = Target abort interrupt generation Enabled during DMA transfer.

DMA Interrupt Status Register (SDH_DMAINTSTS)

Register	Offset	R/W	Description	Reset Value
SDH_DMAINTSTS	SDHx_BA+0x414	R/W	DMA Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						WEOTIF	ABORTIF

Bits	Description	
[31:2]	Reserved	Reserved.
[1]	WEOTIF	**Wrong EOT Encountered Interrupt Flag (Read Only)** When DMA Scatter-Gather function is enabled, and EOT of the descriptor is encountered before DMA transfer finished (that means the total sector count of all PAD is less than the sector count of SD host), this bit will be set. 0 = No EOT encountered before DMA transfer finished. 1 = EOT encountered before DMA transfer finished. **Note:** This bit is read only, but can be cleared by writing '1' to it.
[0]	ABORTIF	**DMA Read/Write Target Abort Interrupt Flag (Read Only)** 0 = No bus ERROR response received. 1 = Bus ERROR response received. **Note 1:** This bit is read only, but can be cleared by writing '1' to it. **Note 2:** When DMA's bus master received ERROR response, it means that target abort is happened. DMA will stop transfer and respond this event and then go to IDLE state. When target abort occurred or WEOTIF is set, software must reset DMA and SD host, and then transfer those data again.

Global Control and Status Register (SDH_GCTL)

Register	Offset	R/W	Description	Reset Value
SDH_GCTL	SDHx_BA+0x800	R/W	Global Control and Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						SDEN	GCTLRST

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**SDEN**	**Secure Digital Functionality Enable Bit** 0 = SD functionality Disabled. 1 = SD functionality Enabled.
[0]	**GCTLRST**	**Software Engine Reset** 0 = No effect. 1 = Reset SD host. The contents of control register will not be cleared. This bit will auto cleared after reset complete.

Global Interrupt Control Register (SDH_GINTEN)

Register	Offset	R/W	Description	Reset Value
SDH_GINTEN	SDHx_BA+0x804	R/W	Global Interrupt Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							DTAIEN

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	DTAIEN	DMA READ/WRITE Target Abort Interrupt Enable Bit 0 = DMA READ/WRITE target abort interrupt generation Disabled. 1 = DMA READ/WRITE target abort interrupt generation Enabled.

Global Interrupt Status Register (SDH_GINTSTS)

Register	Offset	R/W	Description	Reset Value
SDH_GINTSTS	SDHx_BA+0x808	R/W	Global Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							DTAIF

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**DTAIF**	**DMA READ/WRITE Target Abort Interrupt Flag (Read Only)** This bit indicates DMA received an ERROR response from internal AHB bus during DMA read/write operation. When Target Abort is occurred, please reset all engine. 0 = No bus ERROR response received. 1 = Bus ERROR response received. **Note:** This bit is read only, but can be cleared by writing '1' to it.

SD Control and Status Register (SDH_CTL)

Register	Offset	R/W	Description	Reset Value
SDH_CTL	SDHx_BA+0x820	R/W	SD Control and Status Register	0x0101_0000

31	30	29	28	27	26	25	24
Reserved				SDNWR			
23	22	21	20	19	18	17	16
BLKCNT							
15	14	13	12	11	10	9	8
DBW	CTLRST	CMDCODE					
7	6	5	4	3	2	1	0
CLKKEEP	CLK8OEN	CLK74OEN	R2EN	DOEN	DIEN	RIEN	COEN

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**SDNWR**	**NWR Parameter for Block Write Operation** This value indicates the NWR parameter for data block write operation in SD clock counts. The actual clock cycle will be SDNWR+1.
[23:16]	**BLKCNT**	**Block Counts to Be Transferred or Received** This field contains the block counts for data-in and data-out transfer. For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, software can use this function to accelerate data transfer and improve performance. Don't fill 0x0 to this field. **Note:** For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, the actual total length is BLKCNT * (BLKLEN +1).
[15]	**DBW**	**SD Data Bus Width Selection** 0 = Data bus width is 1-bit. 1 = Data bus width is 4-bit.
[14]	**CTLRST**	**Software Engine Reset** 0 = No effect. 1 = Reset the internal state machine and counters. The contents of control register will not be cleared (but RIEN, DIEN, DOEN and R2EN will be cleared). This bit will be auto cleared after few clock cycles.
[13:8]	**CMDCODE**	**SD Command Code** The bits contain the SD command code (0x00 – 0x3F).
[7]	**CLKKEEP**	**SD Clock Enable Control** 0 = SD host decided when to output clock and when to disable clock output automatically. 1 = SD clock always keeps free running.
[6]	**CLK8OEN**	**Generating 8 Clock Cycles Output Enable Bit** 0 = No effect. (Please use DMARST (SDH_CTL [0]) to clear this bit.) 1 = Enabled. The SD host will output 8 clock cycles. **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).

[5]	**CLK74OEN**	**Initial 74 Clock Cycles Output Enable Bit** 0 = No effect. (Please use DMARST (SDH_CTL [0]) to clear this bit.) 1 = Enabled. The SD host will output 74 clock cycles to SD card. **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).
[4]	**R2EN**	**Response R2 Input Enable Bit** 0 = No effect. (Please use DMARST (SDH_CTL [0]) to clear this bit.) 1 = Enabled. The SD host will wait to receive a response R2 from SD card and store the response data into DMC's Flash buffer (exclude CRC7). **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).
[3]	**DOEN**	**Data Output Enable Bit** 0 = No effect. (Please use DMARST (SDH_CTL [0]) to clear this bit.) 1 = Enabled. The SD host will transfer block data and the CRC16 value to SD card. **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).
[2]	**DIEN**	**Data Input Enable Bit** 0 = No effect. (Please use DMARST (SDH_CTL [0]) to clear this bit.) 1 = Enabled. The SD host will wait to receive block data and the CRC16 value from SD card. **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).
[1]	**RIEN**	**Response Input Enable Bit** 0 = No effect. (Please use DMARST (SDH_CTL [0]) to clear this bit.) 1 = Enabled. The SD host will wait to receive a response from SD card. **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).
[0]	**COEN**	**Command Output Enable Bit** 0 = No effect. (Please use DMARST (SDH_CTL [0]) to clear this bit.) 1 = Enabled. The SD host will output a command to SD card. **Note:** When operation is finished, this bit will be cleared automatically, so don't write 0 to this bit (the controller will be abnormal).

SD Command Argument Register (SDH_CMDARG)

Register	Offset	R/W	Description	Reset Value
SDH_CMDARG	SDHx_BA+0x824	R/W	SD Command Argument Register	0x0000_0000

31	30	29	28	27	26	25	24
ARGUMENT							
23	22	21	20	19	18	17	16
ARGUMENT							
15	14	13	12	11	10	9	8
ARGUMENT							
7	6	5	4	3	2	1	0
ARGUMENT							

Bits	Description	
[31:0]	ARGUMENT	**SD Command Argument** This register contains a 32-bit value specifies the argument of SD command from host controller to SD card. Before trigger COEN (SDH_CTL [0]), software should fill argument in this field.

SD Interrupt Control Register (SDH_INTEN)

Register	Offset	R/W	Description	Reset Value
SDH_INTEN	SDHx_BA+0x828	R/W	SD Interrupt Control Register	0x0000_0A00

31	30	29	28	27	26	25	24
Reserved	CDSRC	Reserved					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	WKIEN	DITOIEN	RTOIEN	Reserved		CDIEN	
7	6	5	4	3	2	1	0
Reserved						CRCIEN	BLKDIEN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**CDSRC**	**SD Card Detect Source Selection** 0 = From SD card's DAT3 pin. Host need clock to get data on pin DAT3. Please make sure CLKKEEP (SDH_CTL[7]) is 1 in order to generate free running clock for DAT3 pin. 1 = From GPIO pin.
[29:15]	**Reserved**	Reserved.
[14]	**WKIEN**	**Wake-up Signal Generating Enable Bit** Enable/Disable wake-up signal generating of SD controller when card is inserted or removed. 0 = SD Card interrupt to wake-up chip Disabled. 1 = SD Card interrupt to wake-up chip Enabled.
[13]	**DITOIEN**	**Data Input Time-out Interrupt Enable Bit** Enable/Disable interrupts generation of SD controller when data input time-out. Time-out value is specified at TOUT register. 0 = DITOIF (SDH_INTEN[13]) trigger interrupt Disabled. 1 = DITOIF (SDH_INTEN[13]) trigger interrupt Enabled.
[12]	**RTOIEN**	**Response Time-out Interrupt Enable Bit** Enable/Disable interrupts generation of SD controller when receiving response or R2 time-out. Time-out value is specified at TOUT register. 0 = RTOIF (SDH_INTEN[12]) trigger interrupt Disabled. 1 = RTOIF (SDH_INTEN[12]) trigger interrupt Enabled.
[11:9]	**Reserved**	Reserved.
[8]	**CDIEN**	**SD Card Detection Interrupt Enable Bit** Enable/Disable interrupts generation of SD controller when card is inserted or removed. 0 = CDIF (SDH_INTEN[8]) trigger interrupt Disabled. 1 = CDIF (SDH_INTEN[8]) trigger interrupt Enabled.

[7:2]	**Reserved**	Reserved.
[1]	**CRCIEN**	**CRC7, CRC16 and CRC Status Error Interrupt Enable Bit** 0 = CRCIF (SDH_INTEN[1]) trigger interrupt Disabled. 1 = CRCIF (SDH_INTEN[1]) trigger interrupt Enabled.
[0]	**BLKDIEN**	**Block Transfer Done Interrupt Enable Bit** 0 = BLKDIF (SDH_INTEN[0]) trigger interrupt Disabled. 1 = BLKDIF (SDH_INTEN[0]) trigger interrupt Enabled.

SD Interrupt Status Register (SDH_INTSTS)

Register	Offset	R/W	Description	Reset Value
SDH_INTSTS	SDHx_BA+0x82C	R/W	SD Interrupt Status Register	0x000X_008C

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					DAT1STS	Reserved	CDSTS
15	14	13	12	11	10	9	8
Reserved		DITOIF	RTOIF	Reserved		Reserved	CDIF
7	6	5	4	3	2	1	0
DAT0STS	CRCSTS			CRC16	CRC7	CRCIF	BLKDIF

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**DAT1STS**	**DAT1 Pin Status of SD Card (Read Only)** This bit indicates the DAT1 pin status of SD card.
[17]	**Reserved**	Reserved.
[16]	**CDSTS**	**Card Detect Status of SD (Read Only)** This bit indicates the card detect pin status of SD, and is used for card detection. When there is a card inserted in or removed from SD, software should check this bit to confirm if there is really a card insertion or removal. If CDSRC (SDH_INTEN[30]) = 0, to select DAT3 for card detection:. 0 = Card removed. 1 = Card inserted. If CDSRC (SDH_INTEN[30]) = 1, to select GPIO for card detection:. 0 = Card inserted. 1 = Card removed.
[15:14]	**Reserved**	Reserved.
[13]	**DITOIF**	**Data Input Time-out Interrupt Flag (Read Only)** This bit indicates that SD host counts to time-out value when receiving data (waiting start bit). 0 = Not time-out. 1 = Data input time-out. **Note:** This bit is read only, but can be cleared by writing '1' to it.
[12]	**RTOIF**	**Response Time-out Interrupt Flag (Read Only)** This bit indicates that SD host counts to time-out value when receiving response or R2 (waiting start bit). 0 = Not time-out. 1 = Response time-out. **Note:** This bit is read only, but can be cleared by writing '1' to it.

[11:9]	**Reserved**	Reserved.
[8]	**CDIF**	**SD Card Detection Interrupt Flag (Read Only)** This bit indicates that SD card is inserted or removed. Only when CDIEN (SDH_INTEN[8]) is set to 1, this bit is active. 0 = No card is inserted or removed. 1 = There is a card inserted in or removed from SD. **Note:** This bit is read only, but can be cleared by writing '1' to it.
[7]	**DAT0STS**	**DAT0 Pin Status of Current Selected SD Port (Read Only)** This bit is the DAT0 pin status of current selected SD port.
[6:4]	**CRCSTS**	**CRC Status Value of Data-out Transfer (Read Only)** SD host will record CRC status of data-out transfer. Software could use this value to identify what type of error is during data-out transfer. 010 = Positive CRC status. 101 = Negative CRC status. 111 = SD card programming error occurs.
[3]	**CRC16**	**CRC16 Check Status of Data-in Transfer (Read Only)** SD host will check CRC16 correctness after data-in transfer. 0 = Fault. 1 = OK.
[2]	**CRC7**	**CRC7 Check Status (Read Only)** SD host will check CRC7 correctness during each response in. If that response does not contain CRC7 information (ex. R3), then software should turn off CRCIEN (SDH_INTEN[1]) and ignore this bit. 0 = Fault. 1 = OK.
[1]	**CRCIF**	**CRC7, CRC16 and CRC Status Error Interrupt Flag (Read Only)** This bit indicates that SD host has occurred CRC error during response in, data-in or data-out (CRC status error) transfer. When CRC error is occurred, software should reset SD engine. Some response (ex. R3) doesn't have CRC7 information with it; SD host will still calculate CRC7, get CRC error and set this flag. In this condition, software should ignore CRC error and clears this bit manually. 0 = No CRC error is occurred. 1 = CRC error is occurred. **Note:** This bit is read only, but can be cleared by writing '1' to it.
[0]	**BLKDIF**	**Block Transfer Done Interrupt Flag (Read Only)** This bit indicates that SD host has finished all data-in or data-out block transfer. If there is a CRC16 error or incorrect CRC status during multiple block data transfer, the transfer will be broken and this bit will also be set. 0 = Not finished yet. 1 = Done. **Note:** This bit is read only, but can be cleared by writing '1' to it.

SD Receiving Response Token Register 0 (SDH_RESP0)

Register	Offset	R/W	Description	Reset Value
SDH_RESP0	SDHx_BA+0x830	R	SD Receiving Response Token Register 0	0x0000_0000

31	30	29	28	27	26	25	24
RESPTK0							
23	22	21	20	19	18	17	16
RESPTK0							
15	14	13	12	11	10	9	8
RESPTK0							
7	6	5	4	3	2	1	0
RESPTK0							

Bits	Description	
[31:0]	RESPTK0	**SD Receiving Response Token 0** SD host controller will receive a response token for getting a reply from SD card when RIEN (SDH_CTL[1]) is set. This field contains response bit 47-16 of the response token.

SD Receiving Response Token Register 1 (SDH_RESP1)

Register	Offset	R/W	Description	Reset Value
SDH_RESP1	SDHx_BA+0x834	R	SD Receiving Response Token Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
RESPTK1							

Bits	Description	
[7:0]	RESPTK1	**SD Receiving Response Token 1** The SD host controller will receive a response token for getting a reply from SD card when RIEN (SDH_CTL[1]) is set. This register contains the bit 15-8 of the response token.

SD Block Length Register (SDH_BLEN)

Register	Offset	R/W	Description	Reset Value
SDH_BLEN	SDHx_BA+0x838	R/W	SD Block Length Register	0x0000_01FF

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					BLKLEN		
7	6	5	4	3	2	1	0
BLKLEN							

Bits	Description	
[10:0]	BLKLEN	**SD BLOCK LENGTH in Byte Unit** An 11-bit value specifies the SD transfer byte count of a block. The actual byte count is equal to BLKLEN+1. **Note:** The default SD block length is 512 bytes

SD Response/Data-in Time-out Register (SDH_TOUT)

Register	Offset	R/W	Description	Reset Value
SDH_TOUT	SDHx_BA+0x83C	R/W	SD Response/Data-in Time-out Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
TOUT							
15	14	13	12	11	10	9	8
TOUT							
7	6	5	4	3	2	1	0
TOUT							

Bits	Description	
[23:0]	TOUT	**SD Response/Data-in Time-out Value** A 24-bit value specifies the time-out counts of response and data input. SD host controller will wait start bit of response or data-in until this value reached. The time period depends on SD engine clock frequency. Do not write a small number into this field, or you may never get response or data due to time-out. **Note:** Filling 0x0 into this field will disable hardware time-out function.

6.32 Programmable Serial I/O (PSIO)

6.32.1 Overview

Programmable Serial I/O (PSIO) provides a simple way to implement simple serial signal processing, e.g. UART and IR. The PSIO can control when the pin will output high or low and how long the pin need to output high or low. It also provides the easy way to sample the pin state.

6.32.2 Features

- Supports up to 8 PSIO pins, from PSIO pin0 to PSIO pin7
- Supports 6 clock sources, they are HXT, LXT, HIRC, LIRC, PLL/2, PCLK1
- Supports one clock divider, which can be divided from 1 to 255
- Supports slot controller for timing sequence control
  - Supports 4 slot controllers, 8 slots in each slot controller
  - Supports counting from 1 PSIO clock to 15 PSIO clocks in each slot
  - Supports 3 slot repeat modes:
    - Normal repeat mode
    - Normal repeat mode with infinity loops
    - Whole repeat mode
  - Supports 4 slot trigger conditions:
    - Triggered by software
    - Triggered by falling edge
    - Triggered by rising edge
    - Triggered by rising edge or falling edge
- Supports PSIO PIN for pin state control
  - Supports 8 check points to connect with slots in each pin
  - Supports 8 check point actions in each check point.
  - Supports 7 kinds of check point action to setting
    - Output high
    - Output low
    - Output data
    - Output toggle
    - Input data
    - Input status
    - Input status update
  - Supports 4 I/O modes, input, output, open-drain, and quasi
  - Supports switch I/O mode in different check points
- Supports 4 kinds of Interrupt trigger conditions
  - Two sets of configurable slot interrupt controllers
  - Mismatch interrupt when PSIO is enabled with PDMA

- Transfer Error interrupt
- Slot controller counting done interrupt

- Supports PDMA function

6.32.3 Block Diagram

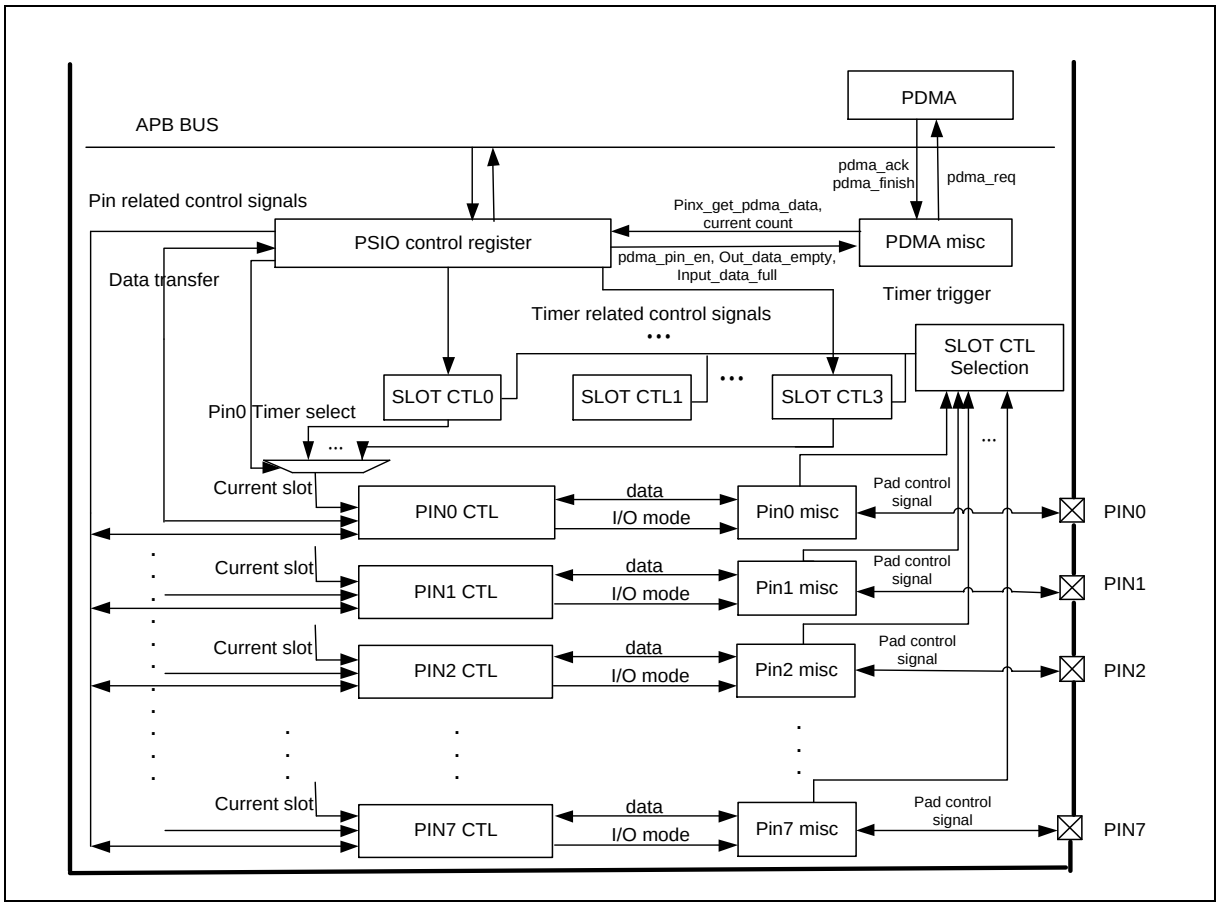
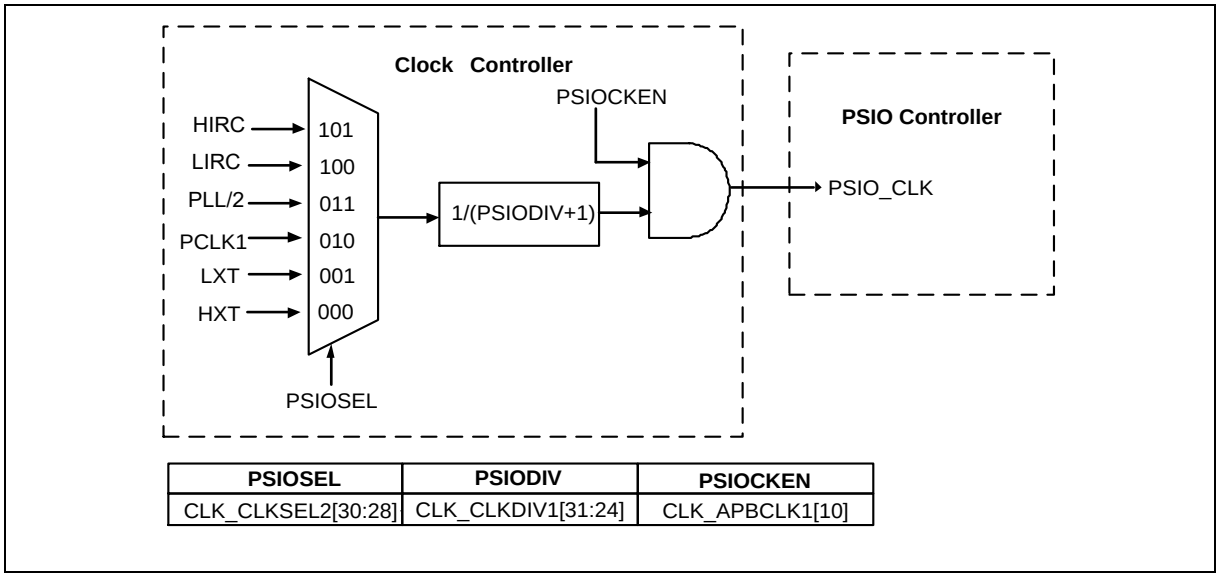


Figure 6.32-1 PSIO Block Diagram



PSIOSEL	PSIODIV	PSIOCKEN
CLK_CLKSEL2[30:28]	CLK_CLKDIV1[31:24]	CLK_APBCLK1[10]

Figure 6.32-2 PSIO Clock Control Diagram (8-bit Pre-scale Counter in Clock Controller)

6.32.4 Basic Configuration

- Clock Source Configuration
  - Select the source of PSIO peripheral clock on PSIOSEL (CLK_CLKSEL2[30:28]).
  - Select the clock divider number of PSIO peripheral clock on PSIODIV(CLK_CLKDIV1[31:24]).
  - Enable PSIO peripheral clock in PSIOCKEN (CLK_APBCLK1[10]).
- Reset Configuration
  - Reset PSIO controller in PSIORST (SYS_IPRST2[10]).

6.32.5 Data Rate Limitation

Due to only one PDMA channel for Tx and Rx independently, there is a data rate limitation for Tx and Rx in PDMA mode. The data rate is inversely proportional to the number of pins with PDMA enable. Table shows the fastest internal clock frequency and the maximum date rate with PDMA in the busy bus cases.

	1Pin	2Pin	3Pin	4Pin	5Pin	6Pin	7Pin	8Pin
Fastest Slot controller CLK Frequency	40.42 MHz	20.21 MHz	13.47 MHz	10.10 MHz	8.08 MHz	6.73 MHz	5.77 MHz	5.05 MHz
Data rate (bit/s)	40.42 Mbps	20.21 Mbps	13.47 Mbps	10.10 Mbps	8.08 Mbps	6.73 Mbps	5.77 Mbps	5.04 Mbps
Data rate (byte/s)	5.05 Mbps	2.52 Mbps	1.68 Mbps	1.26 Mbps	1.01 Mbps	0.84 Mbps	0.72 Mbps	0.63 Mbps

Table 6.32-1 PSIO Output Data Rate when Bus Traffic is Heavy

Note: Assuming it takes 18 cycles to read a data from SRAM and write it to peripheral or read it from peripheral and write it to SRAM, (Consider PDMA takes 2 cycles to access SRAM, 13 cycles for PDMA transfer data, and 3 cycles for PDMA access PSIO) and 2 cycles for PSIO sync flag. Assuming that when PSIO sends a request to the PDMA, the PDMA is just starting to execute another request. So it would take 18*2+2 = 38 cycles per data. Assuming the CPU clock frequency is 48 MHz, and 1 cycle period is 20.48 ns. And the buffer data size is 32 bit.

$$Data\ Rates = \frac{PSIO\ Buffer\ Data\ size\ (bits)}{PSIO\ pin\ numbers \times PDMA\ channels \times CPU\ Cycles\ of\ transfer\ data(ns)}$$

When pin numbers = 1, and the data size is 32 bit,

$$Data\ Rates = \frac{32(bits)}{1 \times 1 \times 38\ Cycles(ns)} = \frac{32(bits)}{38 \times 20.83(ns)} = 40.42\left(\frac{Mbits}{sec}\right) = 5.05\left(\frac{Mbytes}{sec}\right)$$

the fastest slot controller clock frequency is 40.42 MHz

When pin numbers = 2, and the data size is 32 bit,

$$Data\ Rates = \frac{32(bits)}{2 \times 1 \times 38\ Cycles(ns)} = \frac{32(bits)}{2 \times 38 \times 20.83(ns)} = 20.21\left(\frac{Mbits}{sec}\right) = 2.52\left(\frac{Mbytes}{sec}\right)$$

the fastest slot controller clock frequency is 20.21 MHz

Assuming it takes 16 cycles to read a data from SRAM and write it to peripheral or read it from peripheral and write it to SRAM. (Consider PDMA takes 1 cycles to access SRAM, 13 cycles for PDMA transfer data, and 2 cycles for PDMA access PSIO) and 2 cycles for PSIO sync flag. Assuming that when PSIO

sends a request to the PDMA, the PDMA is free. So it would take 16+2 = 18 cycles per data. Table shows the fastest internal clock frequency and the maximum date rate with PDMA in the best bus cases. Transfer data size is 32-bit and the CPU clock frequency is 48 MHz.

	1Pin	2Pin	3Pin	4Pin	5Pin	6Pin	7Pin	8Pin
Fastest slot controller CLK frequency	48 MHz	42.66 MHz	28.44 MHz	21.33 MHz	17.06 MHz	14.22 MHz	12.19 MHz	10.66 MHz
Data rate (bit/s)	48 Mbps	42.66 Mbps	28.44 Mbps	21.33 Mbps	17.06 Mbps	14.22 Mbps	12.19 Mbps	10.66 Mbps
Data rate (byte/s)	6 Mbps	5.33 Mbps	3.55 Mbps	2.66 Mbps	2.13 Mbps	1.77 Mbps	1.52 Mbps	1.33 Mbps

Table 6.32-2 PSIO Output Data Rate With Bus Is Free

When pin numbers = 2, and the data size is 32 bit,

$$Data\ Rates = \frac{32(bits)}{2 \times 1 \times 18\ Cycles(ns)} = \frac{32(bits)}{18 \times 20.83(ns)} = 42.66\left(\frac{Mbits}{sec}\right) = 5.33\left(\frac{Mbytes}{sec}\right)$$

the fastest slot controller clock frequency is 42.66 MHz

When pin numbers = 8, and the data size is 32 bit,

$$Data\ Rates = \frac{32(bits)}{8 \times 1 \times 18\ Cycles(ns)} = \frac{32(bits)}{8 \times 18 \times 20.83(ns)} = 10.66\left(\frac{Mbits}{sec}\right) = 1.33\left(\frac{Mbytes}{sec}\right)$$

the fastest slot controller clock frequency is 10.66 MHz

For the fastest input data rate, consider the PDMA will read the PSIO then update it to SRAM. The last pin that PDMA read will only takes 8 cycles to finish the whole data transfer. So when we calculated the fastest input data frequency, the total cycles that PDMA need to read data from PSIO can subtract 8 cycles. Note that the fastest input data rate does not represent the sample rate that the PSIO can sample in input pin.

	1Pin	2Pin	3Pin	4Pin	5Pin	6Pin	7Pin	8Pin
Fastest slot controller CLK frequency	48 MHz	48 MHz	33.39 MHz	24 MHz	18.73 MHz	15.36 MHz	13.01 MHz	11.29 MHz
Data rate (bit/s)	48 Mbps	48 Mbps	33.39 Mbps	24 Mbps	18.73 Mbps	15.36 Mbps	13.01 Mbps	11.29 Mbps
Data rate (byte/s)	6 Mbps	6 Mbps	4.17 Mbps	3 Mbps	2.34 Mbps	1.92 Mbps	1.62 Mbps	1.41 Mbps

Table 6.32-3 PSIO Input Data Rate with Bus Is Free

When pin numbers = 3, and the data size is 32 bit,

$$Data\ Rates = \frac{32(bits)}{(3 \times 18\ -\ 8)\ Cycles(ns)} = \frac{32(bits)}{(46) \times 20.83(ns)} = 33.39\left(\frac{Mbits}{sec}\right) = 4.17\left(\frac{Mbytes}{sec}\right)$$

the fastest slot controller clock frequency is 42.66 MHz

When pin numbers = 8, and the data size is 32 bit,

$$Data\ Rates = \frac{32(bits)}{(8 \times 18 - 8)Cycles(ns)} = \frac{32(bits)}{(136) \times 20.83(ns)} = 11.29\left(\frac{Mbits}{sec}\right) = 1.41\left(\frac{Mbytes}{sec}\right)$$

the fastest slot controller clock frequency is 11.29 MHz

6.32.6 Functional Description

There are a total of 4 slot controllers and 8 pins in PSIO. Each pin can choose one of the 4 slot controllers as the counting source. The slot controller has 8 slots. Each Slot can set counting cycles from 0 to 15. There have slot period repeat mode, which can choose some slot repeat counting. There also have I/O mode switch point, which can change I/O mode type at the selected check point.

The PSIO provides data transfer with PDMA, it can support up to 8 pins transfer data.

6.32.6.1 *Basic Operation*

Each slot can set counting cycles in PSIO_SCnSLOT. After filling the slot controller slot in PSIO_SCnSLOT, trigger source of slot controller is set from TRIGSRC(PSIO_SCnCTL[15:14]). There are four kind of trigger source, software trigger, START(PSIO_SCnCTL[16]), falling edge trigger from related input PIN, rising edge trigger from related input PIN, and rising edge or falling edge trigger from related input PIN.

Each PSIO pin can be linked to one of the slot controllers by SCSEL(PSIOn_GENCTL[25:24]). The pin can link the check point to the slot of related slot controller by setting CKPT0(PSIOn_CPCTL0[3:0]) to CKPT7 (PSIOn_CPCTL0[31:28]). After the slot controller start counting, pin will do the corresponding check point action filled in PSIOn_CPCTL1 when the slot of the slot controller is the same as the pin's check point.

When the slot controller finishes all setting slots, BUSY(PSIO_SCnCTL[24]) will be cleared to 0, and the BUSY flag will be cleared to 0, too.

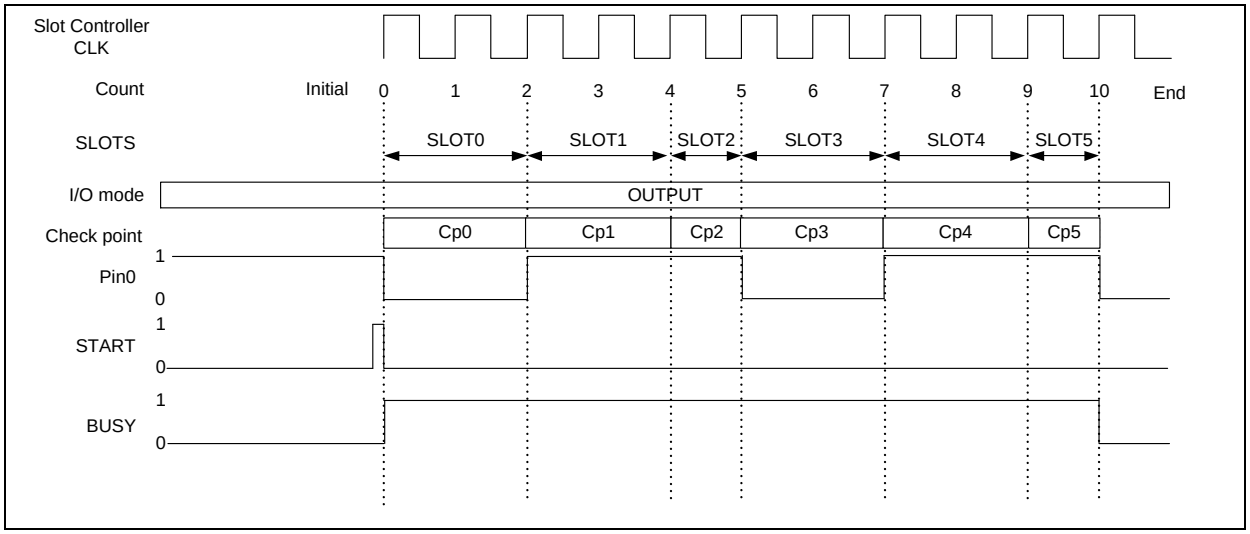


Figure 6.32-3 Basic Operation

6.32.6.2 *Slot Controller Trigger Condition*

There are two ways to trigger Slot Controller. First way to trigger slot controller is by programming 1 to START (PSIO_SCnCTL[16]), then the slot controller will be trigger immediately. If the slot controller also enable the PDMA function to transfer data from memory space to the PSIO pin which is intend to output data, then the slot controller will not start counting until the output data is ready.

The second way to trigger slot controller is by external pin edge detection. PSIO provides three kinds of edge trigger, they are falling edge trigger, rising edge trigger and both edge trigger. When the pin rises or fall, it must keep its pin state at least 2 PSIO_CLK or the edge variation will not be detected.

If the slot controller is triggered and the related pin action is output data, but there is no output data in output data buffer, then the output data underflow flag will be set.

6.32.6.3 *Slot Controller Repeat Mode*

There are three kinds of repeat mode for slot controller in PSIO. The first kind is partial repeat mode with finite loops. Initial repeat slot (INISLOT, PSIO_SCnCTL[3:0]) and end repeat slot (ENDSLOT, PSIO_SCnCTL[7:4]) decide which slots to repeat. Slot period loop count (SPLCNT, PSIO_SCnCTL[13:8]) decides how many times slots should be repeated.

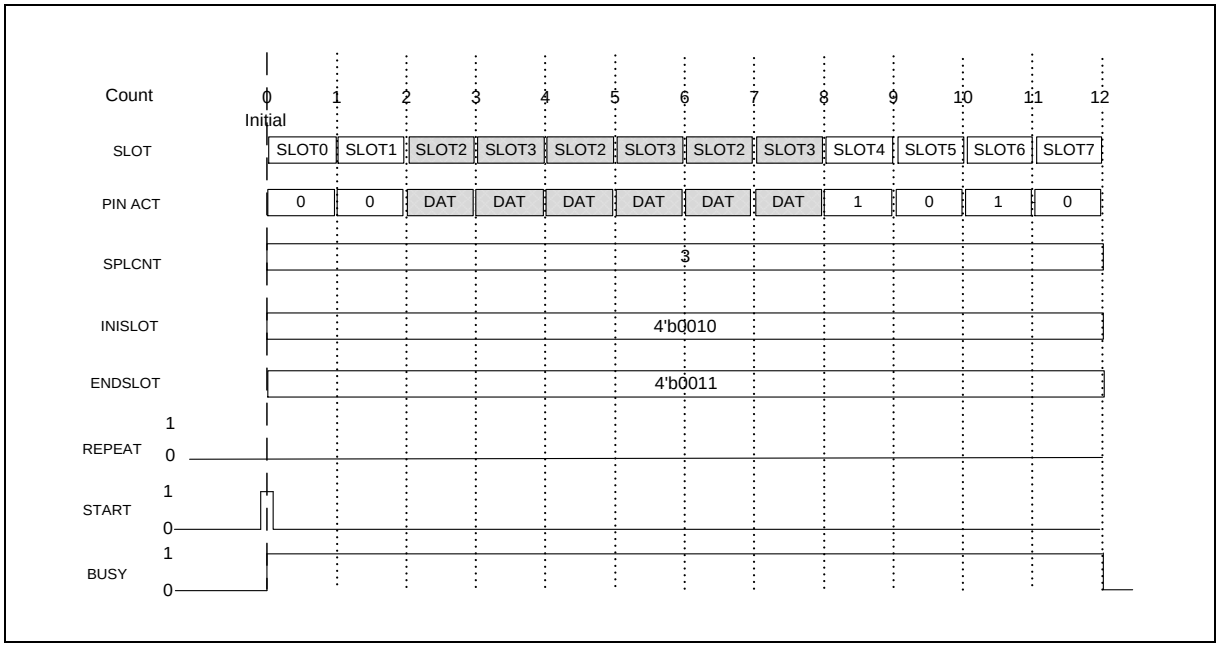


Figure 6.32-4 Slot Controller Normal Repeat Mode

The second kind is normal repeat mode with infinite loops. Filling 6'b111111 to SPLCNT(PSIO_SCnCTL[13:8]) will set slot controller enter partial repeat mode with infinite loops. In partial repeat mode with infinite loops, the slot controller will repeat counting until output data buffer is underflow or input data buffer is overflow or configuring 0 to START(PSIO_SCnCTL[16]), then the slot controller will stop at the end repeat slot when it receives stop signal.

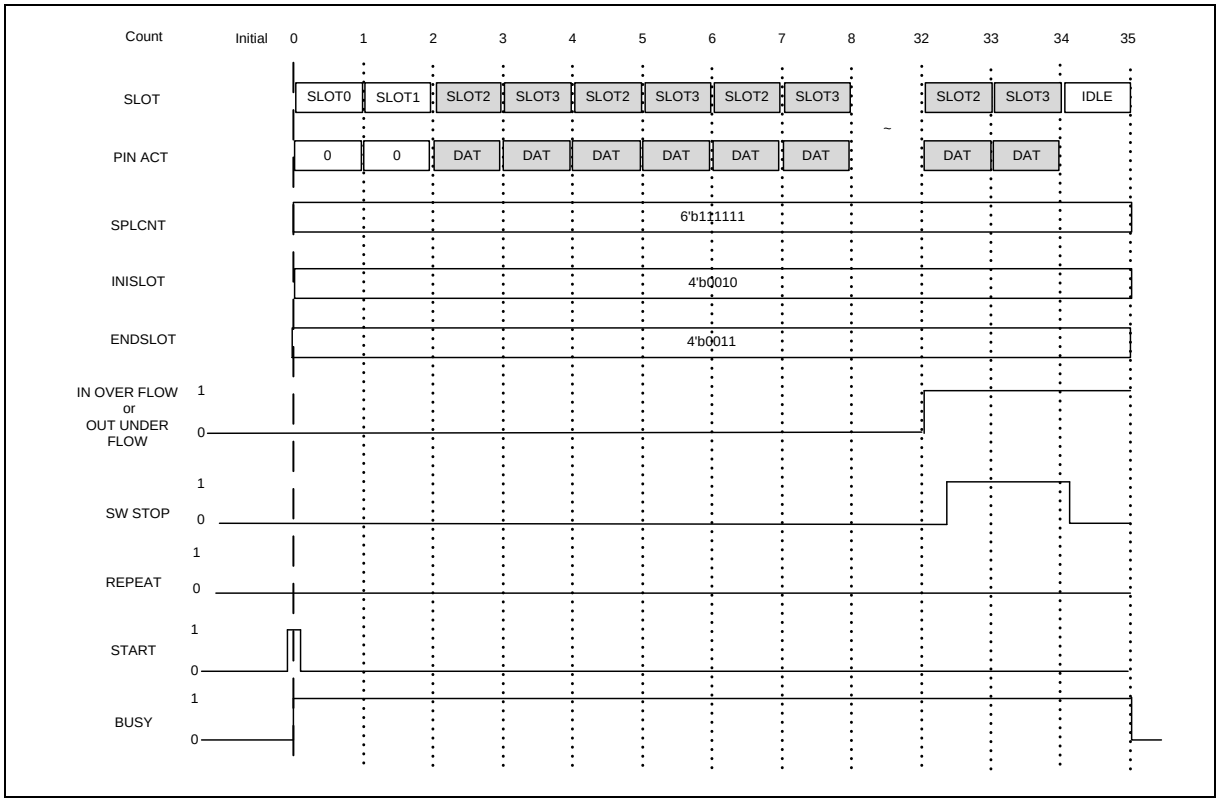


Figure 6.32-5 Slot Controller Normal Repeat Mode with Infinite Loops

The third kind of setting repeat mode with infinite loops is whole slots repeat mode. Configuring 1 to Repeat bit (PSIO_SCnCTL[17]) enables the whole repeat endless mode. In this mode, the enabled slot controller counts the slots enabled in PSIO_SCnSLOT in infinite loop. Only when the output data buffer is underflow or input data buffer is overflow or configuring 0 to START(PSIO_SCnCTL[16]), then the slot controller will stop at the last enable slot when it receives stop signal.

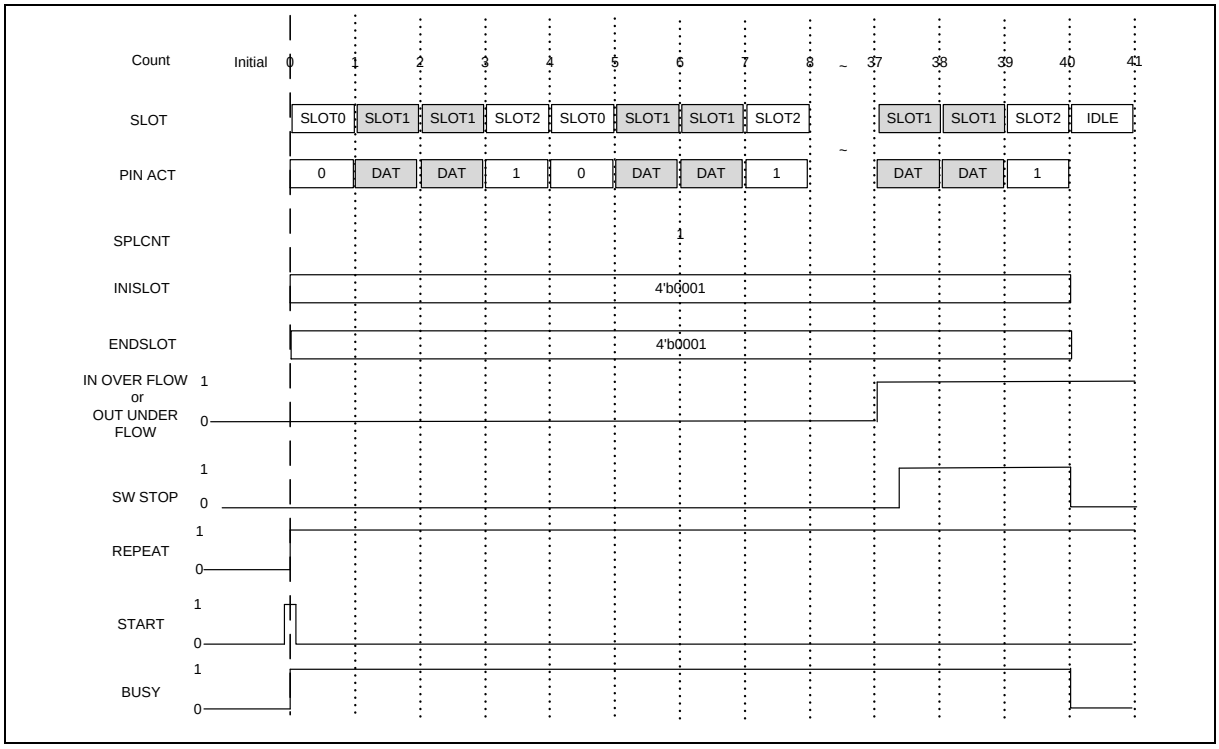


Figure 6.32-6 Slot Controller Repeat Mode with Whole Repeat Mode

If the pin enables input data and output data in the same loop with switch point function, the slot controller is set in repeat mode with finite loops, the conditions of stop counting is the output data underflow or input data overflow occurs.

6.32.6.4 *Pin Initial State, Interval State and IDLE Flag*

There are two control bits that determine the pin state is Initial or interval. The pin uses initial or interval as default setting depending on the IDLE(PSIO_SCnCTL[25]) and the slot controller has started or not. Before the slot controller starts counting, the pin state is determined by the initial bit. After the slot controller finishes counting and the IDLE(PSIO_SCnCTL[25]) is 0, the pin state is determined by the interval bit. As long as the slot controller starts counting, the IDLE(PSIO_SCnCTL[25]) will be cleared to 0 automatically. Only when software writes 1 to set IDLE(PSIO_SCnCTL[25]) to 1, the IDLE(PSIO_SCnCTL[25]) will be set to 1 or it will always keep 0.

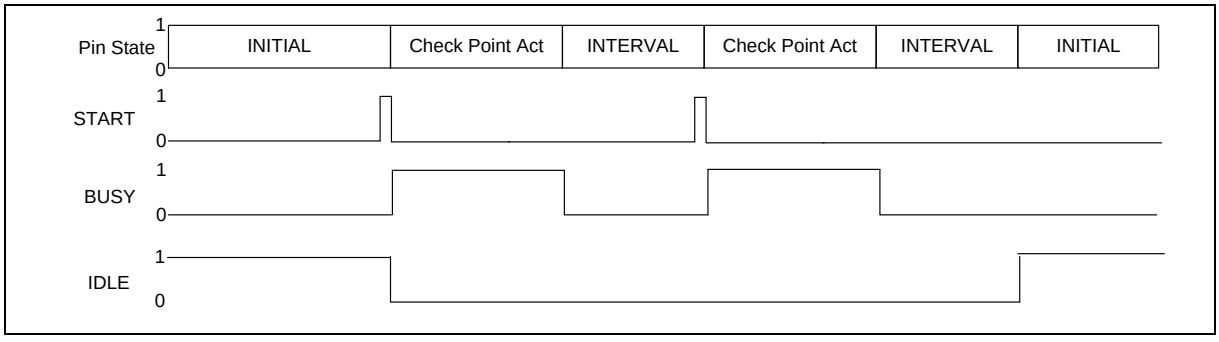


Figure 6.32-7 Initial State, Interval State and IDLE Flag

6.32.6.5 *Pin Switch I/O State during Counting*

In Slot action, user can use switch point function to change the I/O state at different slot. PSIO supports at most switching I/O operation mode twice in a loop. User can configure MODESW1, MODESW0,

SW0CP, and SW1CP in PSIOn_GENCTL.

There is a hardware limitation for mode switch from output to input. We need to consider the pad delay and the device response time to choose the slot length of input check point which can affect the time PSIO sample the device responds. The input delay from input pin to input data buffer is 4 PSIO_CLK.

The PIN OUTPUT delay represents the delay from PSIO to chip pin. The device response time represents the time that device responds the PSIO. The PIN INPUT delay represents the delay from chip pin to PSIO. For PSIO switch the I/O mode at check point 2, and the device respond is transfer before count 4. PSIO input data buffer will sample it at count 8. Due to the input data sample delay and PIN OUTPUT DELAY, PIN INPUT DELAY, device response time, the shortest slot length of check point 2 is 8 for the switch I/O mode in input mode to sample the device responds.

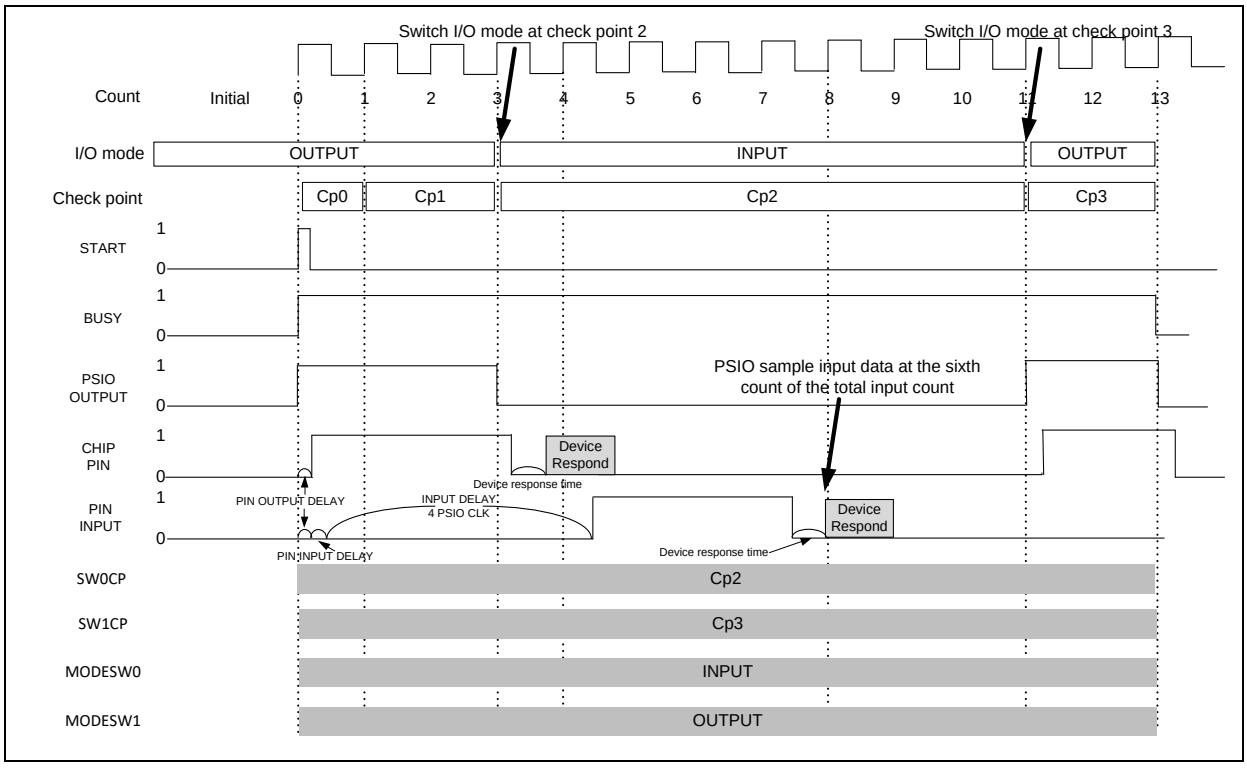


Figure 6.32-8 Switch I/O Mode

The relationship between PIN OUTPUT DELAY, PIN INPUT DELAY, Device response time and Devise is shown in Figure 6.32-9.

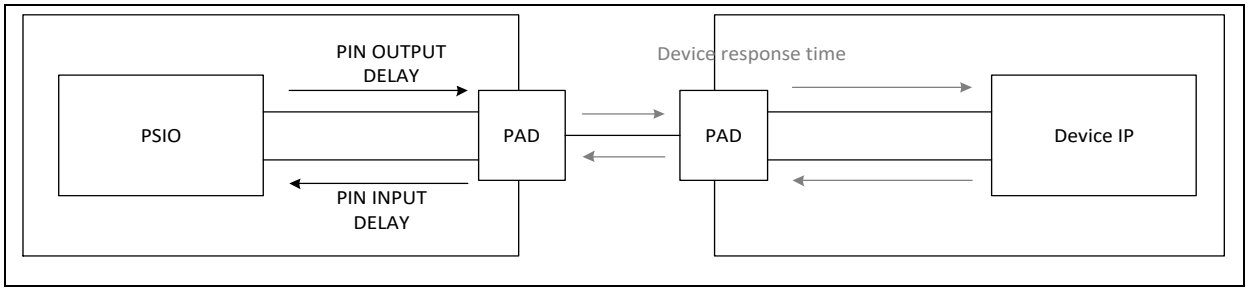


Figure 6.32-9 PIN Delay Time and Devise Response Time

The PIN OUTPUT DELAY and PIN INPUT DELAY may vary from different pins for the routing. But the Maximum PIN OUTPUT DELAY variation of each pins and PIN INPUT DELAY variation of each pins are shorter than the “one PSIO clock length”, as shown in Table 6.32-4. “one PSIO clock length” here means the frequency configured by the PSIO clock source and PSIO clock divider, and its frequency does not exceed the slew rate that I/O can support.

	MAX PSIO OUTPUT DELAY variation	**MAX PSIO INPUT DELAY variation**
	1 PSIO CLK length	1 PSIO CLK length

Table 6.32-4 PIN OUTPUT DELAY and PIN INPUT DELAY

6.32.6.6 *Input/Output Data Buffer*

Each PSIO_PIN has its own 32-bit input data buffer and output data buffer to let the buffer operation be more efficient. The 32-bit data buffer can be sliced into shorter buffer when data width is shorter than 16-bit by setting OUTDEPTH(PSIOn_DATCTL[25:24]) and INDEPTH(PSIOn_DATCTL[29:28]).

When the OUTDATWD (PSIOn_DATCTL[4:0]) and INDATWD (PSIOn_DATCTL[12:9]) are larger than 16-bit, the setting of OUTDEPTH(PSIOn_DATCTL[25:24]) and INDEPTH(PSIOn_DATCTL[29:28]) will be ignored. The PSIO_PIN will have one 32-bit data buffer.

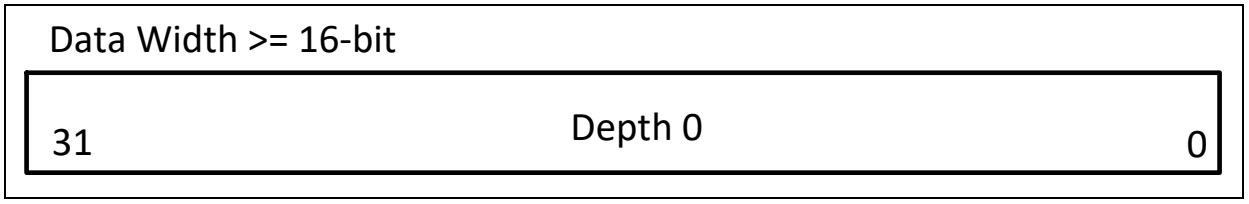


Figure 6.32-10 Data Width Larger Than 16-bit

When the OUTDATWD (PSIOn_DATCTL[4:0]) and INDATWD (PSIOn_DATCTL[12:9]) is between 8-bit to 16-bit, the 32-bit data buffer can be sliced into two 16-bit buffer by setting OUTDEPTH[0](PSIOn_DATCTL[24]) and INDEPTH[0](PSIOn_DATCTL[28]).

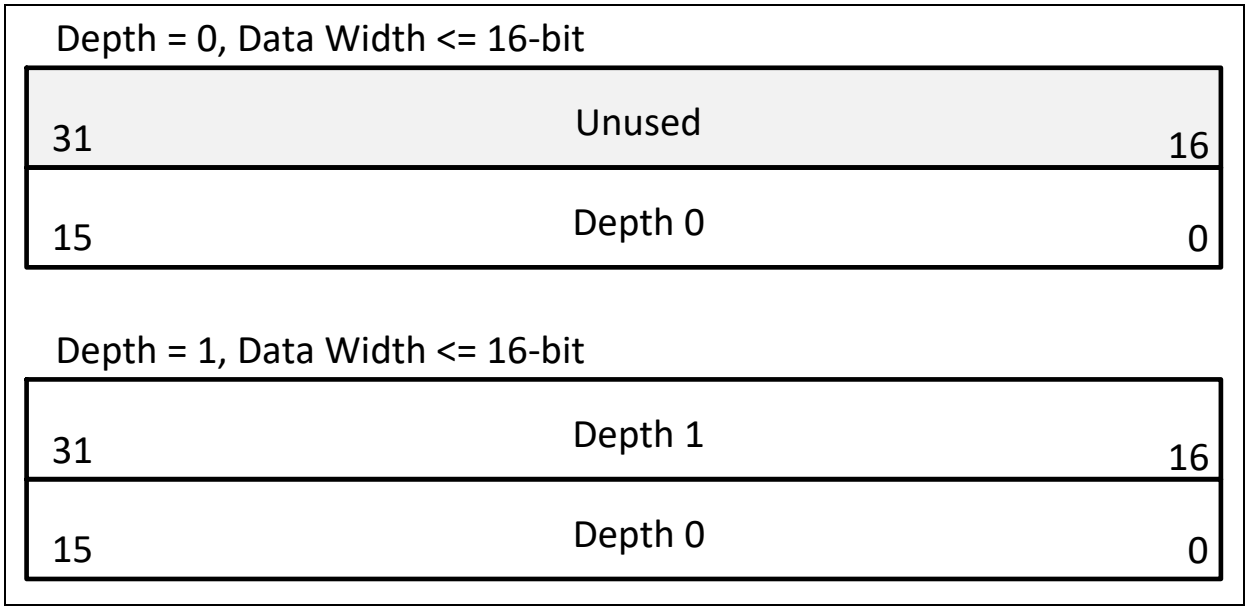


Figure 6.32-11 Data Width Between 16-bit and 8-bit

When the OUTDATWD (PSIOn_DATCTL[4:0]) and INDATWD (PSIOn_DATCTL[12:9]) is between 1-bit to 8-bit, the 32-bit data buffer can be sliced into four 8-bit buffer by setting OUTDEPTH[1:0](PSIOn_DATCTL[25:24]) and INDEPTH[1:0](PSIOn_DATCTL[29:28]).

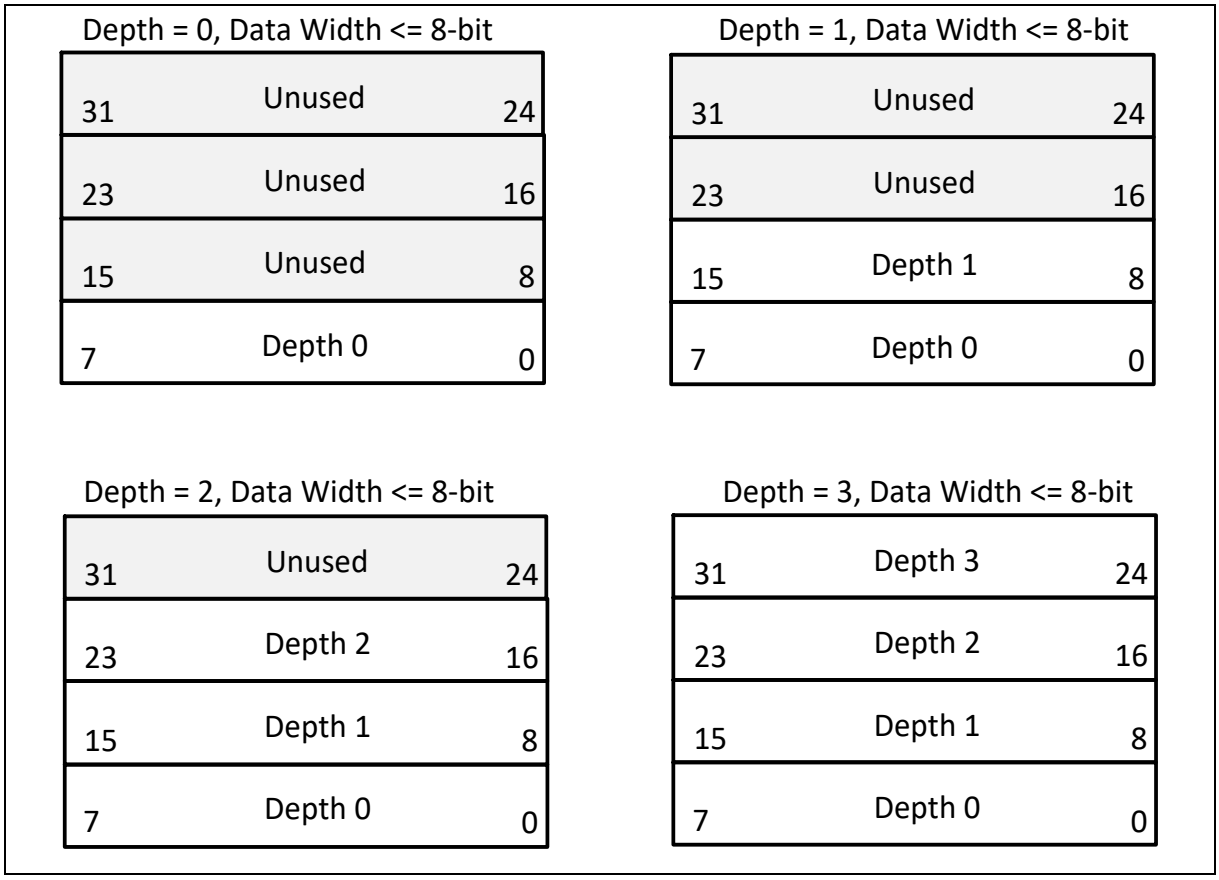


Figure 6.32-12 Data Width Smaller Than 8-bit

When PSIO outputs data, the relation of output data buffer and the empty flag is shown in Figure 6.32-13.

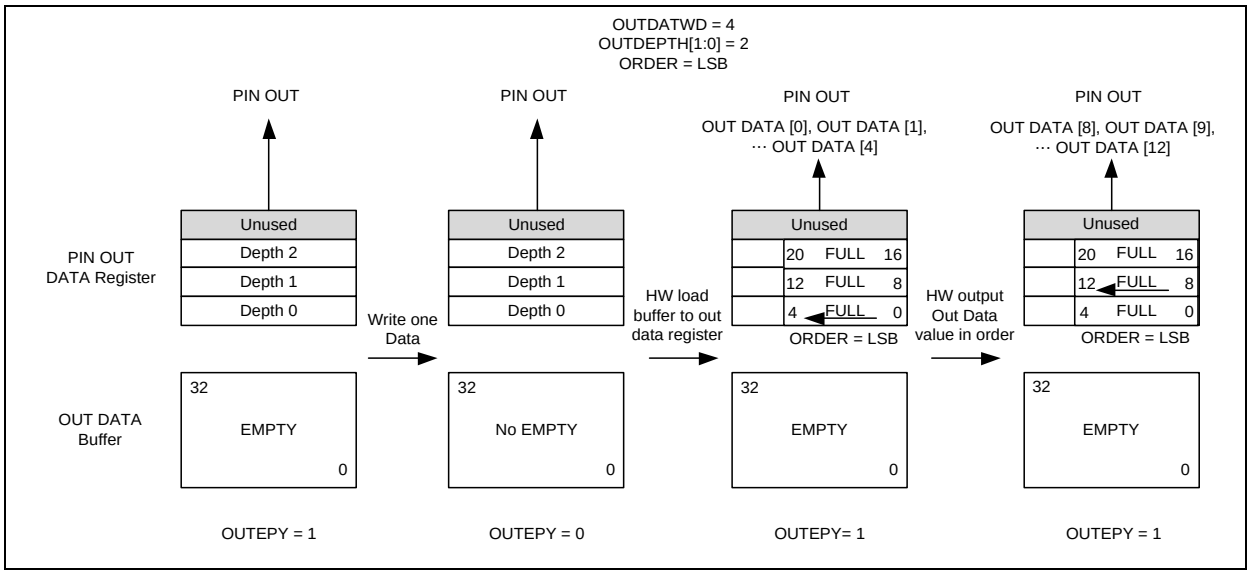


Figure 6.32-13 OUTPUT DATA Buffer Example

When PSIO inputs data, the relation of input data buffer and the full flag is shown in Figure 6.32-14.

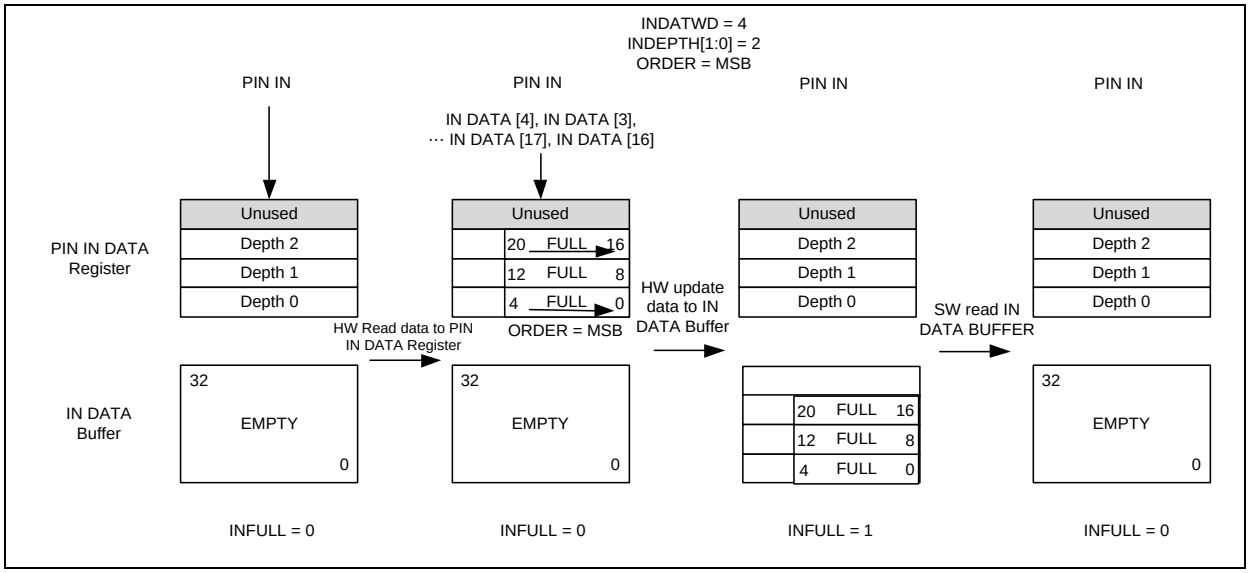


Figure 6.32-14 INPUT DATA Buffer Example

When output data underflow occurs, PSIO will output the data from the start address of output data buffer again.

After the input data overflow occurs, the pin state will not be recorded in the input data buffer and the current input data buffer will be kept.

6.32.6.7 *Interrupt Trigger Condition*

There are four kinds of interrupts in PSIO. The first kind of interrupts is configurable interrupt. User can set interrupt will be trigger at which slot end, and this interrupt trigger is also affected by repeat mode.

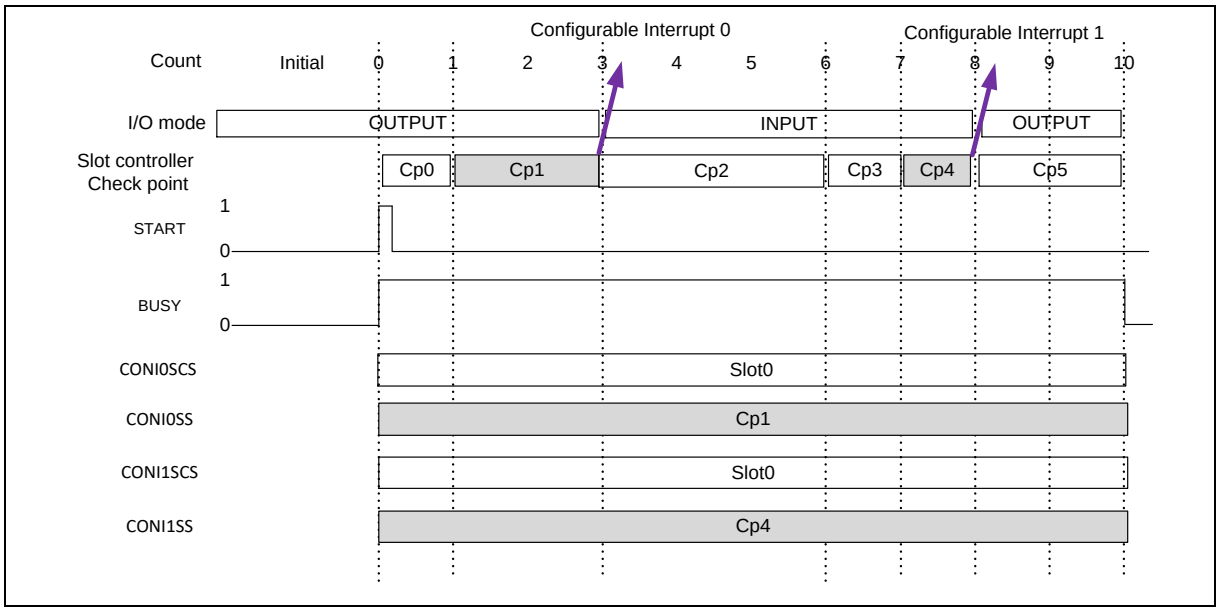


Figure 6.32-15 Configurable Interrupt Flag

The second kind of interrupts is mismatch (MISMATCH), which will be triggered only when using multiple pins transfer data with PDMA, but the date transfer rate between the slots are not consistent.

The third kind of interrupts is transfer error (TERRIF), which will be triggered when there is a transfer error like input data overflow (INOVERx) or output data underflow (OUTUFx)

The forth kind of interrupts is Slot Controller Counting Done(SCxIF), which will be triggered only when the slot controller stop counting.

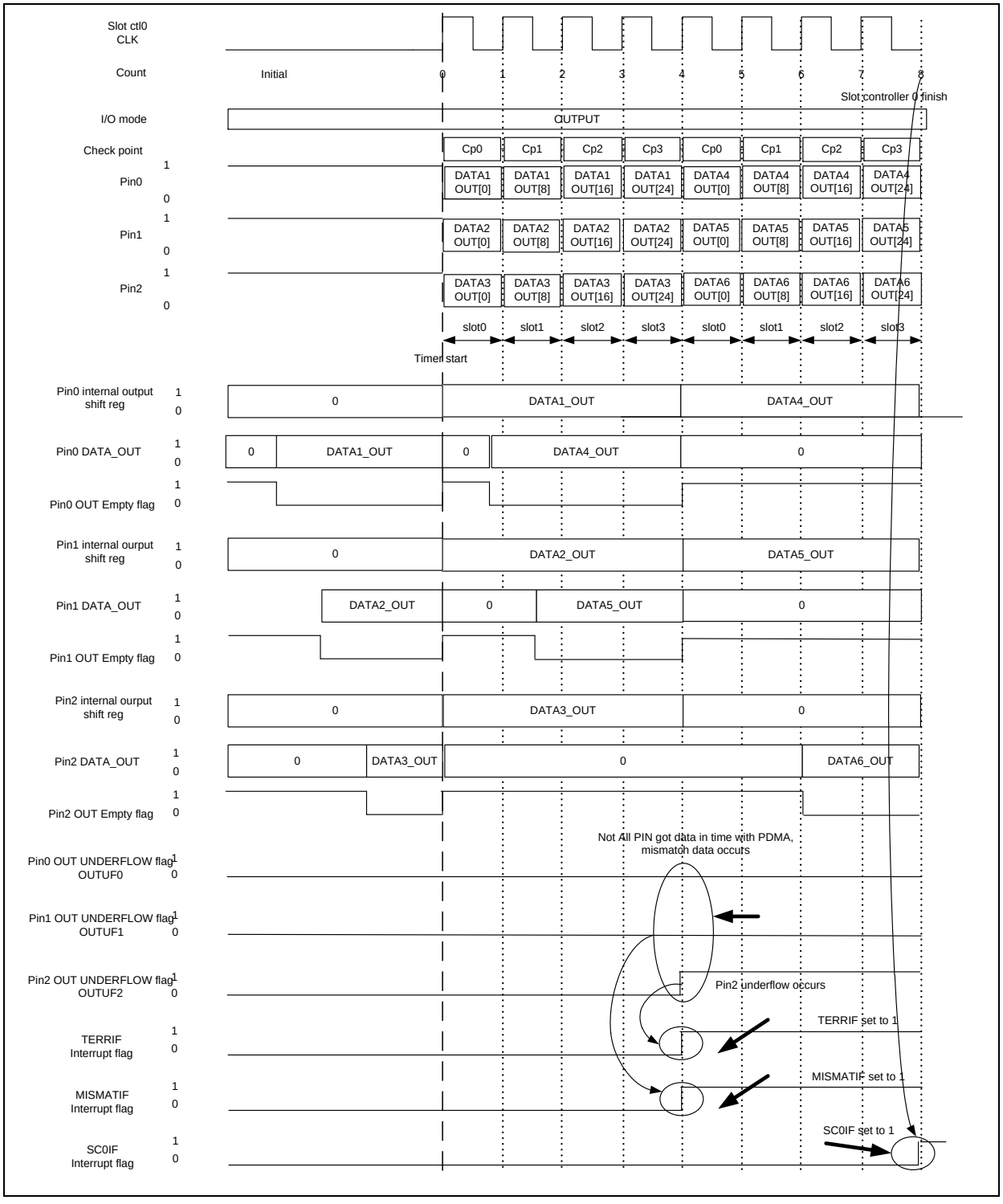


Figure 6.32-16 MISMATIF, TERRIF and SCnIF

6.32.6.8 *Data Transfer with PDMA*

User can use PDMA to increase data transfer rate. Due to the PDMA cannot transfer data to the buffer address roundly, PSIO provide a fixed address, PSIO_PODAT and PSIO_PIDAT, for PDMA to transfer data to data buffer of different pin roundly. After PDMA transfers data to PSIO_PODAT or get data from PSIO_PIDAT, PSIO will move PSIO_PODAT to the coordinate PSIOn_OUTDAT or move the coordinate

PSIOn_INDAT to PSIO_PIDAT. The order of PSIO moving PSIO_PODAT to the coordinate PSIOn_OUTDAT or moving the coordinate PSIOn_INDAT to PSIO_PIDAT is ascending from pin0 to pin7.

When PSIO pin is in PDMA OUTPUT mode, all the enable slots in slot controller of PDMA output mode must link to the related check points in the enable pins of PDMA output mode, or the pin may stop at unexpected slot.

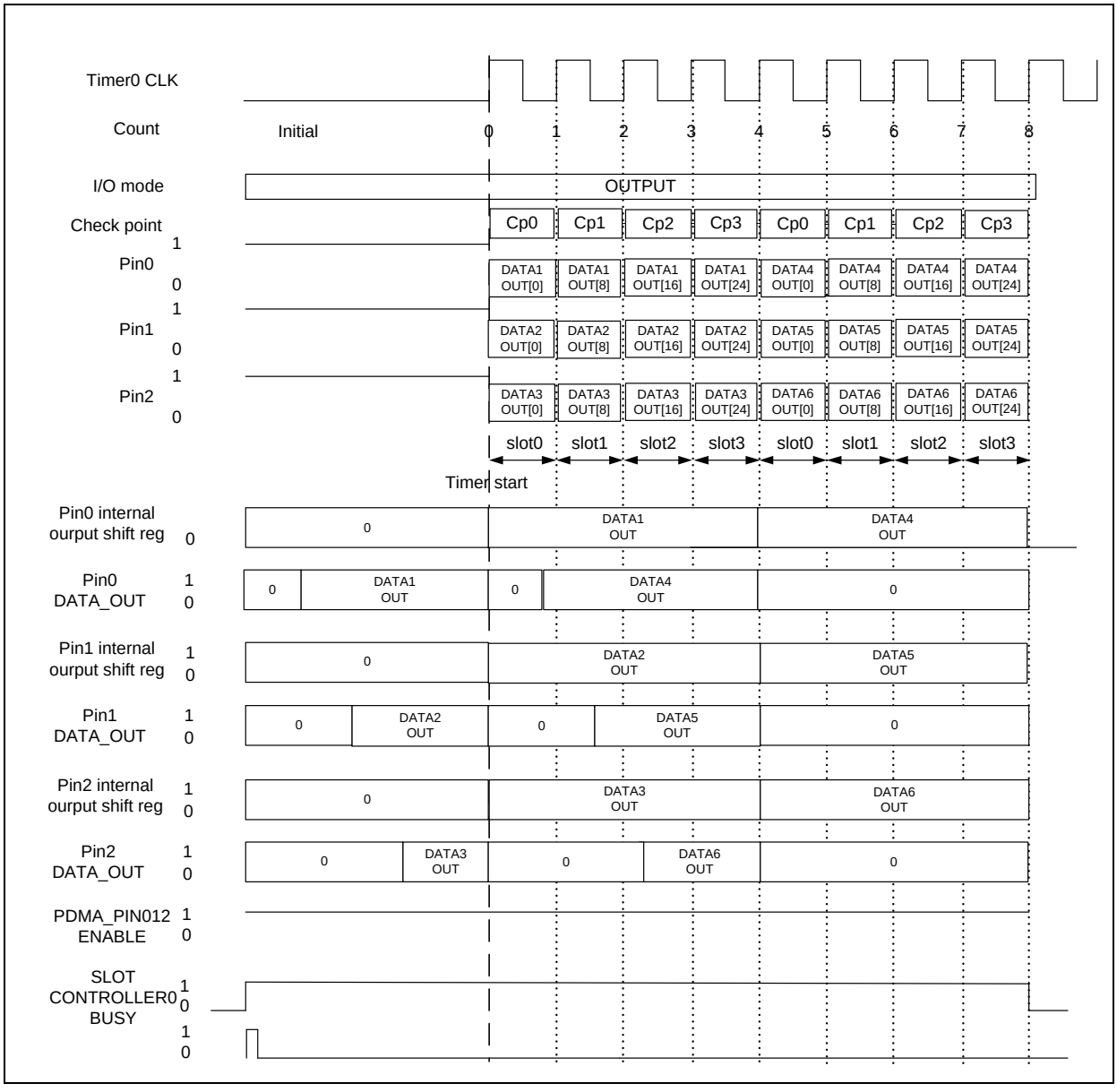


Figure 6.32-17 Data Transfer with PDMA

6.32.7 Programing Flow

6.32.7.1 *Normal Programing*

1. Set count of slot controller slot, repeat mask, and repeat loop count in PSIO_SCnSLOT and PSIO_SCnCTL.
2. Set I/O switch point, I/O initial mode, initial output, interval output, and corresponding slot controller in PSIOn_GENCTL.
3. Set the transmit data in PSIOn_OUTDAT, order, data width and data depth in PSIOn_DATCTL.
4. Set check point and check point act in PSIOn_CPCTL0 and PSIOn_CPCTL1.
5. Set interrupt trigger point at which slot controller and which slot in PSIO_INTCTL.
6. Set slot controller start counting condition at PSIO_SCnCTL, and BUSY(PSIO_SCnCTL) flag will be set to 1 automatically.
7. When slot controller stop counting, BUSY(PSIO_SCnCTL) flag will be cleared to 0 automatically.
8. Check PSIOn_INDAT and PSIOn_INTSTS if there is input data.

6.32.7.2 *Normal Programing Example*

1. Set slot0 to slot5 of slot controller 0 to determine the slot length.
    1) Set the slot 0 length as 1 in SLOT0(PSIO_SC0SLOT[3:0] = 0x1).
    2) Set the slot 1 length as 2 in SLOT1(PSIO_SC0SLOT[7:4] = 0x2).
    3) Set the slot 2 length as 3 in SLOT2(PSIO_SC0SLOT[11:8] = 0x3).
    4) Set the slot 3 length as 1 in SLOT3(PSIO_SC0SLOT[15:12] = 0x1).
    5) Set the slot 4 length as 2 in SLOT4(PSIO_SC0SLOT[19:16] = 0x2).
    6) Set the slot 5 length as 1 in SLOT5(PSIO_SC0SLOT[23:20] = 0x1).
2. Set slot repeat function to determine the slot repeat period.
    1) Set repeat times as 1 in SPLCNT(PSIO_SC0CTL[13:8] = 0x1, Slot Period Loop Count).
    2) Set the initial repeat slot as SLOT3 in INISLOT(PSIO_SC0CTL[3:0] = 0x4, Initial Slot Period).
    3) Set the end repeat slot as SLOT4 in ENDSLOT(PSIO_SC0CTL[7:4] = 0x5, End Slot Period).
3. Set PIN0 general control register(PSIO0_GENCTL).
    1) Set the initial pin state as high in INITIAL(PSIO0_GENCTL[3:2] = 0x1).
    2) Set the interval pin state low in INTERVAL(PSIO0_GENCTL[5:4] = 0x0).
    3) Set the I/O mode will switch at CHECK POINT 1 in SW0CP(PSIO0_GENCTL[11:8] = 0x2).
    4) Set the I/O mode will switch at CHECK POINT 2 in SW1CP(PSIO0_GENCTL[15:12] = 0x3).
    5) Set the I/O mode will switch to INPUT mode at CHECK POINT 1 in MODESW0(PSIO0_GENCTL[17:16] = 0x0).
    6) Set the I/O mode will switch to OUTPUT mode at CHECK POINT 2 in MODESW1(PSIO0_GENCTL[19:18] = 0x4).
    7) Set the initial I/O mode is OUTPUT mode in IOMODE(PSIO0_GENCTL[5:4] = 0x1).

4. Set PIN0 data controller (PSIO0_DATCTL) to determine the data property.
   1) Set the data order is LSB in ORDER(PSIO0_DATCTL[16] = 0x0).
   2) Set the input data width is 8 bit in INDATWD(PSIO0_DATCTL[12:8] = 0x7).
   3) Set the input data depth is 4 in INDEPTH(PSIO0_DATCTL[29:28] = 0x3).
5. Set PIN0 check point controller 0 (PSIO0_CPCTL0) to determine which slot will be connect to the check point.
   1) Connect the slot1 to check point0 by configuring 0x2(SLOT1) to CKPT0(PSIO0_CPCTL0[3:0]).
   2) Connect the slot2 to check point1 by configuring 0x3(SLOT2) to CKPT1(PSIO0_CPCTL0[7:4]).
   3) Connect the slot3 to check point2 by Configuring 0x4(SLOT3) to CKPT2(PSIO0_CPCTL0[11:8]).
6. Set PIN0 check point controller 1 (PSIO0_CP1CTL) to determine the check point action at the check points.
   1) Set PIN0 output low at the check point0 by configuring 0x0(output low) to CKPT0ACT(PSIO0_CPCTL1[2:0]).
   2) Set PIN0 record pin state to input data at the check point1 by configuring 0x4(input data) to CKPT1ACT(PSIO0_CPCTL1[6:4]).
   3) Set PIN0 output high at the check point2 by configuring 0x1(output high) to CKPT2ACT(PSIO0_CPCTL1[10:8]).
7. Set PIN1 general control register(PSIO0_GENCTL).
   1) Set the initial pin state as low in INITIAL(PSIO0_GENCTL[3:2] = 0x0).
   2) Set the interval pin state low in INTERVAL(PSIO0_GENCTL[5:4] = 0x0).
   3) Set the initial I/O mode is OUTPUT mode in IOMODE(PSIO0_GENCTL[5:4] = 0x1).
8. Set PIN1 data controller (PSIO0_DATCTL) to determine the data property.
   1) Set the data order is MSB in ORDER(PSIO0_DATCTL[16] = 0x1).
   2) Set the input data width is 2 bit in INDATWD(PSIO0_DATCTL[12:8] = 0x1).
   3) Set the input data depth is 4 in INDEPTH(PSIO0_DATCTL[29:28] = 0x3).
9. Set PIN1 check point controller 0 (PSIO0_CPCTL0) to determine which slot will be link to the check point.
   1) Connect the slot0 to check point0 by configuring 0x1(SLOT0) to CKPT0(PSIO0_CPCTL0[3:0]).
   2) Connect the slot2 to check point1 by configuring 0x3(SLOT2) to CKPT1(PSIO0_CPCTL0[7:4]).
   3) Connect the slot4 to check point2 by Configuring 0x5(SLOT4) to CKPT2(PSIO0_CPCTL0[11:8]).
10. Set PIN1 check point controller 1 (PSIO0_CP1CTL) to determine the check point action at the check points.
    1) Set PIN1 output high at the check point0 by configuring 0x1(output high) to CKPT0ACT(PSIO0_CPCTL1[2:0]).
    2) Set PIN1 output data from output data buffer at the check point1 by configuring 0x2(output data) to CKPT1ACT(PSIO0_CPCTL1[6:4]).

3) Set PIN1 output high at the check point2 by configuring 0x1(output high) to CKPT2ACT(PSIO0_CPCTL1[10:8]).

11. Set PSIO_INTCTL to determine when the configurable interrupt will be trigger.

1) Select the slot controller 0 as the trigger slot controller of configurable interrupt by Set 0x0 (slot controller 0) to CONI0SCS (PSIO_INTCTL[9:8]).

2) Set the configurable interrupt will be trigger after slot 3 of slot controller 0 by configuring 0x4 (slot 3) to CONI0SS (PSIO_INTCTL[2:0]).

12. Set PSIO_SC0CTL to start the slot controller.

1) Select the trigger source is software trigger by configuring 0 to TRIGSRC(PSIO_SC0CTL[15:14], SC0 Trigger Source).

2) Enable slot controller 0 by set 1 to START(PSIO_SC0CTL[16], SC0 START).

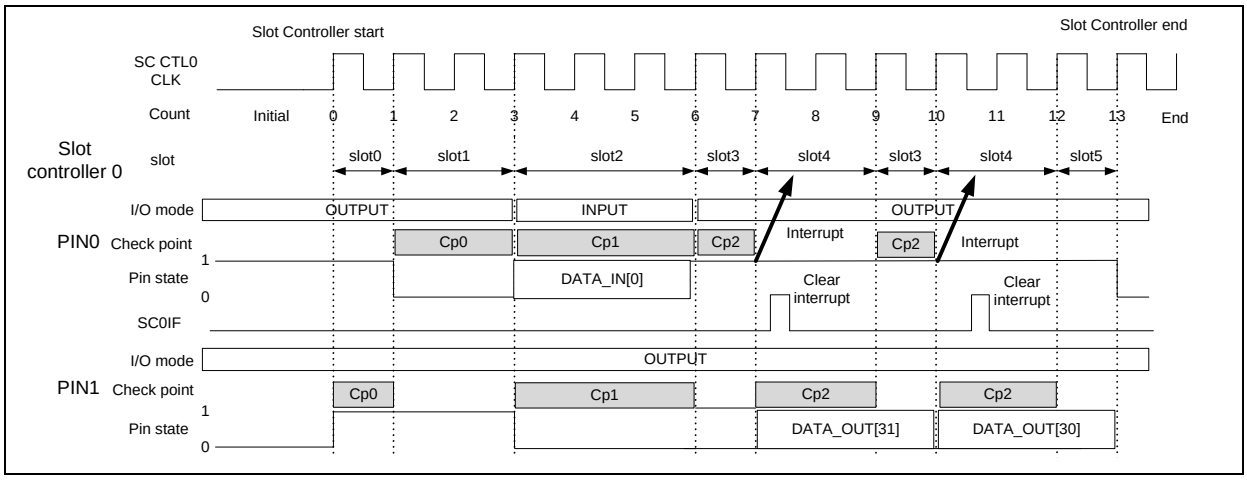


Figure 6.32-18 Normal Programing Example

6.32.7.3 *Programing with PDMA Repeat Mode*

1. Set which pins use PDMA to transfer data in PSIO_PDMACTL.
2. Set count of slot controller slot, repeat slots, and repeat loop count in PSIO_SCnSLOT and PSIO_SCnCTL.
3. Set I/O switch point, I/O initial mode, initial output, final output, and corresponding slot controller in PSIOn_GENCTL.
4. Set the transmit data in PSIOn_OUTDAT, order, data width and data depth in PSIOn_DATCTL.
5. Set check point and check point act in PSIOn_CPCTL0 and PSIOn_CPCTL1.
6. Set interrupt trigger point at which slot controller and which slot in PSIO_INTCTL.
7. Set slot controller start counting condition at PSIO_SCnCTL.
8. When slot controller stop counting, BUSY(PSIO_SCnCTL) flag will be cleared to 0 automatically.
9. Check PSIOn_INTSTS if there is input status, and also can check the specified SRAM address data which are transferred by PDMA.

6.32.7.4 *Programing Example 1 - PDMA with OUTPUT DATA and SLOT CONTROLLER REPEAT mode*

1. Set slot0 to slot3 of slot controller 0 to determine the slot length.

1) Set 1 to SLOT0(PSIO_SC0SLOT[3:0])
2) Set 2 to SLOT1(PSIO_SC0SLOT[7:4])
3) Set 2 to SLOT2(PSIO_SC0SLOT[11:8])
4) Set 2 to SLOT3(PSIO_SC0SLOT[15:12])

2. Set slot repeat function to determine the slot repeat mode.
   1) Set 1 to REPEAT(PSIO_SC0CTL[17], Whole Repeat Mode) to determine the repeat mode.
3. Set PIN0 general control register(PSIO0_GENCTL).
   1) Set 1 to INITIAL(PSIO0_GENCTL[3:2]) to determine the initial pin state.
   2) Set 0 to INTERVAL(PSIO0_GENCTL[5:4]) to determine the interval pin state.
   3) Set 0x1 to IOMODE(PSIO0_GENCTL[5:4]) to determine the initial I/O mode is OUTPUT mode.
4. Set PIN0 data controller (PSIO0_DATCTL) to determine the data property.
   1) Set 0 to ORDER(PSIO0_DATCTL[16]) to determine the data order is LSB.
   2) Set 0x0 to OUTDATWD(PSIO0_DATCTL[4:0]) to determine the output data width is 1 bit.
   3) Set 0x3 to OUTDEPTH(PSIO0_DATCTL[25:24]) to determine the output data depth is 4.
5. Set PIN0 check point controller 0 (PSIO0_CPCTL0) to determine which slot will be link to the check point.
   1) Set 0x1(SLOT0) to CKPT0(PSIO0_CPCTL0[3:0]) to link the slot1 to check point0.
   2) Set 0x2(SLOT1) to CKPT1(PSIO0_CPCTL0[7:4]) to link the slot2 to check point1.
   3) Set 0x3(SLOT2) to CKPT2(PSIO0_CPCTL0[11:8]) to link the slot3 to check point2.
   4) Set 0x4(SLOT3) to CKPT3(PSIO0_CPCTL0[15:12]) to link the slot4 to check point3.
6. Set PIN0 check point controller 1 (PSIO0_CP1CTL) to determine the check point action at the check points.
   1) Set 0x0(output 0) to CKPT0ACT(PSIO0_CPCTL1[2:0]) to output 0 at the check point 0.
   2) Set 0x2(output data) to CKPT1ACT(PSIO0_CPCTL1[6:4]) to output data at the check point1.
   3) Set 0x2(output data) to CKPT2ACT(PSIO0_CPCTL1[10:8]) to output data at the check point2.
   4) Set 0x2(output data) to CKPT3ACT(PSIO0_CPCTL1[14:12]) to output data at the check point3.
7. Set PSIO_INTCTL to determine when the configurable interrupt will be trigger.
   1) Set 0x0 (slot controller 0) to CONI0SCS (PSIO_INTCTL[9:8]) to select the slot controller 0.
   2) Set 0x4 (slot 3) to CONI0SS (PSIO_INTCTL[2:0]) to let the configurable interrupt will be trigger after slot 3 of slot controller 0.
8. Set PDMA related setting to output data from SRAM to PSIO.
   1) Set 0x20000000 to PDMA_DSCTn_SA as source address.
   2) Set PSIO_PODAT address (PSIO_BA + 0x18) to PDMA_DSCTn_DA as detination address.
   3) Set TXCNT(PDMA_DSCTn_CTL[31:16]) = 2 to set PDMA transfer data to PSIO 3 times

4) Set 0x10 to TXWIDTH(PDMA_DSCTn_CTL[13:12]) to select the data width is 32 bit.

5) Set 0x11 to DAINC(PDMA_DSCTn_CTL[11:10]) to select the destination address imcrement is 0.

9. Set PSIO_SC0CTL to start the slot controller.

1) Set 0 to TRIGSRC(PSIO_SC0CTL[15:14], SC0 Trigger Source) to determine the trigger source is by software.

2) Set 1 to START(PSIO_SC0CTL[16], SC0 START) to enable slot controller by software.

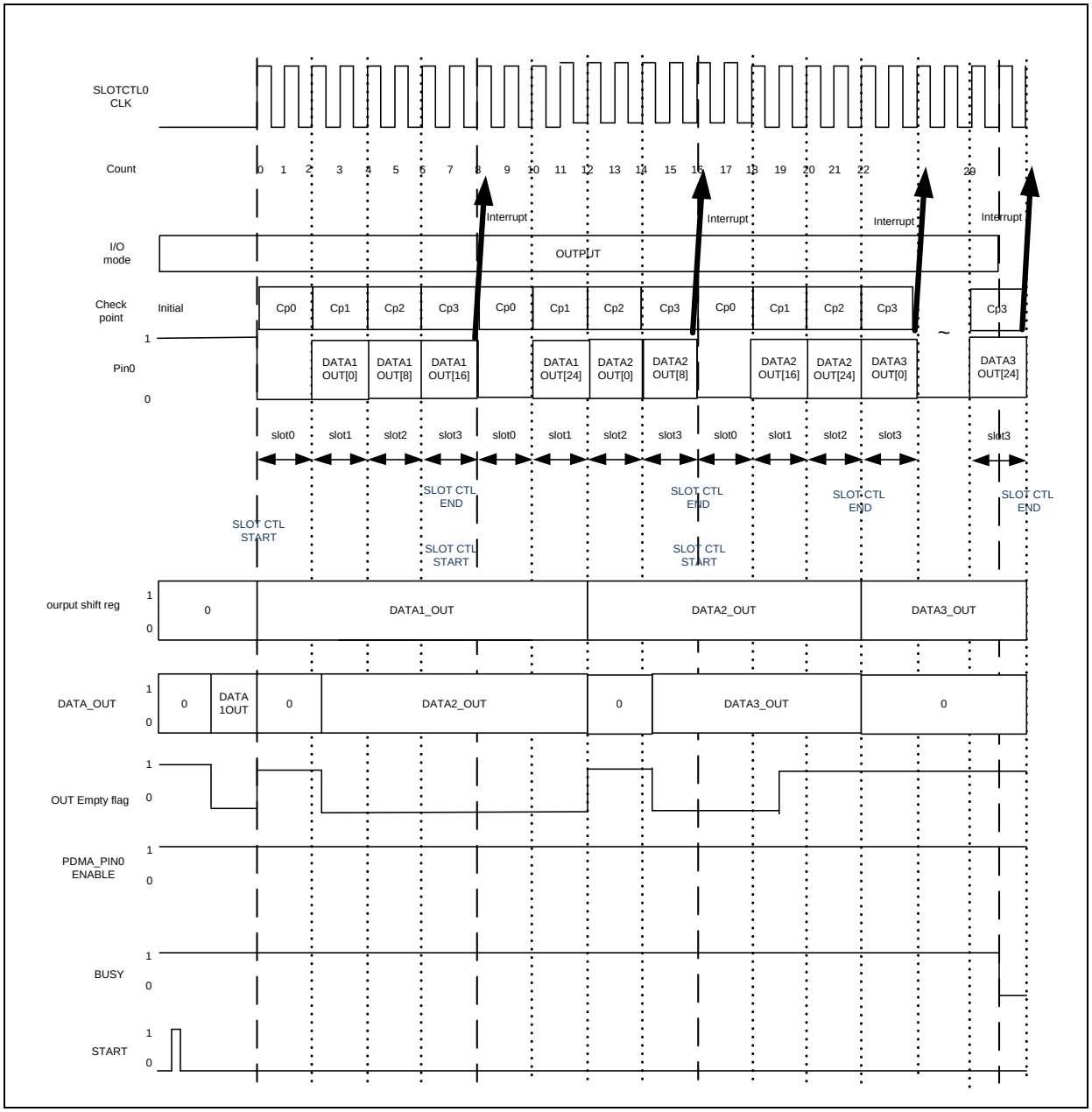


Figure 6.32-19 Programing Example with PDMA and Repeat Mode

6.32.7.5 *Programing Example 2 - PDMA with OUTPUT DATA and Endless SLOT_PERIOD_LOOP_CNT*

1. Set slot0 to slot7 of slot controller 0 to determine the slot length.

1) Set the slot 0 length as 1 in SLOT0(PSIO_SC0SLOT[3:0] = 0x1).
2) Set the slot 1 length as 1 in SLOT1(PSIO_SC0SLOT[7:4] = 0x1).
3) Set the slot 2 length as 1 in SLOT2(PSIO_SC0SLOT[11:8] = 0x1).
4) Set the slot 3 length as 1 in SLOT3(PSIO_SC0SLOT[15:12] = 0x1).
5) Set the slot 4 length as 1 in SLOT4(PSIO_SC0SLOT[19:16] = 0x1).
6) Set the slot 5 length as 1 in SLOT5(PSIO_SC0SLOT[23:20] = 0x1).
7) Set the slot 6 length as 1 in SLOT6(PSIO_SC0SLOT[27:24] = 0x1).
8) Set the slot 7 length as 1 in SLOT7(PSIO_SC0SLOT[31:28] = 0x1).

2. Set slot period loop count to endless and determine initial slot and end slot.
   1) Set 0b111111 to SPLCNT (PSIO_SC0CTL[13:8], Slot Period Loop Count) let slot period with endless loop.
   2) Set 0x04 to INISLOT(PSIO_SC0CTL[3:0], Initial Slot Period) let slot period start from SLOT3.
   3) Set 0x06 ENDSLOT(PSIO_SC0CTL[7:4], End Slot Period) let slot period start end at SLOT5.
3. Set PIN0 general control register(PSIO0_GENCTL).
   1) Set 1 to INITIAL(PSIO0_GENCTL[3:2]) to determine the initial pin state.
   2) Set 0 to INTERVAL(PSIO0_GENCTL[5:4]) to determine the interval pin state.
   3) Set 0x1 to IOMODE(PSIO0_GENCTL[5:4]) to determine the initial I/O mode is OUTPUT mode.
4. Set PIN0 data controller (PSIO0_DATCTL) to determine the data property.
   1) Set 0 to ORDER(PSIO0_DATCTL[16]) to determine the data order is LSB.
   2) Set 0x0 to OUTDATWD(PSIO0_DATCTL[4:0]) to determine the output data width is 1 bit.
   3) Set 0x3 to OUTDEPTH(PSIO0_DATCTL[25:24]) to determine the output data depth is 4.
5. Set PIN0 check point controller 0 (PSIO0_CPCTL0) to determine which slot will be link to the check point.
   1) Set 0x1(SLOT0) to CKPT0(PSIO0_CPCTL0[3:0]) to link the slot0 to check point0.
   2) Set 0x2(SLOT1) to CKPT1(PSIO0_CPCTL0[7:4]) to link the slot1 to check point1.
   3) Set 0x3(SLOT2) to CKPT2(PSIO0_CPCTL0[11:8]) to link the slot2 to check point2.
   4) Set 0x4(SLOT3) to CKPT3(PSIO0_CPCTL0[15:12]) to link the slot3 to check point3.
   5) Set 0x5(SLOT4) to CKPT4(PSIO0_CPCTL0[19:16]) to link the slot4 to check point4.
   6) Set 0x6(SLOT5) to CKPT5(PSIO0_CPCTL0[23:20]) to link the slot5 to check point5.
   7) Set 0x7(SLOT6) to CKPT6(PSIO0_CPCTL0[27:24]) to link the slot6 to check point6.
   8) Set 0x8(SLOT7) to CKPT7(PSIO0_CPCTL0[31:28]) to link the slot7 to check point7.
6. Set PIN0 check point controller 1 (PSIO0_CP1CTL) to determine the check point action at the check points.
   1) Set 0x0(output 0) to CKPT0ACT(PSIO0_CPCTL1[2:0]) to output 0 at the check point 0.
   2) Set 0x1(output 1) to CKPT1ACT(PSIO0_CPCTL1[6:4]) to output 1 at the check point1.
   3) Set 0x0(output 0) to CKPT2ACT(PSIO0_CPCTL1[10:8]) to output 0 at the check point2.

4) Set 0x2(output data) to CKPT3ACT(PSIO0_CPCTL1[14:12]) to output data at the check point3.

5) Set 0x2(output data) to CKPT4ACT(PSIO0_CPCTL1[18:16]) to output data at the check point4.

6) Set 0x2(output data) to CKPT5ACT(PSIO0_CPCTL1[22:20]) to output data at the check point5.

7. Set PDMA related setting to output data from SRAM to PSIO.

   1) Set 0x20000000 to PDMA_DSCTn_SA as source address.

   2) Set PSIO_PODAT address (PSIO_BA + 0x18) to PDMA_DSCTn_DA as detination address.

   3) Set TXCNT(PDMA_DSCTn_CTL[31:16]) = 2 to set PDMA transfer data to PSIO 3 times

   4) Set 0x10 to TXWIDTH(PDMA_DSCTn_CTL[13:12]) to select the data width is 32 bit.

   5) Set 0x11 to DAINC(PDMA_DSCTn_CTL[11:10]) to select the destination address imcrement is 0.

8. Set PSIO_SC0CTL to start the slot controller.

   1) Set 0 to TRIGSRC(PSIO_SC0CTL[15:14], SC0 Trigger Source) to determine the trigger source is by software.

   2) Set 1 to START(PSIO_SC0CTL[16], SC0 START) to enable slot controller by software.

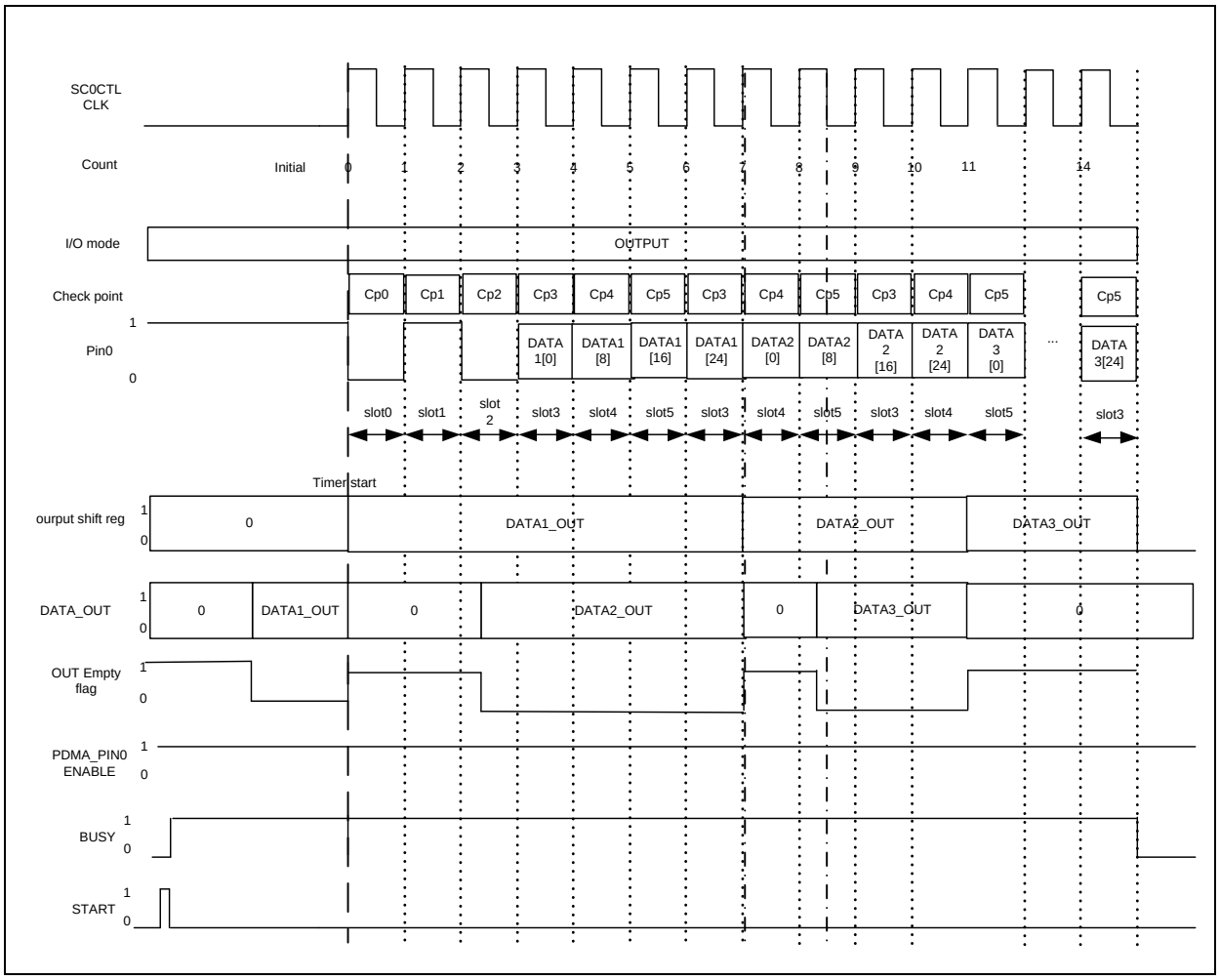


Figure 6.32-20 PDMA with OUTPUT DATA and Endless SLOT_PERIOD_LOOP_CNT

6.32.7.6 *Programing Example 3 - PDMA with OUTPUT DATA and SLOT CONTROLLER WHOLE REPEAT mode with 3 pins*

1. Set slot0 to slot3 of slot controller 0 to determine the slot length.
   1) Set the slot 0 length as 1 in SLOT0(PSIO_SC0SLOT[3:0] = 0x1).
   2) Set the slot 1 length as 1 in SLOT1(PSIO_SC0SLOT[7:4] = 0x1).
   3) Set the slot 2 length as 1 in SLOT2(PSIO_SC0SLOT[11:8] = 0x1).
   4) Set the slot 3 length as 1 in SLOT3(PSIO_SC0SLOT[15:12] = 0x1).
2. Set slot0 to slot3 of slot controller 1 to determine the slot length.
   1) Set the slot 0 length as 1 in SLOT0(PSIO_SC1SLOT[3:0] = 0x1).
   2) Set the slot 1 length as 1 in SLOT1(PSIO_SC1SLOT[7:4] = 0x1).
   3) Set the slot 2 length as 1 in SLOT2(PSIO_SC1SLOT[11:8] = 0x1).
   4) Set the slot 3 length as 1 in SLOT3(PSIO_SC1SLOT[15:12] = 0x1).
3. Set slot0 to slot3 of slot controller 2 to determine the slot length.
   1) Set the slot 0 length as 1 in SLOT0(PSIO_SC2SLOT[3:0] = 0x1).

2) Set the slot 1 length as 1 in SLOT1(PSIO_SC2SLOT[7:4] = 0x1).
3) Set the slot 2 length as 1 in SLOT2(PSIO_SC2SLOT[11:8] = 0x1).
4) Set the slot 3 length as 1 in SLOT3(PSIO_SC2SLOT[15:12] = 0x1).

4. Set slot repeat function of slot controller to determine the slot repeat mode.
   1) Set 1 to REPEAT(PSIO_SC0CTL[17], Whole Repeat Mode) to determine the repeat mode of slot controller 0.
   2) Set 1 to REPEAT(PSIO_SC1CTL[17], Whole Repeat Mode) to determine the repeat mode of slot controller 1.
   3) Set 1 to REPEAT(PSIO_SC2CTL[17], Whole Repeat Mode) to determine the repeat mode. of slot controller 2.
5. Set PIN0 general control register(PSIO0_GENCTL).
   1) Set 1 to INITIAL(PSIO0_GENCTL[3:2]) to determine the initial pin state.
   2) Set 0 to INTERVAL(PSIO0_GENCTL[5:4]) to determine the interval pin state.
   3) Set 0x1 to IOMODE(PSIO0_GENCTL[5:4]) to determine the initial I/O mode is OUTPUT mode.
6. Set PIN1 general control register(PSIO1_GENCTL).
   1) Set 1 to INITIAL(PSIO1_GENCTL[3:2]) to determine the initial pin state.
   2) Set 0 to INTERVAL(PSIO1_GENCTL[5:4]) to determine the interval pin state.
   3) Set 0x1 to IOMODE(PSIO1_GENCTL[5:4]) to determine the initial I/O mode is OUTPUT mode.
7. Set PIN2 general control register(PSIO2_GENCTL).
   1) Set 1 to INITIAL(PSIO2_GENCTL[3:2]) to determine the initial pin state.
   2) Set 0 to INTERVAL(PSIO2_GENCTL[5:4]) to determine the interval pin state.
   3) Set 0x1 to IOMODE(PSIO2_GENCTL[5:4]) to determine the initial I/O mode is OUTPUT mode.
8. Set PIN0 data controller (PSIO0_DATCTL) to determine the data property.
   1) Set 0 to ORDER(PSIO0_DATCTL[16]) to determine the data order is LSB.
   2) Set 0x0 to OUTDATWD(PSIO0_DATCTL[4:0]) to determine the output data width is 1 bit.
   3) Set 0x3 to OUTDEPTH(PSIO0_DATCTL[25:24]) to determine the output data depth is 4.
9. Set PIN1 data controller (PSIO1_DATCTL) to determine the data property.
   1) Set 0 to ORDER(PSIO1_DATCTL[16]) to determine the data order is LSB.
   2) Set 0x0 to OUTDATWD(PSIO1_DATCTL[4:0]) to determine the output data width is 1 bit.
   3) Set 0x3 to OUTDEPTH(PSIO1_DATCTL[25:24]) to determine the output data depth is 4.
10. Set PIN2 data controller (PSIO2_DATCTL) to determine the data property.
    1) Set 0 to ORDER(PSIO2_DATCTL[16]) to determine the data order is LSB.
    2) Set 0x0 to OUTDATWD(PSIO2_DATCTL[4:0]) to determine the output data width is 1 bit.
    3) Set 0x3 to OUTDEPTH(PSIO2_DATCTL[25:24]) to determine the output data depth is 4.
11. Set PIN0 check point controller 0 (PSIO0_CPCTL0) to determine which slot will be link to the check point.
    1) Set 0x1(SLOT0) to CKPT0(PSIO0_CPCTL0[3:0]) to link the slot0 to check point0.

2) Set 0x2(SLOT1) to CKPT1(PSIO0_CPCTL0[7:4]) to link the slot1 to check point1.
3) Set 0x3(SLOT2) to CKPT2(PSIO0_CPCTL0[11:8]) to link the slot2 to check point2.
4) Set 0x4(SLOT3) to CKPT3(PSIO0_CPCTL0[15:12]) to link the slot3 to check point3.

12. Set PIN1 check point controller 0 (PSIO1_CPCTL0) to determine which slot will be link to the check point.
    1) Set 0x1(SLOT0) to CKPT0(PSIO1_CPCTL0[3:0]) to link the slot0 to check point0.
    2) Set 0x2(SLOT1) to CKPT1(PSIO1_CPCTL0[7:4]) to link the slot1 to check point1.
    3) Set 0x3(SLOT2) to CKPT2(PSIO1_CPCTL0[11:8]) to link the slot2 to check point2.
    4) Set 0x4(SLOT3) to CKPT3(PSIO1_CPCTL0[15:12]) to link the slot3 to check point3.
13. Set PIN2 check point controller 0 (PSIO2_CPCTL0) to determine which slot will be link to the check point.
    1) Set 0x1(SLOT0) to CKPT0(PSIO2_CPCTL0[3:0]) to link the slot0 to check point0.
    2) Set 0x2(SLOT1) to CKPT1(PSIO2_CPCTL0[7:4]) to link the slot1 to check point1.
    3) Set 0x3(SLOT2) to CKPT2(PSIO2_CPCTL0[11:8]) to link the slot2 to check point2.
    4) Set 0x4(SLOT3) to CKPT3(PSIO2_CPCTL0[15:12]) to link the slot3 to check point3.
14. Set PIN0 check point controller 1 (PSIO0_CP1CTL) to determine the check point action at the check points.
    1) Set 0x2(output data) to CKPT0ACT(PSIO0_CPCTL1[2:0]) to output data at the check point 0.
    2) Set 0x2(output data) to CKPT1ACT(PSIO0_CPCTL1[6:4]) to output data at the check point1.
    3) Set 0x2(output data) to CKPT2ACT(PSIO0_CPCTL1[10:8]) to output data at the check point2.
    4) Set 0x2(output data) to CKPT3ACT(PSIO0_CPCTL1[14:12]) to output data at the check point3.
15. Set PIN1 check point controller 1 (PSIO1_CP1CTL) to determine the check point action at the check points.
    1) Set 0x2(output data) to CKPT0ACT(PSIO1_CPCTL1[2:0]) to output data at the check point 0.
    2) Set 0x2(output data) to CKPT1ACT(PSIO1_CPCTL1[6:4]) to output data at the check point1.
    3) Set 0x2(output data) to CKPT2ACT(PSIO1_CPCTL1[10:8]) to output data at the check point2.
    4) Set 0x2(output data) to CKPT3ACT(PSIO1_CPCTL1[14:12]) to output data at the check point3.
16. Set PIN2 check point controller 1 (PSIO2_CP1CTL) to determine the check point action at the check points.
    1) Set 0x2(output data) to CKPT0ACT(PSIO2_CPCTL1[2:0]) to output data at the check point 0.
    2) Set 0x2(output data) to CKPT1ACT(PSIO2_CPCTL1[6:4]) to output data at the check point1.
    3) Set 0x2(output data) to CKPT2ACT(PSIO2_CPCTL1[10:8]) to output data at the check point2.

4) Set 0x2(output data) to CKPT3ACT(PSIO2_CPCTL1[14:12]) to output data at the check point3.

17. Set PDMA related setting to output data from SRAM to PSIO.
    1) Set 0x20000000 to PDMA_DSCTn_SA as source address.
    2) Set PSIO_PODAT address (PSIO_BA + 0x18) to PDMA_DSCTn_DA as destination address.
    3) Set TXCNT(PDMA_DSCTn_CTL[31:16]) = 5 to set PDMA transfer data to PSIO 6 times
    4) Set 0x10 to TXWIDTH(PDMA_DSCTn_CTL[13:12]) to select the data width is 32 bit.
    5) Set 0x11 to DAINC(PDMA_DSCTn_CTL[11:10]) to select the destination address imcrement is 0.
18. Set PSIO_SC0CTL to start the slot controller.
    1) Set 0 to TRIGSRC(PSIO_SC0CTL[15:14], SC0 Trigger Source) to determine the trigger source is by software.
    2) Set 1 to START(PSIO_SC0CTL[16], SC0 START) to enable slot controller by software.

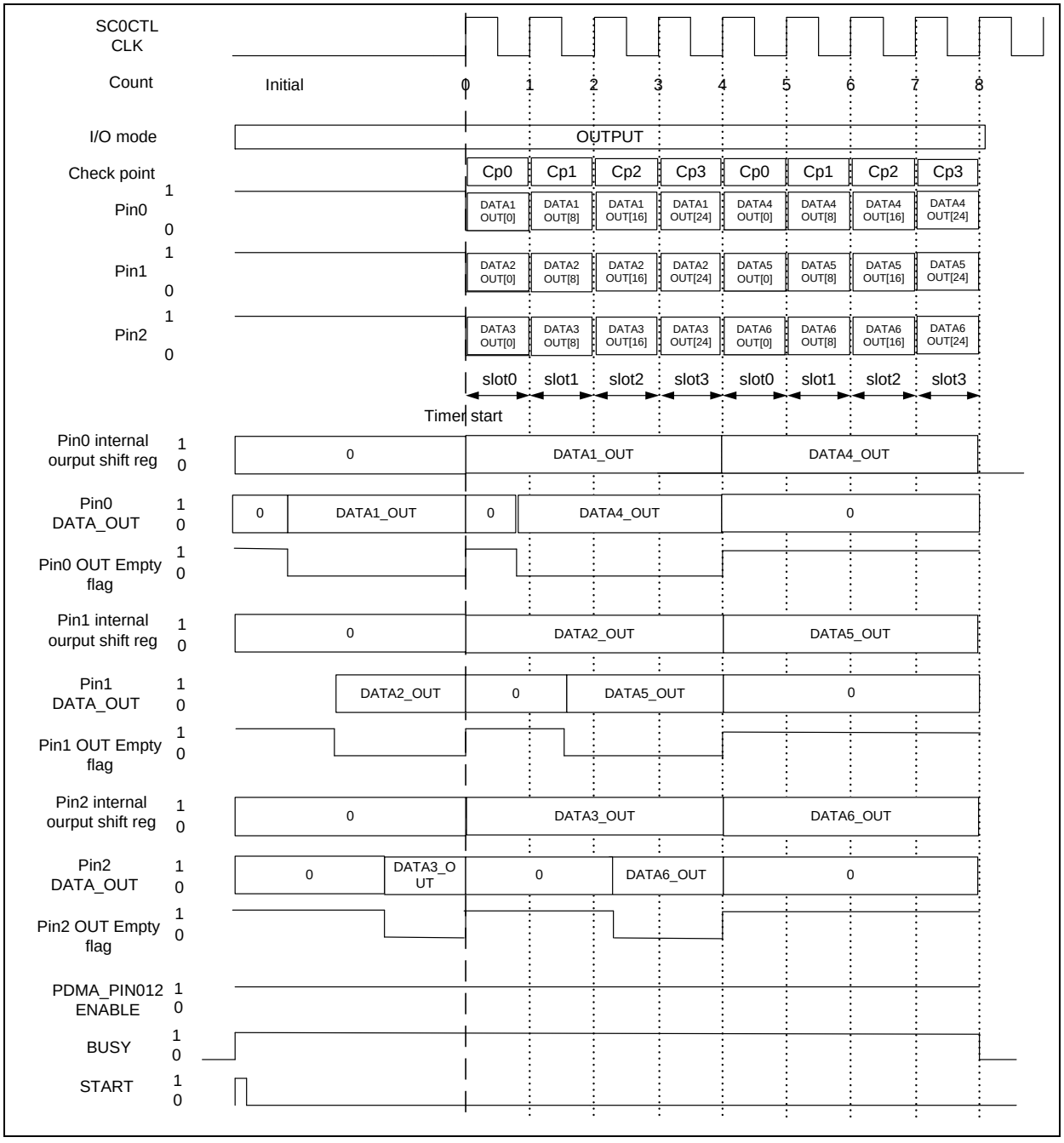


Figure 6.32-21 PDMA with OUTPUT DATA and SLOT CONTROLLER WHOLE REPEAT Mode with 3 Pins

6.32.7.7 *Programing Example 4: PDMA with INPUT DATA and SLOT CONTROLLER WHOLE REPEAT mode*

1. Set slot0 to slot3 of slot controller 0 to determine the slot length.
   1) Set 1 to SLOT0(PSIO_SC0SLOT[3:0])
   2) Set 1 to SLOT1(PSIO_SC0SLOT[7:4])
   3) Set 1 to SLOT2(PSIO_SC0SLOT[11:8])
   4) Set 1 to SLOT3(PSIO_SC0SLOT[15:12])

2. Set slot repeat function to determine the slot repeat mode.
    1) Set 1 to REPEAT(PSIO_SC0CTL[17], Whole Repeat Mode) to determine the repeat mode.
3. Set PIN0 general control register(PSIO0_GENCTL).
    1) Set 0x00 to IOMODE(PSIO0_GENCTL[5:4]) to determine the initial I/O mode is INPUT mode.
    2) Set 0 to SCSEL(PSIO0_GENCTL[25:24])
4. Set PIN0 data controller (PSIO0_DATCTL) to determine the data property.
    1) Set 0 to ORDER(PSIO0_DATCTL[16]) to determine the data order is LSB.
    2) Set 0x0 to INDATWD(PSIO0_DATCTL[12:8]) to determine the input data width is 1 bit.
    3) Set 0x3 to INDEPTH(PSIO0_DATCTL[29:28]) to determine the input data depth is 4.
5. Set PIN0 check point controller 0 (PSIO0_CPCTL0) to determine which slot will be link to the check point.
    1) Set 0x1(SLOT0) to CKPT0(PSIO0_CPCTL0[3:0]) to link the slot0 to check point0.
    2) Set 0x2(SLOT1) to CKPT1(PSIO0_CPCTL0[7:4]) to link the slot1 to check point1.
    3) Set 0x3(SLOT2) to CKPT2(PSIO0_CPCTL0[11:8]) to link the slot2 to check point2.
    4) Set 0x4(SLOT3) to CKPT3(PSIO0_CPCTL0[15:12]) to link the slot3 to check point3.
6. Set PIN0 check point controller 1 (PSIO0_CP1CTL) to determine the check point action at the check points.
    1) Set 0x5 (input data) to CKPT0ACT(PSIO0_CPCTL1[2:0]) to input data at the check point 0.
    2) Set 0x5(input data) to CKPT1ACT(PSIO0_CPCTL1[6:4]) to input data at the check point1.
    3) Set 0x5 (input data) to CKPT2ACT(PSIO0_CPCTL1[10:8]) to input data at the check point2.
    4) Set 0x5 (input data) to CKPT3ACT(PSIO0_CPCTL1[14:12]) to input data at the check point3.
7. Set PDMA related setting to receive data from PSIO to SRAM.
    1) Set PSIO_PIDAT address (PSIO_BA + 0x1C) to PDMA_DSCTn_SA as source address.
    2) Set 0x20000000 to PDMA_DSCTn_DA as destination address.
    3) Set TXCNT(PDMA_DSCTn_CTL[31:16]) = 2 to set PDMA transfer data from PSIO 3 times.
    4) Set 0x10 to TXWIDTH(PDMA_DSCTn_CTL[13:12]) to select the data width is 32 bit.
    5) Set 0x11 to SAINC(PDMA_DSCTn_CTL[9:8]) to select the source address imcrement is 0.
8. Set 0x1 to TRIGSRC(PSIO_SC0CTL[15:14], SC0 Trigger Source) to determine the trigger source is by falling edge.

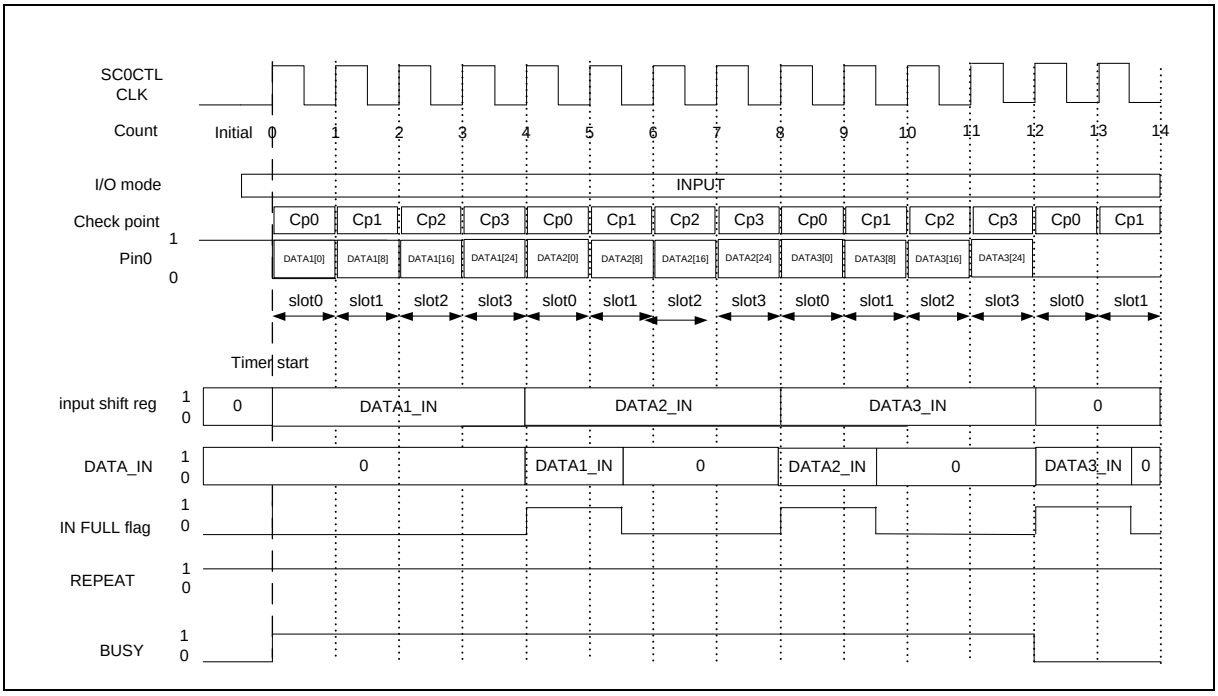


Figure 6.32-22 PDMA with INPUT DATA and SLOT CONTROLLER WHOLE REPEAT Mode

6.32.7.8 *Programing Example 5: PDMA with INPUT DATA and Endless SLOT_PERIOD_LOOP_CNT*

1. Set slot0 to slot7 of slot controller 0 to determine the slot length.
   1) Set the slot 0 length as 1 in SLOT0(PSIO_SC0SLOT[3:0] = 0x1).
   2) Set the slot 1 length as 1 in SLOT1(PSIO_SC0SLOT[7:4] = 0x1).
   3) Set the slot 2 length as 1 in SLOT2(PSIO_SC0SLOT[11:8] = 0x1).
   4) Set the slot 3 length as 1 in SLOT3(PSIO_SC0SLOT[15:12] = 0x1).
   5) Set the slot 4 length as 1 in SLOT4(PSIO_SC0SLOT[19:16] = 0x1).
2. Set slot period loop count to endless and determine initial slot and end slot.
   1) Set 0b111111 to SPLCNT (PSIO_SC0CTL[13:8], Slot Period Loop Count) let slot period with endless loop.
   2) Set 0b0010(SLOT1) to INISLOT(PSIO_SC0CTL[3:0], Initial Slot Period) let slot period start from SLOT1.
   3) Set 0b0101(SLOT) to ENDSLOT(PSIO_SC0CTL[7:4], End Slot Period) let slot period start end at SLOT4.
3. Set PIN0 general control register(PSIO0_GENCTL).
   1) Set 0x00 to IOMODE(PSIO0_GENCTL[5:4]) to determine the initial I/O mode is INPUT mode.
   2) Set 0 to SCSEL(PSIO0_GENCTL[25:24])
4. Set PIN0 data controller (PSIO0_DATCTL) to determine the data property.
   1) Set 0 to ORDER(PSIO0_DATCTL[16]) to determine the data order is LSB.
   2) Set 0x0 to INDATWD(PSIO0_DATCTL[12:8]) to determine the input data width is 1 bit.
   3) Set 0x3 to INDEPTH(PSIO0_DATCTL[29:28]) to determine the input data depth is 4.

5. Set PIN0 check point controller 0 (PSIO0_CPCTL0) to determine which slot will be link to the check point.
   1) Set 0x1(SLOT0) to CKPT0(PSIO0_CPCTL0[3:0]) to link the slot0 to check point0.
   2) Set 0x2(SLOT1) to CKPT1(PSIO0_CPCTL0[7:4]) to link the slot1 to check point1.
   3) Set 0x3(SLOT2) to CKPT2(PSIO0_CPCTL0[11:8]) to link the slot2 to check point2.
   4) Set 0x4(SLOT3) to CKPT3(PSIO0_CPCTL0[15:12]) to link the slot3 to check point3.
   5) Set 0x5(SLOT4) to CKPT4(PSIO0_CPCTL0[19:16]) to link the slot4 to check point4.
6. Set PIN0 check point controller 1 (PSIO0_CP1CTL) to determine the check point action at the check points.
   1) Set 0b101(input status) to CKPT0ACT(PSIO0_CPCTL1[2:0]) to input status at the check point 0.
   2) Set 0b100(input data) to CKPT1ACT(PSIO0_CPCTL1[6:4]) to input data at the check point1.
   3) Set 0b100(input data) to CKPT2ACT(PSIO0_CPCTL1[10:8]) to input data at the check point2.
   4) Set 0b100 (input data) to CKPT3ACT(PSIO0_CPCTL1[14:12]) to input data at the check point3.
   5) Set 0b100 (input data) to CKPT4ACT(PSIO0_CPCTL1[18:16]) to input data at the check point4.
7. Set PDMA related setting to receive data from PSIO to SRAM.
   1) Set PSIO_PIDAT address (PSIO_BA + 0x1C) to PDMA_DSCTn_SA as source address.
   2) Set 0x20000000 to PDMA_DSCTn_DA as destination address.
   3) Set TXCNT(PDMA_DSCTn_CTL[31:16]) = 1 to set PDMA transfer data from PSIO 2 times.
   4) Set 0x10 to TXWIDTH(PDMA_DSCTn_CTL[13:12]) to select the data width is 32 bit.
   5) Set 0x11 to SAINC(PDMA_DSCTn_CTL[9:8]) to select the source address increment is 0.
8. Set PSIO_SC0CTL to start the slot controller.
   1) Set 0x1 to TRIGSRC(PSIO_SC0CTL[15:14], SC0 Trigger Source) to determine the trigger source is by falling edge.

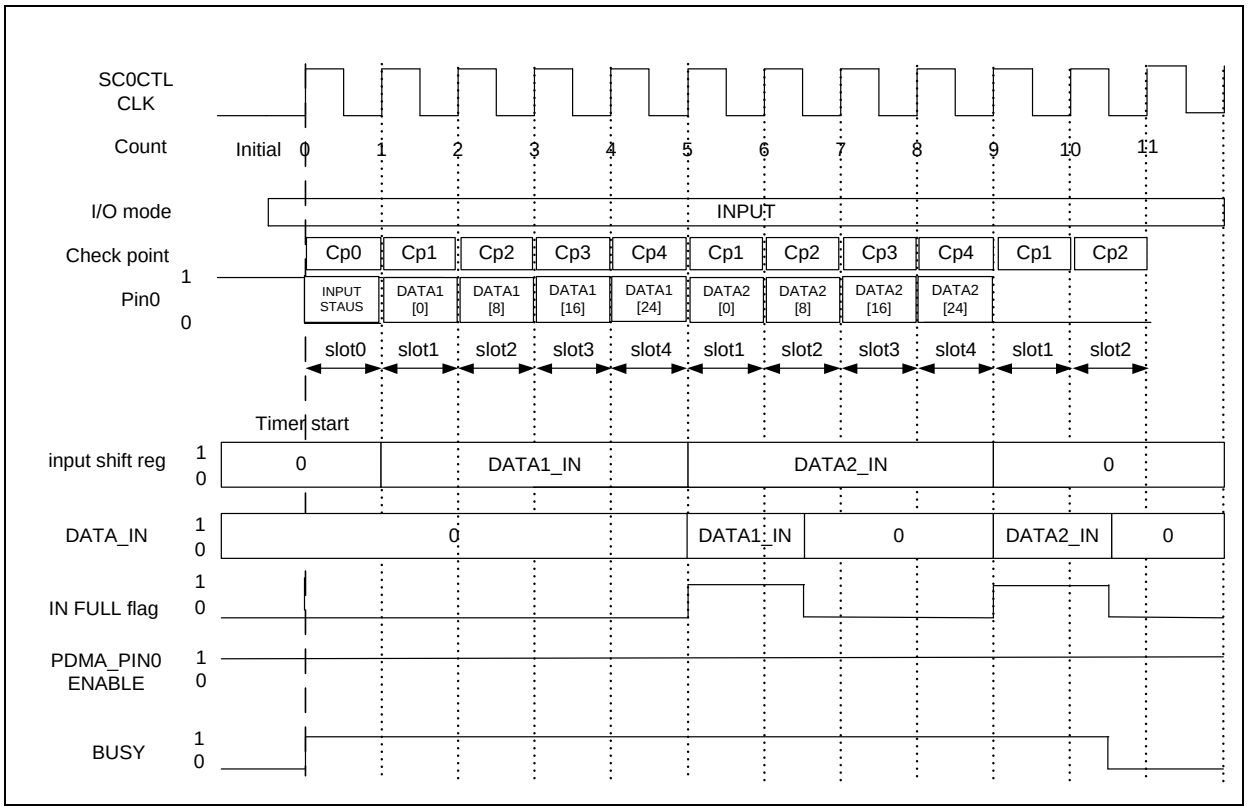


Figure 6.32-23 PDMA with INPUT DATA and Endless SLOT_PERIOD_LOOP_CNT

6.32.7.9 *Programing Example 6: PDMA with INPUT DATA and SLOT CONTROLLER WHOLE REPEAT mode with 2 pins*

1. Set slot0 to slot3 of slot controller 0 to determine the slot length.
   1) Set the slot 0 length as 1 in SLOT0(PSIO_SC0SLOT[3:0] = 0x1).
   2) Set the slot 1 length as 1 in SLOT1(PSIO_SC0SLOT[7:4] = 0x1).
   3) Set the slot 2 length as 1 in SLOT2(PSIO_SC0SLOT[11:8] = 0x1).
   4) Set the slot 3 length as 1 in SLOT3(PSIO_SC0SLOT[15:12] = 0x1).
2. Set slot0 to slot3 of slot controller 1 to determine the slot length.
   1) Set the slot 0 length as 1 in SLOT0(PSIO_SC1SLOT[3:0] = 0x1).
   2) Set the slot 1 length as 1 in SLOT1(PSIO_SC1SLOT[7:4] = 0x1).
   3) Set the slot 2 length as 1 in SLOT2(PSIO_SC1SLOT[11:8] = 0x1).
   4) Set the slot 3 length as 1 in SLOT3(PSIO_SC1SLOT[15:12] = 0x1).
3. Set slot0 to slot3 of slot controller 2 to determine the slot length.
   1) Set the slot 0 length as 1 in SLOT0(PSIO_SC2SLOT[3:0] = 0x1).
   2) Set the slot 1 length as 1 in SLOT1(PSIO_SC2SLOT[7:4] = 0x1).
   3) Set the slot 2 length as 1 in SLOT2(PSIO_SC2SLOT[11:8] = 0x1).
   4) Set the slot 3 length as 1 in SLOT3(PSIO_SC2SLOT[15:12] = 0x1).
4. Set slot repeat function of slot controller to determine the slot repeat mode.

1) Set 1 to REPEAT(PSIO_SC0CTL[17], Whole Repeat Mode) to determine the repeat mode of slot controller 0.
2) Set 1 to REPEAT(PSIO_SC1CTL[17], Whole Repeat Mode) to determine the repeat mode of slot controller 1.
3) Set 1 to REPEAT(PSIO_SC2CTL[17], Whole Repeat Mode) to determine the repeat mode. of slot controller 2.

5. Set PIN0 general control register(PSIO0_GENCTL).
   1) Set 0b10 to IOMODE(PSIO0_GENCTL[5:4]) to determine the initial I/O mode is INPUT mode.
   2) Set 0 to SCSEL(PSIO0_GENCTL[25:24])
6. Set PIN1 general control register(PSIO1_GENCTL).
   1) Set 0b10 to IOMODE(PSIO1_GENCTL[5:4]) to determine the initial I/O mode is INPUT mode.
   2) Set 0 to SCSEL(PSIO1_GENCTL[25:24])
7. Set PIN2 general control register(PSIO2_GENCTL).
   1) Set 0b10 to IOMODE(PSIO2_GENCTL[5:4]) to determine the initial I/O mode is INPUT mode.
   2) Set 0 to SCSEL(PSIO1_GENCTL[25:24])
8. Set PIN0 data controller (PSIO0_DATCTL) to determine the data property.
   1) Set 0 to ORDER(PSIO0_DATCTL[16]) to determine the data order is LSB.
   2) Set 0x0 to INDATWD(PSIO0_DATCTL[12:8]) to determine the input data width is 1 bit.
   3) Set 0x3 to INDEPTH(PSIO0_DATCTL[29:28]) to determine the input data depth is 4.
9. Set PIN1 data controller (PSIO1_DATCTL) to determine the data property.
   1) Set 0 to ORDER(PSIO1_DATCTL[16]) to determine the data order is LSB.
   2) Set 0x0 to INDATWD(PSIO1_DATCTL[12:8]) to determine the input data width is 1 bit.
   3) Set 0x3 to INDEPTH(PSIO1_DATCTL[29:28]) to determine the input data depth is 4.
10. Set PIN2 data controller (PSIO2_DATCTL) to determine the data property.
    1) Set 0 to ORDER(PSIO2_DATCTL[16]) to determine the data order is LSB.
    2) Set 0x0 to INDATWD(PSIO2_DATCTL[12:8]) to determine the input data width is 1 bit.
    3) Set 0x3 to INDEPTH(PSIO2_DATCTL[29:28]) to determine the input data depth is 4.
11. Set PIN0 check point controller 0 (PSIO0_CPCTL0) to determine which slot will be link to the check point.
    1) Set 0x1(SLOT0) to CKPT0(PSIO0_CPCTL0[3:0]) to link the slot0 to check point0.
    2) Set 0x2(SLOT1) to CKPT1(PSIO0_CPCTL0[7:4]) to link the slot1 to check point1.
    3) Set 0x3(SLOT2) to CKPT2(PSIO0_CPCTL0[11:8]) to link the slot2 to check point2.
    4) Set 0x4(SLOT3) to CKPT3(PSIO0_CPCTL0[15:12]) to link the slot3 to check point3.
12. Set PIN1 check point controller 0 (PSIO1_CPCTL0) to determine which slot will be link to the check point.
    1) Set 0x1(SLOT0) to CKPT0(PSIO1_CPCTL0[3:0]) to link the slot0 to check point0.
    2) Set 0x2(SLOT1) to CKPT1(PSIO1_CPCTL0[7:4]) to link the slot1 to check point1.

3) Set 0x3(SLOT2) to CKPT2(PSIO1_CPCTL0[11:8]) to link the slot2 to check point2.

4) Set 0x4(SLOT3) to CKPT3(PSIO1_CPCTL0[15:12]) to link the slot3 to check point3.

13. Set PIN2 check point controller 0 (PSIO2_CPCTL0) to determine which slot will be link to the check point.

1) Set 0x1(SLOT0) to CKPT0(PSIO2_CPCTL0[3:0]) to link the slot0 to check point0.

2) Set 0x2(SLOT1) to CKPT1(PSIO2_CPCTL0[7:4]) to link the slot1 to check point1.

3) Set 0x3(SLOT2) to CKPT2(PSIO2_CPCTL0[11:8]) to link the slot2 to check point2.

4) Set 0x4(SLOT3) to CKPT3(PSIO2_CPCTL0[15:12]) to link the slot3 to check point3.

14. Set PIN0 check point controller 1 (PSIO0_CP1CTL) to determine the check point action at the check points.

1) Set 0b100 (input data) to CKPT0ACT(PSIO0_CPCTL1[2:0]) to input data at the check point 0.

2) Set 0b100 (input data) to CKPT1ACT(PSIO0_CPCTL1[6:4]) to input data at the check point1.

3) Set 0b100 (input data) to CKPT2ACT(PSIO0_CPCTL1[10:8]) to input data at the check point2.

4) Set 0b100 (input data) to CKPT3ACT(PSIO0_CPCTL1[14:12]) to input data at the check point3.

15. Set PIN1 check point controller 1 (PSIO1_CP1CTL) to determine the check point action at the check points.

1) Set 0b100 (input data) to CKPT0ACT(PSIO1_CPCTL1[2:0]) to input data at the check point 0.

2) Set 0b100 (input data) to CKPT1ACT(PSIO1_CPCTL1[6:4]) to input data at the check point1.

3) Set 0b100 (input data) to CKPT2ACT(PSIO1_CPCTL1[10:8]) to input data at the check point2.

4) Set 0b100 (input data) to CKPT3ACT(PSIO1_CPCTL1[14:12]) to input data at the check point3.

16. Set PIN2 check point controller 1 (PSIO2_CP1CTL) to determine the check point action at the check points.

1) Set 0b100 (input data) to CKPT0ACT(PSIO2_CPCTL1[2:0]) to input data at the check point 0.

2) Set 0b100 (input data) to CKPT1ACT(PSIO2_CPCTL1[6:4]) to input data at the check point1.

3) Set 0b100 (input data) to CKPT2ACT(PSIO2_CPCTL1[10:8]) to input data at the check point2.

4) Set 0b100 (input data) to CKPT3ACT(PSIO2_CPCTL1[14:12]) to input data at the check point3.

17. Set PDMA related setting to receive data from PSIO to SRAM.

1) Set PSIO_PIDAT address (PSIO_BA + 0x1C) to PDMA_DSCTn_SA as source address.

2) Set 0x20000000 to PDMA_DSCTn_DA as destination address.

3) Set TXCNT(PDMA_DSCTn_CTL[31:16]) = 3 to set PDMA transfer data from PSIO 4 times.

4) Set 0x10 to TXWIDTH(PDMA_DSCTn_CTL[13:12]) to select the data width is 32 bit.

5) Set 0x11 to SAINC(PDMA_DSCTn_CTL[9:8]) to select the source address imcrement is 0.

18. Set PSIO_SC0CTL to start the slot controller.

Set 0x1 to TRIGSRC(PSIO_SC0CTL[15:14], SC0 Trigger Source) to determine the trigger source is by falling edge.

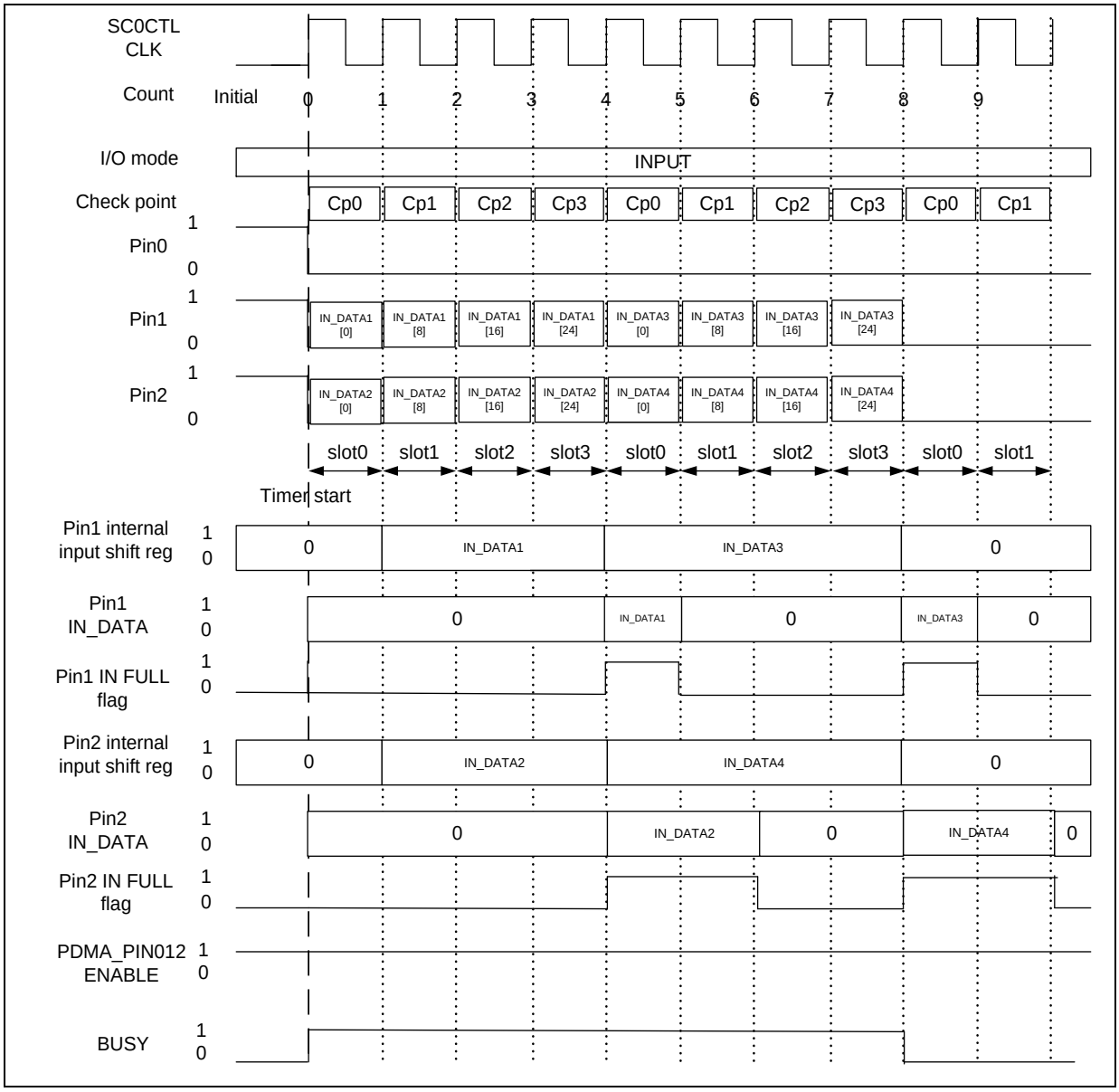


Figure 6.32-24 PDMA with INPUT DATA and SLOT CONTROLLER WHOLE REPEAT Mode with 2 Pins

6.32.8 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
PSIO Base Address: **PSIO_BA = 0x400C_3000**				
PSIO_INTCTL	PSIO_BA+0x00	R/W	PSIO Interrupt Control Register	0x0000_0000
PSIO_INTEN	PSIO_BA+0x04	R/W	PSIO Interrupt Enable Register	0x0000_0000
PSIO_INTSTS	PSIO_BA+0x08	R/W	PSIO Interrupt Status Register	0x0000_0000
PSIO_TRANSTS	PSIO_BA+0x0C	R/W	PSIO Transfer Status Register	0x0000_0000
PSIO_ISSTS	PSIO_BA+0x10	R/W	PSIO Input Status State Register	0x0000_0000
PSIO_PDMACTL	PSIO_BA+0x14	R/W	PSIO PDMA Control Register	0x0000_0000
PSIO_PODAT	PSIO_BA+0x18	W	PSIO PDMA Output Data Register	0x0000_0000
PSIO_PIDAT	PSIO_BA+0x1C	R/W	PSIO PDMA Input Data Register	0x0000_0000
PSIO_SCnCTL **n=0,1..3**	PSIO_BA+0x20+(0x08 * n)	R/W	PSIO Slot Controller n Control Register	0x0200_0000
PSIO_SCnSLOT **n=0,1..3**	PSIO_BA+0x24+(0x08 * n)	R/W	PSIO Slot Controller n Slot Register	0x0000_0000
PSIOn_GENCTL **n=0,1..7**	PSIO_BA+0x40+(0x20 * n)	R/W	PSIOn General Control Register	0x0000_0000
PSIOn_DATCTL **n=0,1..7**	PSIO_BA+0x44+(0x20 * n)	R/W	PSIOn Data Control Register	0x3300_0000
PSIOn_INSTS **n=0,1..7**	PSIO_BA+0x48+(0x20 * n)	R	PSIOn Input Status Register	0x0000_0000
PSIOn_INDAT **n=0,1..7**	PSIO_BA+0x4C+(0x20 * n)	R	PSIOn Input Data Register	0x0000_0000
PSIOn_OUTDAT **n=0,1..7**	PSIO_BA+0x50+(0x20 * n)	W	PSIOn Output Data Register	0x0000_0000
PSIOn_CPCTL0 **n=0,1..7**	PSIO_BA+0x54+(0x20 * n)	R/W	PSIOn Check Point Control Register 0	0x0000_0000
PSIOn_CPCTL1 **n=0,1..7**	PSIO_BA+0x58+(0x20 * n)	R/W	PSIOn Check Point Control Register 1	0x0000_0000

6.32.9 Register Description

PSIO Interrupt Control Register (PSIO_INTCTL)

Register	Offset	R/W	Description	Reset Value
PSIO_INTCTL	PSIO_BA+0x00	R/W	PSIO Interrupt Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CONI1SCS**		**Reserved**		**CONI0SCS**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CONI1SS				**CONI0SS**			

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:12]	**CONI1SCS**	**Configurable Interrupt 1 Slot Controller Selection** Select Slot controller for interrupt 1. 00 = Slot controller 0. 01 = Slot controller 1. 10 = Slot controller 2. 11 = Slot controller 3.
[11]	**Reserved**	Reserved.
[9:8]	**CONI0SCS**	**Configurable Interrupt 0 Slot Controller Selection** Select Slot controller for interrupt 0. 00 = Slot controller 0. 01 = Slot controller 1. 10 = Slot controller 2. 11 = Slot controller 3.
[7]	**Reserved**	Reserved.
[6:4]	**CONI1SS**	**Configurable Interrupt 1 Slot Selection** 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7.

		Others = Reserved.
[3]	**Reserved**	Reserved.
[2:0]	**CONI0SS**	**Configurable Interrupt 0 Slot Selection** 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved.

PSIO Interrupt Enable Register (PSIO_INTEN)

Register	Offset	R/W	Description	Reset Value
PSIO_INTEN	PSIO_BA+0x04	R/W	PSIO Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
SC3IE	SC2IE	SC1IE	SC0IE	TERRIE	MISMATIE	CON1IE	CON0IE

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**SC3IE**	**Slot Controller 3 Done Interrupt Enable Bit** This field is used to enable Slot controller 3 finish interrupt. 0 = Slot controller 3 finish interrupt Disabled. 1 = Slot controller 3 finish interrupt Enabled.
[6]	**SC2IE**	**Slot Controller 2 Done Interrupt Enable Bit** This field is used to enable Slot controller 2 finish interrupt. 0 = Slot controller 2 finish interrupt Disabled. 1 = Slot controller 2 finish interrupt Enabled.
[5]	**SC1IE**	**Slot Controller 1 Done Interrupt Enable Bit** This field is used to enable Slot controller 1 finish interrupt. 0 = Slot controller 1 finish interrupt Disabled. 1 = Slot controller 1 finish interrupt Enabled.
[4]	**SC0IE**	**Slot Controller 0 Done Interrupt Enable Bit** This field is used to enable Slot controller 0 finish interrupt. 0 = Slot controller 0 finish interrupt Disabled. 1 = Slot controller 0 finish interrupt Enabled.
[3]	**TERRIE**	**Transfer Error Interrupt Enable Bit** This field is used to enable transfer error interrupt. 0 = Transfer error interrupt Disabled. 1 = Transfer error interrupt Enabled.
[2]	**MISMATIE**	**Mismatch Interrupt Enable Bit** This field is used to enable mismatch interrupt. 0 = Mismatch interrupt Disabled. 1 = Mismatch interrupt Enabled.
[1]	**CON1IE**	**Configurable Interrupt 1 Enable Bit**

		This field is used to enable selective interrupt 1. 0 = Selective interrupt 1 Disabled. 1 = Selective interrupt 1 Enabled.
[0]	**CON0IE**	**Configurable Interrupt 0 Enable Bit** This field is used to enable selective interrupt 0. 0 = Selective interrupt 0 Disabled. 1 = Selective interrupt 0 Enabled.

PSIO Interrupt Status Register (PSIO_INTSTS)

Register	Offset	R/W	Description	Reset Value
PSIO_INTSTS	PSIO_BA+0x08	R/W	PSIO Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
SC3IF	SC2IF	SC1IF	SC0IF	TERRIF	MISMATIF	CON1IF	CON0IF

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**SC3IF**	**Slot Controller 3 Counting Done Interrupt Status Flag** This field is used for slot controller 3 finish interrupt status flag. 0 = Slot controller 3 done interrupt did not occur. 1 = Slot controller 3 done interrupt occurred. **Note:** This bit can be cleared by writing 1.
[6]	**SC2IF**	**Slot Controller 2 Counting Done Interrupt Status Flag** This field is used for slot controller 2 finish interrupt status flag. 0 = Slot controller 2 done interrupt did not occur. 1 = Slot controller 2 done interrupt occurred. **Note:** This bit can be cleared by writing 1.
[5]	**SC1IF**	**Slot Controller 1 Counting Done Interrupt Status Flag** This field is used for slot controller 1 finish interrupt status flag. 0 = Slot controller 1 done interrupt did not occur. 1 = Slot controller 1 done interrupt occurred. **Note:** This bit can be cleared by writing 1.
[4]	**SC0IF**	**Slot Controller 0 Counting Done Interrupt Status Flag** This field is used for slot controller 0 finish interrupt status flag. 0 = Slot controller 0 done interrupt did not occur. 1 = Slot controller 0 done interrupt occurred. **Note:** This bit can be cleared by writing 1.
[3]	**TERRIF**	**Transfer Error Interrupt Status Flag** This field is used for transfer error interrupt status flag. The transfer error states is at PSIO_TRANSTS register which includes input data overflow flag INOVERn (PSIO_TRANSTS[29,25,21,17,13,9,5,1] and output data underflow flag OUTUFn (PSIO_TRANSTS[31,27,23,19,15,11,7,3]). 0 = Transfer error interrupt did not occur. 1 = Transfer error interrupt occurred.

		Note 1: This field is the status flag of INOVER or OUTUFER. **Note 2:** This bit can only be cleared by writing 1 to coordinate transfer error.
[2]	**MISMATIF**	**Mismatch Interrupt Flag** This flag shows the amounts of data are not the same in each pins with PDMA enabled. If this situation happens, all slot controllers stop counting. 0 = Each pin with PDMA enabled receive or transfer data in the same rate. 1 = Each pin with PDMA enabled receive or transfer data in different rate. **Note 1:** This flag is only effective on the pin with PDMA enabled. **Note 2:** This bit can be cleared by writing 1.
[1]	**CON1IF**	**Configurable Interrupt 1 Flag** The setting interrupt is trigger at the end of the check point of the pin. 0 = Condition in PSIO_INTCTL is not triggered. 1 = Condition in PSIO_INTCTL is triggered. **Note:** This bit can be cleared by writing 1.
[0]	**CON0IF**	**Configurable Interrupt 0 Flag** The setting interrupt is trigger at the end of the check point of the pin. 0 = Condition in PSIO_INTCTL is not triggered. 1 = Condition in PSIO_INTCTL is triggered. **Note:** This bit can be cleared by writing 1.

PSIO Transfer Status Register (PSIO_TRANSTS)

Register	Offset	R/W	Description	Reset Value
PSIO_TRANSTS	PSIO_BA+0x0C	R/W	PSIO Transfer Status Register	0x0000_0000

31	30	29	28	27	26	25	24
OUTUF7	OUTEPY7	INOVER7	INFULL7	OUTUF6	OUTEPY6	INOVER6	INFULL6
23	22	21	20	19	18	17	16
OUTUF5	OUTEPY5	INOVER5	INFULL5	OUTUF4	OUTEPY4	INOVER4	INFULL4
15	14	13	12	11	10	9	8
OUTUF3	OUTEPY3	INOVER3	INFULL3	OUTUF2	OUTEPY2	INOVER2	INFULL2
7	6	5	4	3	2	1	0
OUTUF1	OUTEPY1	INOVER1	INFULL1	OUTUF0	OUTEPY0	INOVER0	INFULL0

Bits	Description	
[31]	**OUTUF7**	**Output Data Underflow Flag7** When PSIO is still output data but PSIOn_OUTDAT have not been ready, this flag will be set to 1. 0 = The pin7 output data is not underflow. 1 = The pin7 output data is underflow. **Note 1:** When output data shortage happened, it will output 0. **Note 2:** When underflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[30]	**OUTEPY7**	**Output Data Empty Flag7 (Read Only)** 0 = The pin7 output data is full. 1 = The pin7 output data is empty.
[29]	**INOVER7**	**Input Data Overflow Flag7** 0 = The pin7 input data does not occur overflow. 1 = The pin7 input data occurs overflow. **Note 1:** When input Overflow happened, it will keep the current data, and discard the upcoming data. **Note 2:** When overflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[28]	**INFULL7**	**Input Data Full Flag7 (Read Only)** 0 = The pin7 input data is empty. 1 = The pin7 input data is full. **Note:** This bit will be cleared automatically when related slot controller started and pin enabled.
[27]	**OUTUF6**	**Output Data Underflow Flag6** When PSIO is still output data but PSIOn_OUTDAT have not been ready, this

		flag will be set to 1. 0 = The pin6 output data is not underflow. 1 = The pin6 output data is underflow. **Note 1:** When output data shortage happened, it will output 0. **Note 2:** When underflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[26]	**OUTEPY6**	**Output Data Empty Flag6 (Read Only)** 0 = The pin6 output data is full. 1 = The pin6 output data is empty.
[25]	**INOVER6**	**Input Data Overflow Flag6** 0 = The pin6 input data does not occur overflow. 1 = The pin6 input data occurs overflow. **Note 1:** When input Overflow happened, it will keep the current data, and discard the upcoming data. **Note 2:** When overflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[24]	**INFULL6**	**Input Data Full Flag6 (Read Only)** 0 = The pin6 input data is empty. 1 = The pin6 input data is full. **Note:** This bit will be cleared automatically when related slot controller started and pin enabled.
[23]	**OUTUF5**	**Output Data Underflow Flag5** When PSIO is still output data but PSIOn_OUTDAT have not been ready. This flag will be set to 1. 0 = The pin5 output data is not underflow. 1 = The pin5 output data is underflow. **Note 1:** When output data shortage happened, it will output 0. **Note 2:** When underflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin is enabled.
[22]	**OUTEPY5**	**Output Data Empty Flag5 (Read Only)** 0 = The pin5 output data is full. 1 = The pin5 output data is empty.
[21]	**INOVER5**	**Input Data Overflow Flag5** 0 = The pin5 input data does not occur overflow. 1 = The pin5 input data occurs overflow. **Note 1:** When input Overflow happened, it will keep the current data, and discard the upcoming data. **Note 2:** When overflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[20]	**INFULL5**	**Input Data Full Flag5 (Read Only)** 0 = The pin5 input data is empty.

		1 = The pin5 input data is full. **Note:** This bit will be cleared automatically when related slot controller started and pin enabled.
[19]	**OUTUF4**	**Output Data Underflow Flag4** When PSIO is still output data but PSIOn_OUTDAT have not been ready. This flag will be set to 1. 0 = The pin4 output data is not underflow. 1 = The pin4 output data is underflow. **Note 1:** When output data shortage happened, it will output 0. **Note 2:** When underflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[18]	**OUTEPY4**	**Output Data Empty Flag4 (Read Only)** 0 = The pin4 output data is full. 1 = The pin4 output data is empty.
[17]	**INOVER4**	**Input Data Overflow Flag4** 0 = The pin4 input data does not occur overflow. 1 = The pin4 input data occurs overflow. **Note 1:** When input Overflow happened, it will keep the current data, and discard the upcoming data. **Note 2:** When overflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[16]	**INFULL4**	**Input Data Full Flag4 (Read Only)** 0 = The pin4 input data is empty. 1 = The pin4 input data is full. **Note:** This bit will be cleared automatically when related slot controller started and pin enabled.
[15]	**OUTUF3**	**Output Data Underflow Flag3** When PSIO is still output data but PSIOn_OUTDAT have not been ready. This flag will be set to 1. 0 = The pin3 output data is not underflow. 1 = The pin3 output data is underflow. **Note 1:** When output data shortage happened, it will output 0. **Note 2:** When underflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[14]	**OUTEPY3**	**Output Data Empty Flag3 (Read Only)** 0 = The pin3 output data is full. 1 = The pin3 output data is empty.
[13]	**INOVER3**	**Input Data Overflow Flag3** 0 = The pin3 input data does not occur overflow. 1 = The pin3 input data occurs overflow. **Note 1:** When input Overflow happened, it will keep the current data, and discard the upcoming data. **Note 2:** When overflow happens, related slot controller will be stopped.

		Note 3: This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[12]	**INFULL3**	**Input Data Full Flag3 (Read Only)** 0 = The pin3 input data is empty. 1 = The pin3 input data is full. **Note:** This bit will be cleared automatically when related slot controller started and pin enabled.
[11]	**OUTUF2**	**Output Data Underflow Flag2** When PSIO is still output data but PSIOn_OUTDAT have not been ready. This flag will be set to 1. 0 = The pin3 output data is not underflow. 1 = The pin3 output data is underflow. **Note 1:** When output data shortage happened, it will output 0. **Note 2:** When underflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[10]	**OUTEPY2**	**Output Data Empty Flag2 (Read Only)** 0 = The pin2 output data is full. 1 = The pin2 output data is empty.
[9]	**INOVER2**	**Input Data Overflow Flag2** 0 = The pin2 input data does not occur overflow. 1 = The pin2 input data occurs overflow. **Note 1:** When input Overflow happened, it will keep the current data, and discard the upcoming data. **Note 2:** When overflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[8]	**INFULL2**	**Input Data Full Flag2 (Read Only)** 0 = The pin2 input data is empty. 1 = The pin2 input data is full. **Note:** This bit will be cleared automatically when related slot controller started and pin is enabled.
[7]	**OUTUF1**	**Output Data Underflow Flag1** When PSIO is still output data but PSIOn_OUTDAT have not been ready. This flag will be set to 1. 0 = The pin1 output data is not underflow. 1 = The pin1 output data is underflow. **Note 1:** When output data shortage happened, it will output 0. **Note 2:** When underflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled
[6]	**OUTEPY1**	**Output Data Empty Flag1 (Read Only)** 0 = The pin1 output data is full. 1 = The pin1 output data is empty.
[5]	**INOVER1**	**Input Data Overflow Flag1**

		0 = The pin1 input data does not occur overflow. 1 = The pin1 input data occurs overflow. **Note 1:** When input Overflow happened, it will keep the current data, and discard the upcoming data. **Note 2:** When overflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[4]	**INFULL1**	**Input Data Full Flag1 (Read Only)** 0 = The pin1 input data is empty. 1 = The pin1 input data is full. **Note:** This bit will be cleared automatically when related slot controller started and pin enabled.
[3]	**OUTUF0**	**Output Data Underflow Flag0** When PSIO is still output data but PSIOn_OUTDAT have not been ready. This flag will be set to 1. 0 = The pin0 output data is not underflow. 1 = The pin0 output data is underflow. **Note 1:** When output data shortage happened, it will output 0. **Note 2:** When underflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[2]	**OUTEPY0**	**Output Data Empty Flag0 (Read Only)** 0 = The pin0 output data is full. 1 = The pin0 output data is empty.
[1]	**INOVER0**	**Input Data Overflow Flag0** 0 = The pin0 input data does not occur overflow. 1 = The pin0 input data occurs overflow. **Note 1:** When input Overflow happened, it will keep the current data, and discard the upcoming data. **Note 2:** When overflow happens, related slot controller will be stopped. **Note 3:** This bit can be cleared by configure 1 to it. **Note 4:** This bit will be cleared automatically when related slot controller started and pin enabled.
[0]	**INFULL0**	**Input Data Full Flag0 (Read Only)** 0 = The pin0 input data is empty. 1 = The pin0 input data is full. **Note:** This bit will be cleared automatically when related slot controller started and pin enabled.

PSIO Input Status State Register (PSIO_ISSTS)

Register	Offset	R/W	Description	Reset Value
PSIO_ISSTS	PSIO_BA+0x10	R/W	PSIO Input Status State Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
INSTSOV7	VALID7	INSTSOV6	VALID6	INSTSOV5	VALID5	INSTSOV4	VALID4
7	6	5	4	3	2	1	0
INSTSOV3	VALID3	INSTSOV2	VALID2	INSTSOV1	VALID1	INSTSOV0	VALID0

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**INSTSOV7**	**Input Status Overflow 7** 0 = The pin7 input status does not overflow. 1 = The pin7 input status occur overflow. **Note:** This overflow bit can be cleared by writing 1.
[14]	**VALID7**	**Input Status Valid 7** 0 = The pin7 input status is not ready. 1 = The pin7 input status is ready. **Note:** This valid bit will be cleared automatically if PSIOn_INSTS is read.
[13]	**INSTSOV6**	**Input Status Overflow 6** 0 = The pin 6 input status does not overflow. 1 = The pin 6 input status occur overflow. **Note:** This overflow bit can be cleared by writing 1.
[12]	**VALID6**	**Input Status Valid 6** 0 = The pin 6 input status is not ready. 1 = The pin 6 input status is ready. **Note:** This valid bit will be cleared automatically if PSIOn_INSTS is read.
[11]	**INSTSOV5**	**Input Status Overflow 5** 0 = The pin 5 input status does not overflow. 1 = The pin 5 input status occur overflow. **Note:** This overflow bit can be cleared by writing 1.
[10]	**VALID5**	**Input Status Valid 5** 0 = The pin 5 input status is not ready. 1 = The pin 5 input status is ready. **Note:** This valid bit will be cleared automatically if PSIOn_INSTS is read.
[9]	**INSTSOV4**	**Input Status Overflow 4**

		0 = The pin 4 input status does not overflow. 1 = The pin 4 input status occur overflow. **Note:** This overflow bit can be cleared by writing 1.
[8]	**VALID4**	**Input Status Valid 4** 0 = The pin 4 input status is not ready. 1 = The pin 4 input status is ready. **Note:** This valid bit will be cleared automatically if PSIOn_INSTS is read.
[7]	**INSTSOV3**	**Input Status Overflow 3** 0 = The pin 3 input status does not overflow. 1 = The pin 3 input status occur overflow. **Note:** This overflow bit can be cleared by writing 1.
[6]	**VALID3**	**Input Status Valid 3** 0 = The pin 3 input Status is not ready. 1 = The pin 3 input Status is ready. **Note:** This valid bit will be cleared automatically if PSIOn_INSTS is read.
[5]	**INSTSOV2**	**Input Status Overflow 2** 0 = The pin 2 input status does not overflow. 1 = The pin 2 input status occur overflow. **Note:** This overflow bit can be cleared by writing 1.
[4]	**VALID2**	**Input Status Valid 2** 0 = The pin 2 input status is not ready. 1 = The pin 2 input status is ready. **Note:** This valid bit will be cleared automatically if PSIOn_INSTS is read.
[3]	**INSTSOV1**	**Input Status Overflow 1** 0 = The pin 1 input status does not overflow. 1 = The pin 1 input status occur overflow. **Note:** This overflow bit can be cleared by writing 1.
[2]	**VALID1**	**Input Status Valid 1** 0 = The pin 1 input status is not ready. 1 = The pin 1 input status is ready. **Note:** This valid bit will be cleared automatically if PSIOn_INSTS is read.
[1]	**INSTSOV0**	**Input Status Overflow 0** 0 = The pin 0 input status does not overflow. 1 = The pin 0 input status occur overflow. **Note:** This overflow bit can be cleared by writing 1.
[0]	**VALID0**	**Input Status Valid 0** 0 = The pin 0 input status is not ready. 1 = The pin 0 input status is ready. **Note:** This valid bit will be cleared automatically if PSIOn_INSTS is read.

PSIO PDMA Control Register (PSIO_PDMACTL)

Register	Offset	R/W	Description	Reset Value
PSIO_PDMACTL	PSIO_BA+0x14	R/W	PSIO PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		INSCSEL		INNUM			
23	22	21	20	19	18	17	16
Reserved		OUTSCSEL		OUTNUM			
15	14	13	12	11	10	9	8
IPIN7EN	IPIN6EN	IPIN5EN	IPIN4EN	IPIN3EN	IPIN2EN	IPIN1EN	IPIN0EN
7	6	5	4	3	2	1	0
OPIN7EN	OPIN6EN	OPIN5EN	OPIN4EN	OPIN3EN	OPIN2EN	OPIN1EN	OPIN0EN

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**INSCSEL**	**PDMA Input Data Slot Controller Selection** 00 = Slot controller 0. 01 = Slot controller 1. 10 = Slot controller 2. 11 = Slot controller 3.
[27:24]	**INNUM**	**PDMA Input Current Number (Read Only)** This register shows the current pin number of input register read by PDMA. 0000 = PDMA IDLE. 0001 = Pin 0. 0010 = Pin 1. 0011 = Pin 2. 0100 = Pin 3. 0101 = Pin 4. 0110 = Pin 5. 0111 = Pin 6. 1000 = Pin 7. 1001 = PDMA WAIT. Others = Reserved.
[21:20]	**OUTSCSEL**	**PDMA Output Data Slot Controller Selection** 00 = Slot controller 0. 01 = Slot controller 1. 10 = Slot controller 2. 11 = Slot controller 3.
[19:16]	**OUTNUM**	**PDMA Output Current Number (Read Only)** This register shows the current pin number of output register write by PDMA. 0000 = PDMA IDLE.

		0001 = Pin 0. 0010 = Pin 1. 0011 = Pin 2. 0100 = P3. 0101 = Pin 4. 0110 = Pin 5. 0111 = Pin 6. 1000 = Pin 7. 1001 = PDMA WAIT. Others = Reserved.
[15]	**IPIN7EN**	**Input PDMA Pin7 Enable Bit** 0 = Pin7 input PDMA function Disabled. 1 = Pin7 input PDMA function Enabled.
[14]	**IPIN6EN**	**Input PDMA Pin6 Enable Bit** 0 = Pin6 input PDMA function Disabled. 1 = Pin6 input PDMA function Enabled.
[13]	**IPIN5EN**	**Input PDMA Pin5 Enable Bit** 0 = Pin5 input PDMA function Disabled. 1 = Pin5 input PDMA function Enabled.
[12]	**IPIN4EN**	**Input PDMA Pin4 Enable Bit** 0 = Pin4 input PDMA function Disabled. 1 = Pin4 input PDMA function Enabled.
[11]	**IPIN3EN**	**Input PDMA Pin3 Enable Bit** 0 = Pin3 input PDMA function Disabled. 1 = Pin3 input PDMA function Enabled.
[10]	**IPIN2EN**	**Input PDMA Pin2 Enable Bit** 0 = Pin2 input PDMA function Disabled. 1 = Pin2 input PDMA function Enabled.
[9]	**IPIN1EN**	**Input PDMA Pin1 Enable Bit** 0 = Pin1 input PDMA function Disabled. 1 = Pin1 input PDMA function Enabled.
[8]	**IPIN0EN**	**Input PDMA Pin0 Enable Bit** 0 = Pin0 input PDMA function Disabled. 1 = Pin0 input PDMA function Enabled.
[7]	**OPIN7EN**	**Output PDMA Pin7 Enable Bit** 0 = Pin7 output PDMA function Disabled. 1 = Pin7 output PDMA function Enabled.
[6]	**OPIN6EN**	**Output PDMA Pin6 Enable Bit** 0 = Pin6 output PDMA function Disabled. 1 = Pin6 output PDMA function Enabled.
[5]	**OPIN5EN**	**Output PDMA Pin5 Enable Bit** 0 = Pin5 output PDMA function Disabled. 1 = Pin5 output PDMA function Enabled.
[4]	**OPIN4EN**	**Output PDMA Pin4 Enable Bit**

		0 = Pin4 output PDMA function Disabled. 1 = Pin4 output PDMA function Enabled.
[3]	**OPIN3EN**	**Output PDMA Pin3 Enable Bit** 0 = Pin3 output PDMA function Disabled. 1 = Pin3 output PDMA function Enabled.
[2]	**OPIN2EN**	**Output PDMA Pin2 Enable Bit** 0 = Pin2 output PDMA function Disabled. 1 = Pin2 output PDMA function Enabled.
[1]	**OPIN1EN**	**Output PDMA Pin1 Enable Bit** 0 = Pin1 output PDMA function Disabled. 1 = Pin1 output PDMA function Enabled.
[0]	**OPIN0EN**	**Output PDMA Pin0 Enable Bit** 0 = Pin0 output PDMA function Disabled. 1 = Pin0 output PDMA function Enabled.

PSIO PDMA Output Data Register (PSIO_PODAT)

Register	Offset	R/W	Description	Reset Value
PSIO_PODAT	PSIO_BA+0x18	W	PSIO PDMA Output Data Register	0x0000_0000

31	30	29	28	27	26	25	24
PDMAOUT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PDMAOUT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PDMAOUT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDMAOUT							

Bits	Description	
[31:0]	**PDMAOUT**	**PDMA Output Data** This register is used for PSIO with PDMA single mode, and set PDMA with fixed address. When PSIO in PDMA mode, setting PDMA to write data to this register. The data in this register will be placed to corresponding PSIOn_OUTDAT register in order, when Output Data Empty Flag is 1 and PDMA mode enabled.

PSIO PDMA Input Data Register (PSIO_PIDAT)

Register	Offset	R/W	Description	Reset Value
PSIO_PIDAT	PSIO_BA+0x1C	R/W	PSIO PDMA Input Data Register	0x0000_0000

31	30	29	28	27	26	25	24
PDMAIN							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PDMAIN							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PDMAIN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDMAIN							

Bits	Description	
[31:0]	**PDMAIN**	**PDMA Input Data** This register is used for PSIO with PDMA single mode, and set PDMA with fixed address. When PSIO in PDMA mode, setting PDMA to read data from this register. The data in this register will be updated from corresponding PSIOn_INDAT register in order, when Input Data Full Flag is 1 and PDMA mode enable.

PSIO Slot Controller n Control Register (PSIO_SCnCTL)

Register	Offset	R/W	Description	Reset Value
PSIO_SCnCTL n=0,1..3	PSIO_BA+0x20+(0x08 * n)	R/W	PSIO Slot Controller n Control Register	0x0200_0000

31	30	29	28	27	26	25	24
Reserved						IDLE	BUSY
23	22	21	20	19	18	17	16
Reserved					STOP	REPEAT	START
15	14	13	12	11	10	9	8
TRIGSRC		SPLCNT					
7	6	5	4	3	2	1	0
ENDSLOT				INISLOT			

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**IDLE**	**PSIO_SCn Idle Flag** 0 = PSIO_SCn is not IDLE. 1 = PSIO_SCn is IDLE. **Note 1:** This bit will be cleared to 0 when slot controller starts to count automatically. **Note 2:** This bit will be set to 1 when configuring it 1 by software. **Note 3:** This bit is set to distinguish INTERVAL (PSIOn_GENCTL[5:4]) and INITIAL (PSIOn_GENCTL[3:2]).
[24]	**BUSY**	**PSIO_SCn Busy Flag** 0 = PSIO_SCn is not busy. 1 = PSIO_SCn is busy. **Note:** This bit will be set to 1 when slot controller starts to count automatically and it will be cleared to 0 automatically when slot controller stops counting, too.
[23:19	**Reserved**	Reserved.
[18]	**STOP**	**PSIO_SCn Stop** 0 = No use. 1 = Stop PSIO_SCn. **Note:** This bit is always read as 0.
[17]	**REPEAT**	**Whole Repeat Mode** Slot controller repeats counting forever. 0 = Repeat mode Disabled. 1 = Repeat mode Enabled. **Note 1:** If this bit is enabled with PDMA mode, slot controller will stop automatically when the PDMA finishes transferring number of data. **Note 2:** If PSIO receives stop instruction during repeat mode, it will stop only when the current loop is finished.
[16]	**START**	**PSIO_SCn Start**

		0 = No use. 1 = Start PSIO_SCn to count and active related PSIO_PIN. **Note:** This bit is always read as 0.
[15:14]	**TRIGSRC**	**PSIO_SCn Trigger Source** 00 = Trigger by software. 01 = Trigger PSIO_SCn when related PSIO_PIN occurred falling edge. 10 = Trigger PSIO_SCn when related PSIO_PIN occurred rising edge. 11 = Trigger PSIO_SCn when related PSIO_PIN occurred rising edge or falling edge. **Note 1:** PSIO slot controller pin can only be triggered by related pins set from SCSEL(PSIOn_GENCTL[25:24]). **Note 2:** Configuring rising or falling signal trigger PSIO, the signal needs to hold for at least two PSIO_CLK for de-bounce or PSIO will not be triggered.
[13:8]	**SPLCNT**	**Slot Period Loop Count** 000000 ~ 111110 is loop count. 000000 = Slot period loop count function is disable. 000001 = Repeat selection loop once, which means total go through selected repeat slots 2 times. 111111 = Loop until stop PSIO slot controller. **Note 1:** If setting this register 111111 with PDMA mode and OUTPUT mode, it will stop automatically when PDMA is finished and output data in shift register is finished. **Note 2:** If setting this register 111111 with PDMA mode and INPUT mode, it will stop automatically when PDMA is finished. **Note 3:** If PSIO receives stop instruction during repeat mode, it will stop only when the current loop is finished.
[7:4]	**ENDSLOT**	**End Slot Period** The end slot of the repeat period. 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved.
[3:0]	**INISLOT**	**Initial Slot Period** The initial slot of the repeat period. 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved.

PSIO Slot Controller n Slot Register (PSIO_SCnSLOT)

Register	Offset	R/W	Description	Reset Value
PSIO_SCnSLOT n=0,1..3	PSIO_BA+0x24+(0x08 * n)	R/W	PSIO Slot Controller n Slot Register	0x0000_0000

31	30	29	28	27	26	25	24
SLOT7				SLOT6			
23	22	21	20	19	18	17	16
SLOT5				SLOT4			
15	14	13	12	11	10	9	8
SLOT3				SLOT2			
7	6	5	4	3	2	1	0
SLOT1				SLOT0			

Bits	Description	
[31:28]	**SLOT7**	**PSIO Slot Controller Slot7 Tick Count** 0 to 15. **Note 1:** Filling in all 0 to this field indicates to disable this slot. **Note 2:** The disabled slot should not be set between the enabled slots, or the order of enabled slot which is after the disabled slot will not be enabled. **Note 3:** The shortest slot length is 6 when I/O mode is switched from output mode to input mode.
[27:24]	**SLOT6**	**PSIO Slot Controller Slot6 Tick Count** 0 to 15. **Note 1:** Filling in all 0 to this field indicates to disable this slot. **Note 2:** The disabled slot should not be set between the enabled slots, or the order of enabled slot which is after the disabled slot will not be enabled. **Note 3:** The shortest slot length is 6 when I/O mode is switched from output mode to input mode.
[23:20]	**SLOT5**	**PSIO Slot Controller Slot5 Tick Count** 0 to 15. **Note 1:** Filling in all 0 to this field indicates to disable this slot. **Note 2:** The disabled slot should not be set between the enabled slots, or the order of enabled slot which is after the disabled slot will not be enabled. **Note 3:** The shortest slot length is 6 when I/O mode is switched from output mode to input mode.
[19:16]	**SLOT4**	**PSIO Slot Controller Slot4 Tick Count** 0 to 15. **Note 1:** Filling in all 0 to this field indicates to disable this slot. **Note 2:** The disabled slot should not be set between the enabled slots, or the order of enabled slot which is after the disabled slot will not be enabled. **Note 3:** The shortest slot length is 6 when I/O mode is switched from output mode to input mode.
[15:12]	**SLOT3**	**PSIO Slot Controller Slot3 Tick Count**

		0 to 15. **Note 1:** Filling in all 0 to this field indicates to disable this slot. **Note 2:** The disabled slot should not be set between the enabled slots, or the order of enabled slot which is after the disabled slot will not be enabled. **Note 3:** The shortest slot length is 6 when I/O mode is switched from output mode to input mode.
[11:8]	**SLOT2**	**PSIO Slot Controller Slot2 Tick Count** 0 to 15. **Note 1:** Filling in all 0 to this field indicates to disable this slot. **Note 2:** The disabled slot should not be set between the enabled slots, or the order of enabled slot which is after the disabled slot will not be enabled. **Note 3:** The shortest slot length is 6 when I/O mode is switched from output mode to input mode.
[7:4]	**SLOT1**	**PSIO Slot Controller Slot1 Tick Count** 0 to 15. **Note 1:** Filling in all 0 to this field indicates to disable this slot. **Note 2:** The disabled slot should not be set between the enabled slots, or the order of enabled slot which is after the disabled slot will not be enabled. **Note 3:** The shortest slot length is 6 when I/O mode is switched from output mode to input mode.
[3:0]	**SLOT0**	**PSIO Slot Controller Slot0 Tick Count** 0 to 15. **Note 1:** Filling in all 0 to this field indicates to disable this slot. **Note 2:** The disabled slot should not be set between the enabled slots, or the order of enabled slot which is after the disabled slot will not be enabled. **Note 3:** The shortest slot length is 6 when I/O mode is switched from output mode to input mode.

PSIOn General Control Register (PSIOn_GENCTL)

Register	Offset	R/W	Description	Reset Value
PSIOn_GENCTL **n=0,1..7**	PSIO_BA+0x40+(0x20 * n)	R/W	PSIOn General Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					PINEN	SCSEL	
23	22	21	20	19	18	17	16
Reserved				MODESW1		MODESW0	
15	14	13	12	11	10	9	8
SW1CP				SW0CP			
7	6	5	4	3	2	1	0
Reserved		INTERVAL		INITIAL		IOMODE	

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26]	**PINEN**	**Pin Enable Bit** 0 = Pin Disabled. 1 = Pin Enabled.
[25:24]	**SCSEL**	**Slot Controller Selection** Select slot controller for check point. 00 = SLOT CONTROLLER0. 01 = SLOT CONTROLLER1. 10 = SLOT CONTROLLER2. 11 = SLOT CONTROLLER3.
[23:22]	**Reserved**	Reserved.
[19:18]	**MODESW1**	**Mode Switch1 Point** Mode at the switch1 point. 00 = Input mode. 01 = Output mode. 10 = Open-drain mode. 11 = Quasi-bidirectional mode.
[17:16]	**MODESW0**	**Mode Switch0 Point** Mode at the switch0 point. 00 = Input mode. 01 = Output mode. 10 = Open-drain mode. 11 = Quasi-bidirectional mode.
[15:12]	**SW1CP**	**Switch1 Check Point** 0000 = No use. 0001 = CHECK POINT 0.

		0010 = CHECK POINT 1. 0011 = CHECK POINT 2. 0100 = CHECK POINT 3. 0101 = CHECK POINT 4. 0110 = CHECK POINT 5. 0111 = CHECK POINT 6. 1000 = CHECK POINT 7. Others = Reserved.
[11:8]	**SW0CP**	**Switch0 Check Point** 0000 = No use. 0001 = CHECK POINT 0. 0010 = CHECK POINT 1. 0011 = CHECK POINT 2. 0100 = CHECK POINT 3. 0101 = CHECK POINT 4. 0110 = CHECK POINT 5. 0111 = CHECK POINT 6. 1000 = CHECK POINT 7. Others = Reserved.
[7:6]	**Reserved**	Reserved.
[5:4]	**INTERVAL**	**Interval Output** The output of PSIO when slot controller stops counting and IDLE (PSIO_SCnCTL[25]) is 0. 00 = Low level. 01 = High level. 10 = Last output. 11 = Toggle. **Note 1:** Only when IOMODE is not input mode, then this register is effective. **Note 2:** This bit is effective only when IDLE(PSIO_SCnCTL[25]) is low.
[3:2]	**INITIAL**	**Initial Output** The output state of PSIO when slot controller stops counting and IDLE (PSIO_SCnCTL[25]) is 1. 00 = Low level. 01 = High level. 10 = Last output. 11 = Toggle. **Note 1:** Only when IOMODE is not input mode, this register is effective. **Note 2:** This bit is effective only when IDLE(PSIO_SCnCTL[25]) is high.
[1:0]	**IOMODE**	**I/O Mode** I/O mode state represent the I/O state when slot controller has not started counting or slot controller has started counting but has not cross the switch I/O mode check point. 00 = Input mode. 01 = Output mode. 10 = Open-drain mode. 11 = Quasi-bidirectional mode. **Note 1:** When slot controller stops counting, it will switch to the current I/O mode setting. **Note 2:** When PSIO uses quasi-bidirectional mode or open-drain mode to

		trigger slot controller, the initial or interval output state needs to be set output high level, or the pin will not be triggered.

PSIOn Data Control Register (PSIOn_DATCTL)

Register	Offset	R/W	Description	Reset Value
PSIOn_DATCTL **n=0,1..7**	PSIO_BA+0x44+(0x20 * n)	R/W	PSIOn Data Control Register	0x3300_0000

31	30	29	28	27	26	25	24
Reserved		INDEPTH		Reserved		OUTDEPTH	
23	22	21	20	19	18	17	16
Reserved							ORDER
15	14	13	12	11	10	9	8
Reserved			INDATWD				
7	6	5	4	3	2	1	0
Reserved			OUTDATWD				

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**INDEPTH**	**Input Data Depth** Represent the data depth of the input buffer, when data width is larger than 16-bit, this setting can be ignored. When the data width is between 9-bit and 16 bit, 0 = INDEPTH[0], the data depth is 1. 1 = INDEPTH[0], the data depth is 2. When the data width is less than or equal to 8-bit, 0 = INDEPTH, the data depth is 1. 1 = INDEPTH, the data depth is 2. 2 = INDEPTH, the data depth is 3. 3 = INDEPTH, the data depth is 4. **Note 1:** The input data depth impacts when the input data full flag and input data over flow flag is set to 1. **Note 2:** There is no difference of data depth no matter using software program data or PDMA program data.
[27:26]	**Reserved**	Reserved.
[25:24]	**OUTDEPTH**	**Output Data Depth** Represent the data depth of the output buffer, when data width is larger than 16-bit, this setting can be ignored. When the data width is between 9-bit and 16 bit, 0 = OUTDEPTH [0], the data depth is 1. 1 = OUTDEPTH [0], the data depth is 2. When the data width is less than or equal to 8-bit, 0 = OUTDEPTH, the data depth is 1. 1 = OUTDEPTH, the data depth is 2. 2 = OUTDEPTH, the data depth is 3. 3 = OUTDEPTH, the data depth is 4.

		Note 1: The output data depth impacts when the output data empty flag and output under flow data flag is set to 1. **Note 2:** There is no difference of data depth no matter using software program data or PDMA program data.
[23:17]	**Reserved**	Reserved.
[16]	**ORDER**	**Order** The order of output data and input data. 0 = Data transfer start form LSB. 1 = Data transfer start form MSB.
[15:13]	**Reserved**	Reserved.
[12:8]	**INDATWD**	**Input Data Width** Indicate the data width of INPUT DATA register. Input data size = INDATWD +1. e.g. 5'b00000 = 1 bit. 5'b11111 = 32 bit.
[7:5]	**Reserved**	Reserved.
[4:0]	**OUTDATWD**	**Output Data Width** Indicate the data width of OUTPUT DATA register. Output data size = OUTDATWD +1. e.g. 5'b00000 = 1 bit. 5'b11111 = 32 bit.

PSIOn Input Status Register (PSIOn_INSTS)

Register	Offset	R/W	Description	Reset Value
PSIOn_INSTS **n=0,1..7**	PSIO_BA+0x48+(0x20 * n)	R	PSIOn Input Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INSTS							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**INSTS**	**Input Status** Status input buffer. This register can be read clear. **Note:** When the valid bit is set, the valid bits number of INSTS is equal to the number of check points from the previous time INSTS update to the current INSTS update.

PSIOn Input Data Register (PSIOn_INDAT)

Register	Offset	R/W	Description	Reset Value
PSIOn_INDAT **n=0,1..7**	PSIO_BA+0x4C+(0x20 * n)	R	PSIOn Input Data Register	0x0000_0000

31	30	29	28	27	26	25	24
INDAT							
23	22	21	20	19	18	17	16
INDAT							
15	14	13	12	11	10	9	8
INDAT							
7	6	5	4	3	2	1	0
INDAT							

Bits	Description	
[31:0]	**INDAT**	**Input Data Buffer** This register can be read clear. **Note:** The input data sample time is according to the slot length. The sampling time is near 3/4 slot. When the slot length is 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, PSIO sample input data when the slot controller count to 1, 2, 2, 3, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9.

PSIOn Output Data Register (PSIOn_OUTDAT)

Register	Offset	R/W	Description	Reset Value
PSIOn_OUTDAT **n=0,1..7**	PSIO_BA+0x50+(0x20 * n)	W	PSIOn Output Data Register	0x0000_0000

31	30	29	28	27	26	25	24
OUTDAT							
23	22	21	20	19	18	17	16
OUTDAT							
15	14	13	12	11	10	9	8
OUTDAT							
7	6	5	4	3	2	1	0
OUTDAT							

Bits	Description	
[31:0]	**OUTDAT**	**Output Data Buffer** This field is used to configure output data.

PSIOn Check Point Control Register 0 (PSIOn_CPCTL0)

Register	Offset	R/W	Description	Reset Value
PSIOn_CPCTL0 **n=0,1..7**	PSIO_BA+0x54+(0x20 * n)	R/W	PSIOn Check Point Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
CKPT7				CKPT6			
23	22	21	20	19	18	17	16
CKPT5				CKPT4			
15	14	13	12	11	10	9	8
CKPT3				CKPT2			
7	6	5	4	3	2	1	0
CKPT1				CKPT0			

Bits	Description	
[30:28]	**CKPT7**	**Check Point 7** This field is used to link check point and slot controller slot. 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved. **Note 1:** If there are two check points that select the same SLOT, the pin will follow settings of the smaller check point number. **Note 2:** The correlated SLOT should be filled in order from SLOT0 to SLOT7, or the check point action will not be triggered.
[26:24]	**CKPT6**	**Check Point 6** 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved.

		Note 1: If there are two check points that select the same SLOT, the pin will follow settings of the smaller check point number. **Note 2:** The correlated SLOT should be filled in order from SLOT0 to SLOT7, or the check point action will not be triggered.
[22:20]	**CKPT5**	**Check Point 5** 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved. **Note 1:** If there are two check points that select the same SLOT, the pin will follow settings of the smaller check point number. **Note 2:** The correlated SLOT should be filled in order from SLOT0 to SLOT7, or the check point action will not be triggered.
[18:16]	**CKPT4**	**Check Point 4** 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved. **Note 1:** If there are two check points that select the same SLOT, the pin will follow settings of the smaller check point number. **Note 2:** The correlated SLOT should be filled in order from SLOT0 to SLOT7, or the check point action will not be triggered.
[14:12]	**CKPT3**	**Check Point 3** 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved. **Note 1:** If there are two check points that select the same SLOT, the pin will follow settings of the smaller check point number. **Note 2:** The correlated SLOT should be filled in order from SLOT0 to SLOT7, or the check point action will not be triggered.
[10:8]	**CKPT2**	**Check Point 2**

		0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved. **Note 1:** If there are two check points that select the same SLOT, the pin will follow settings of the smaller check point number. **Note 2:** The correlated SLOT should be filled in order from SLOT0 to SLOT7, or the check point action will not be triggered.
[6:4]	**CKPT1**	**Check Point 1** 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved. **Note 1:** If there are two check points that select the same SLOT, the pin will follow settings of the smaller check point number. **Note 2:** The correlated SLOT should be filled in order from SLOT0 to SLOT7, or the check point action will not be triggered.
[2:0]	**CKPT0**	**Check Point 0** 0000 = No use. 0001 = SLOT0. 0010 = SLOT1. 0011 = SLOT2. 0100 = SLOT3. 0101 = SLOT4. 0110 = SLOT5. 0111 = SLOT6. 1000 = SLOT7. Others = Reserved. **Note 1:** If there are two check points that select the same SLOT, the pin will follow settings of the smaller check point number. **Note 2:** The correlated SLOT should be filled in order from SLOT0 to SLOT7, or the check point action will not be triggered.

PSIOn Check Point Control Register 1 (PSIOn_CPCTL1)

Register	Offset	R/W	Description	Reset Value
PSIOn_CPCTL1 n=0,1..7	PSIO_BA+0x58+(0x20 * n)	R/W	PSIOn Check Point Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	CKPT7ACT			Reserved	CKPT6ACT		
23	22	21	20	19	18	17	16
Reserved	CKPT5ACT			Reserved	CKPT4ACT		
15	14	13	12	11	10	9	8
Reserved	CKPT3ACT			Reserved	CKPT2ACT		
7	6	5	4	3	2	1	0
Reserved	CKPT1ACT			Reserved	CKPT0ACT		

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**CKPT7ACT**	**Check Point 7 Action** Select check point action at check point7. 000 = Output level low. 001 = Output level high. 010 = Output from data buffer. 011 = Output toggle. 100 = Input data buffer. 101 = Input status. 110 = Input status record and update. 111 = Reserved. **Note:** Pin action must meet the correlated I/O mode (PSIOn_GENCTL[1:0]).
[27]	**Reserved**	Reserved.
[26:24]	**CKPT6ACT**	**Check Point 6 Action** Select check point action at check point6. 000 = Output level low. 001 = Output level high. 010 = Output from data buffer. 011 = Output toggle. 100 = Input data buffer. 101 = Input status. 110 = Input status record and update. 111 = Reserved. **Note:** Pin action must meet the correlated I/O mode (PSIOn_GENCTL[1:0]).
[23]	**Reserved**	Reserved.
[22:20]	**CKPT5ACT**	**Check Point 5 Action**

		Select check point action at check point5. 000 = Output level low. 001 = Output level high. 010 = Output from data buffer. 011 = Output toggle. 100 = **I**nput data buffer. 101 = Input status. 110 = Input status record and update. 111 = Reserved. **Note:** Pin action must meet the correlated I/O mode (PSIOn_GENCTL[1:0]).
[19]	**Reserved**	Reserved.
[18:16]	**CKPT4ACT**	**Check Point 4 Action** Select check point action at check point4. 000 = Output level low. 001 = Output level high. 010 = Output from data buffer. 011 = Output toggle. 100 = Input data buffer. 101 = Input status. 110 = Input status record and update. 111 = Reserved. **Note:** Pin action must meet the correlated I/O mode (PSIOn_GENCTL[1:0]).
[15]	**Reserved**	Reserved.
[14:12]	**CKPT3ACT**	**Check Point 3 Action** Select check point action at check point3. 000 = Output level low. 001 = Output level high. 010 = Output from data buffer. 011 = Output toggle. 100 = Input data buffer. 101 = Input status. 110 = Input status record and update. 111 = Reserved. **Note:** Pin action must meet the correlated I/O mode (PSIOn_GENCTL[1:0]).
[11]	**Reserved**	Reserved.
[10:8]	**CKPT2ACT**	**Check Point 2 Action** Select check point action at check point2. 000 = Output level low. 001 = Output level high. 010 = Output from data buffer. 011 = Output toggle. 100 = Input data buffer. 101 = Input status. 110 = Input status record and update. 111 = Reserved. **Note:** Pin action must meet the correlated I/O mode (PSIOn_GENCTL[1:0]).

[7]	**Reserved**	Reserved.
[6:4]	**CKPT1ACT**	**Check Point 1 Action** Select check point action at check point1. 000 = Output level low. 001 = Output level high. 010 = Output from data buffer. 011 = Output toggle. 100 = Input data buffer. 101 = Input status. 110 = Input status record and update. 111 = Reserved. **Note:** Pin action must meet the correlated I/O mode (PSIOn_GENCTL[1:0]).
[3]	**Reserved**	Reserved.
[2:0]	**CKPT0ACT**	**Check Point 0 Action** Select check point action at check point0. 000 = Output level low. 001 = Output level high. 010 = Output from data buffer. 011 = Output toggle. 100 = Input data buffer. 101 = Input status. 110 = Input status record and update. 111 = Reserved. **Note:** Pin action must meet the correlated I/O mode (PSIOn_GENCTL[1:0]).

6.33 USB 1.1 Device Controller (USBD)

6.33.1 Overview

There is one set of USB 2.0 full-speed device controller and transceiver in this device. It is compliant with USB 2.0 full-speed device specification and supports Control/Bulk/Interrupt/Isochronous transfer types.

In this device controller, there are two main interfaces: the APB bus and USB bus which comes from the USB PHY transceiver. For the APB bus, the CPU can program control registers through it. There are 1.5K byte sizes internal SRAM as data buffer in this controller. For IN or OUT transfer, it is necessary to write data to SRAM or read data from SRAM through the APB interface or SI.E. User needs to set the effective starting address of SRAM for each endpoint buffer through buffer segmentation register (USBD_BUFSEG0~24).

There are 25 endpoints in this controller. Each of the endpoint can be configured as IN or OUT endpoint. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. The block of “Endpoint Control” is also used to manage the data sequential synchronization, endpoint states, current start address, transaction status, and data buffer status for each endpoint.

There are five different interrupt events in this controller. They are the no-event-wake-up, device plug-in or plug-out event, USB events, such as IN ACK, OUT ACK, and BUS events, such as suspend and resume and SOF event, receive Start of Frame(SOF) packet in every 1ms event. Any event will cause an interrupt, and users just need to check the related event flags in interrupt event status register (USBD_INTSTS) to acknowledge what kind of interrupt occurring, and then check the related USB Endpoint Status Register (USBD_EPSTS0~3) to acknowledge what kind of event occurring in this endpoint.

A software-disconnect function is also supported for this USB controller. It is used to simulate the disconnection of this device from the host. If user enables SE0 bit (USBD_SE0), the USB controller will force the output of USB_D+ and USB_D- to level low and its function is disabled. After disabling the SE0 bit, host will enumerate the USB device again.

For more information on the Universal Serial Bus, please refer to *Universal Serial Bus Specification Revision 1.1.*

6.33.2 Features

- Compliant with USB 2.0 Full-Speed specification
- Provides 1 interrupt vector with 5 different interrupt events (NEVWK, VBDET, USB, BUS and SOF)
- Supports Control/Bulk/Interrupt/Isochronous transfer type
- Supports suspend function when no bus activity existing for 3ms
- Supports 25 endpoints for configurable Control/Bulk/Interrupt/Isochronous transfer types and maximum 1.5 Kbyte buffer size
- Provides remote wake-up capability
- Supports double buffer function

6.33.3 Block Diagram

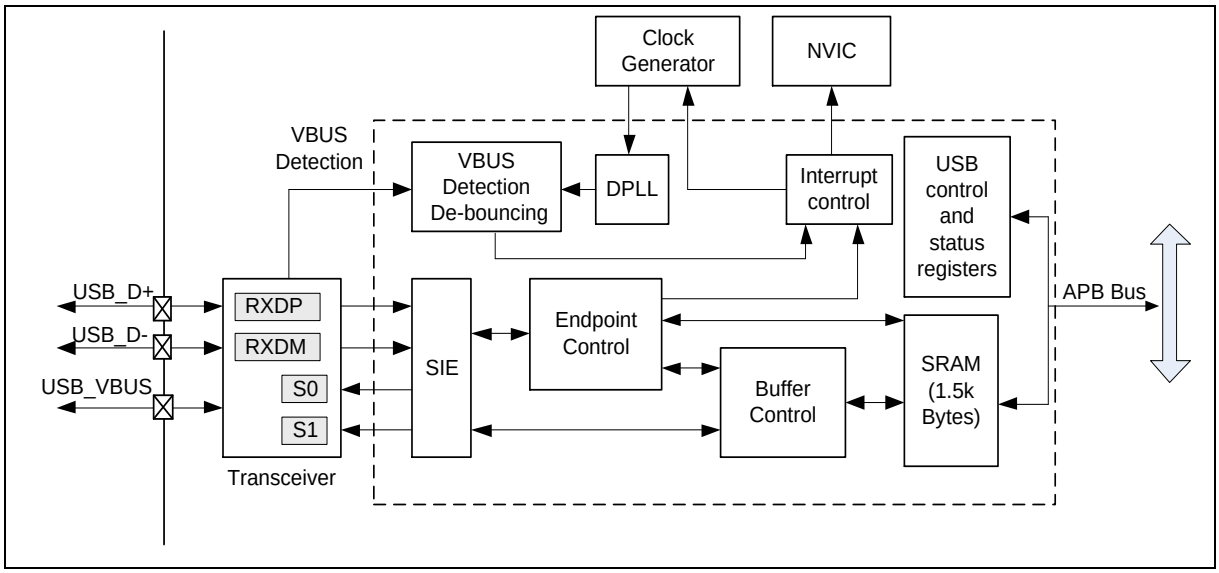


Figure 6.33-1 USB Block Diagram

6.33.4 Basic Configuration

The role of USB frame is determined by USBROLE (SYS_USBPHY[1:0]). The internal USB 3.3V LDO can be enabled by USBEN (SYS_USBPHY[8]). These two configurations are write-protection bits. Before writing to these bits, user must disable the register protection function. Refer to the description of SYS_REGLCTL register for details. The USBD clock source is derived from HIRC48M or PLL. If user would like to achieve crystal-less, HIRC must be selected as USB clock source, or user has to use the PLL related configurations with crystal before USB device controller is enabled. Set the USBDCKEN (CLK_APBCLK0[27]) bit to enable USBD clock and 4-bit pre-scaler USBDIV (CLK_CLKDIV0[7:4]) to generate the proper USBD clock rate.

Note: USBD peripheral clock frequency needs to be greater than 15 MHz when USBD receives/transmits data from/to the host.

6.33.4.1 USB Device 1.1 Basic Configuration

- Clock source Configuration
  - Enable HIRC48MEN (CLK_PWRCTL[18]) or set PLL controller (CLK_PLLCTL)
  - Select the clock divider number of USBD peripheral clock on USBDIV(CLK_CLKDIV0[7:4])
  - Enable USBD peripheral clock in USBDCKEN (CLK_APBCLK0[27]).
- Reset Configuration
  - Reset USBD controller in USBDRST (SYS_IPRST1[27]).

6.33.5 Functional Description

6.33.5.1 Serial Interface Engine (SIE)

The SIE is the front-end of the device controller and handles most of the USB packet protocol. The SIE typically comprehends signaling up to the transaction level. The functions that it handles could include:

- Packet recognition and transaction sequencing
- SOP, EOP, RESET, RESUME signal detection/generation
- Clock/Data separation

- NRZI Data encoding/decoding and bit-stuffing
- CRC generation and checking (for Token and Data)
- Packet ID (PID) generation and checking/decoding
- Serial-Parallel/Parallel-Serial conversion

6.33.5.2 Endpoint Control

This controller supports 25 endpoints. Each of the endpoint can be configured as Control, Bulk, Interrupt, or Isochronous transfer type. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. It is also used to manage the data sequential synchronization, endpoint state control, current endpoint start address, current transaction status, and data buffer status in each endpoint.

6.33.5.3 Digital Phase Lock Loop (DPLL)

The bit rate of USB data is 12 MHz. The DPLL uses the 48 MHz which comes from the clock controller to lock the input data RXDP and RXDM. The 12 MHz bit rate clock is also converted from DPLL.

6.33.5.4 V_{BUS} Detection De-bouncing

A USB device may be plugged-in or plugged-out from the USB host. To monitor the state of a USB device when it is detached from the USB host, the device controller provides hardware de-bouncing for USB V_{BUS} detection interrupt to avoid bounce problems on USB plug-in or plug-out. V_{BUS} detection interrupt appears about 10ms later than USB plug-in or plug-out. User can acknowledge USB plug-in/plug-out by reading USBD_VBUSDET register. The VBUSDET flag represents the current state on the bus without de-bouncing. If VBUSDET is 1, it means the USB cable is plugged-in. If user polls the flag to check USB state, software de-bouncing must be added if needed.

6.33.5.5 Interrupt Control

This USB provides 1 interrupt vector with 5 interrupt events (NEVWK, VBDET, USB, BUS and SOF). The NEVWK event occurs after waking up the system from Power-down mode (the Power-down mode function is defined in system power-down control register, CLK_PWRCTL). The VBDET event is used for USB plug-in or plug-out. The USB event notifies users of some USB requests, such as IN ACK, OUT ACK. And the BUS event notifies users of some bus events, such as suspend and resume. The related bits must be set in the interrupt enable register (USBD_INTEN) of USB Device Controller to enable USB interrupts.

NEVWK interrupt is only presented when no the other USB interrupt events happened more than 20ms after the chip is woken up from Power-down mode. After the chip enters Power-down mode, any change on USB_VBUS, USB_D+ and USB_D- can wake up this chip if USB wake-up function is enabled. If this change is not intentionally, no interrupt but NEVWK interrupt will occur. After woken up by USB, this interrupt will occur when no the other USB interrupt events are presented for more than 20ms. Figure 6.33-2 shows the control flow of wake-up interrupt.

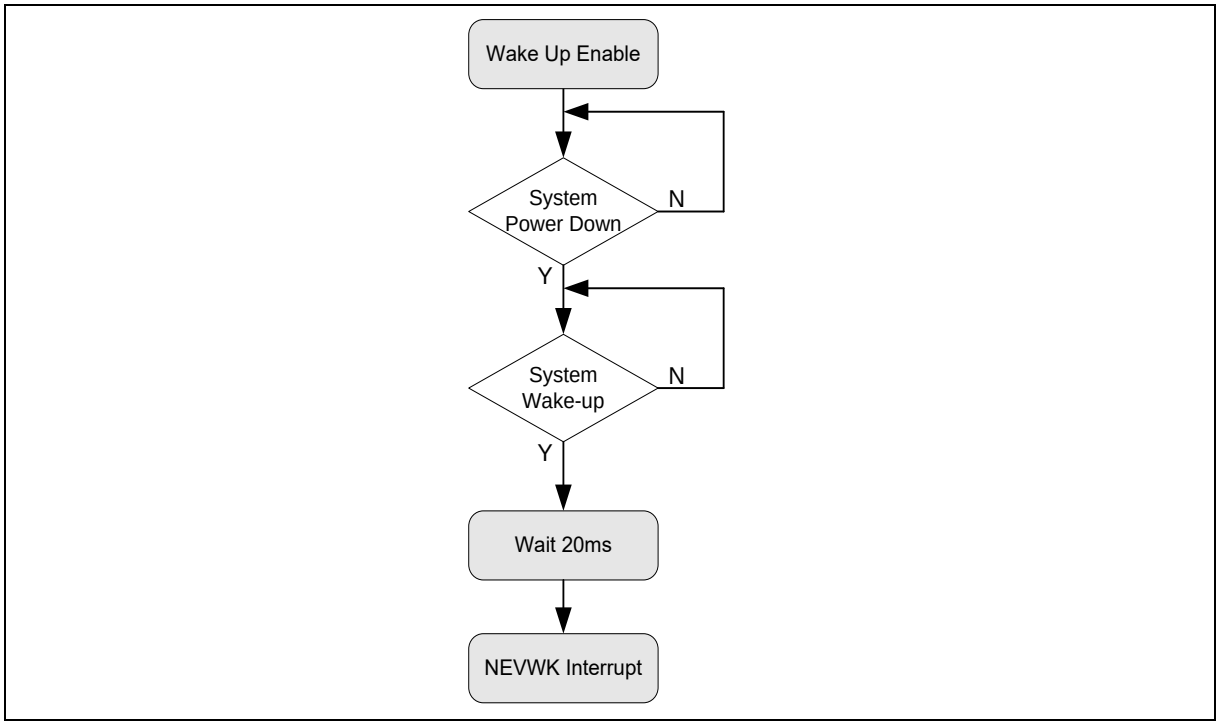


Figure 6.33-2 NEVWK Interrupt Operation Flow

The USB interrupt is used to notify users of any USB event on the bus, and user can read EPSTS (USBD_EPSTS0~3) and EPEVT0~24 (USBD_INTSTS[27:16] or USBD_EPINTSTS[24:0]) to take necessary responses.

Same as USB interrupt, BUS interrupt notifies users of some bus events, like USB reset, suspend, time-out, and resume. A user can read USBD_ATTR to acknowledge bus events.

6.33.5.6 Power Saving

User can write 0 to USBD_ATTR[4] to disable PHY under special circumstances, like suspend, to conserve power.

6.33.5.7 Buffer Control

There is 1.5 Kbytes SRAM in the controller and the 25 endpoints share this buffer. User shall configure each endpoint's effective starting address in the buffer segmentation register before the USB function active. The "Buffer Control" block is used to control each endpoint's effective starting address and its SRAM size is defined in the USBD_MXPLD0~24 register.

Figure 6.33-3 depicts the starting address for each endpoint according to the content of USBD_BUFSEG0~24 and USBD_MXPLD0~24 registers. If the USBD_BUFSEG0 is programmed as 0x08h and USBD_MXPLD0 is set as 0x40h, the SRAM size of endpoint 0 starts from USBD_BA+0x808h and ends in USBD_BA+0x848h.

Note: The USBD SRAM base is USBD_BA+0x800h.

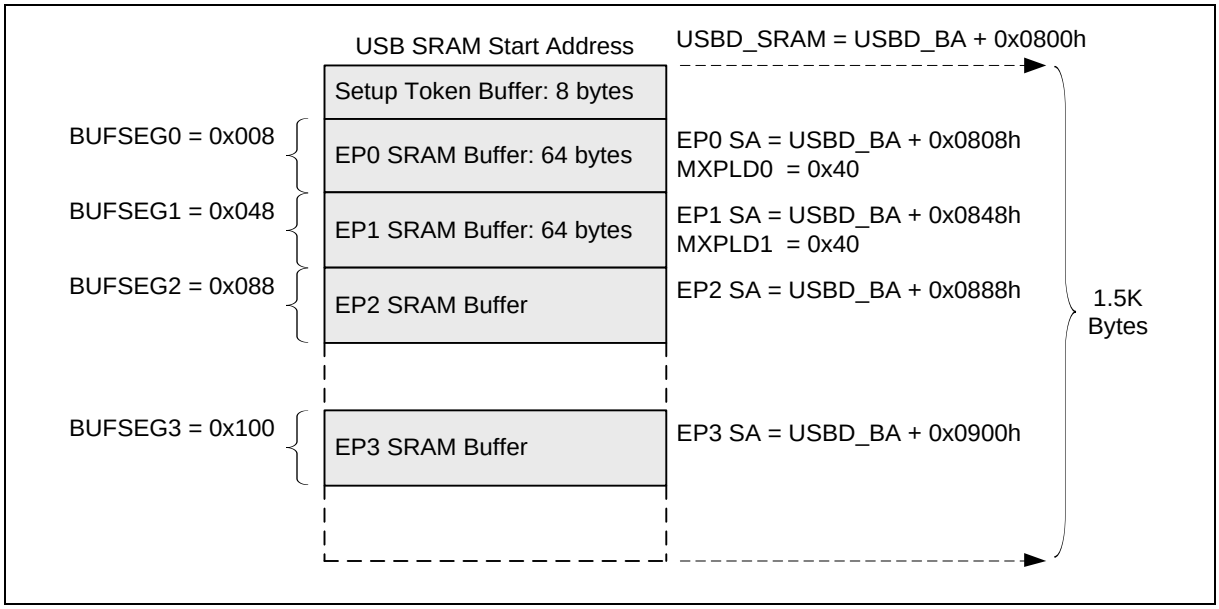


Figure 6.33-3 Endpoint SRAM Structure

6.33.5.8 Handling Transactions with USB Device Peripheral

User can use interrupt or polling USBD_INTSTS to monitor the USB transactions. When transactions occur, USBD_INTSTS will be set by hardware and send an interrupt request to CPU (if related interrupt enabled), or user can polling USBD_INTSTS to get these events without interrupt. The following is the control flow with interrupt enabled.

When USB host has requested data from a device controller, user needs to prepare related data in the specified endpoint buffer in advance. After buffering the required data, user needs to write the actual data length in the specified USBD_MXPLD0~24 register. Once this register is written, the internal signal "In_Rdy" will be asserted and the buffering data will be transmitted immediately after receiving associated IN token from Host. Note that after transferring the specified data, the signal "In_Rdy" will de-assert automatically by hardware.

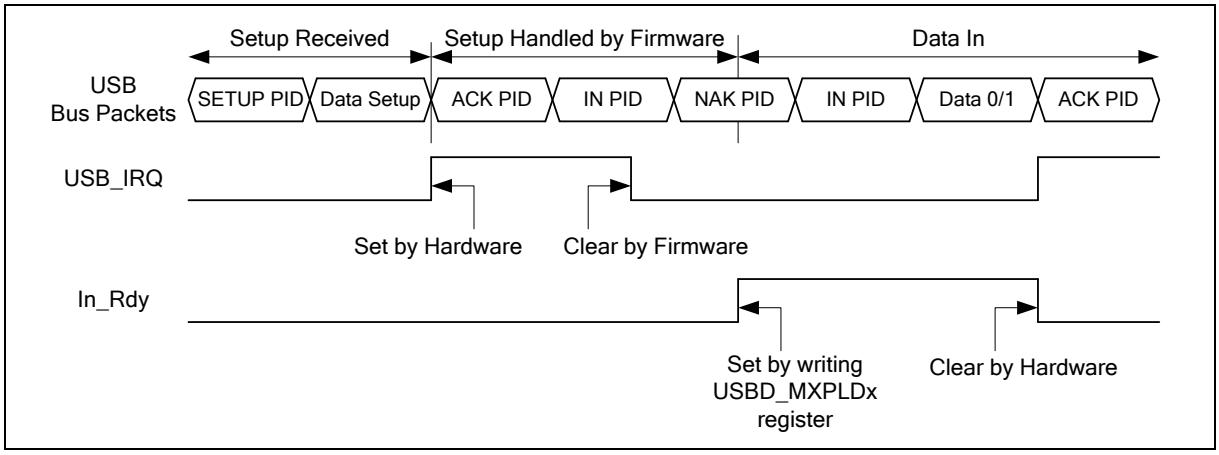


Figure 6.33-4 Setup Transaction Followed by Data IN Transaction

Alternatively, when USB host wants to transmit data to the OUT endpoint in the device controller, hardware will buffer these data to the specified endpoint buffer. After this transaction is completed, hardware will record the data length in specified USBD_MXPLD0~24 register and de-assert the internal signal "Out_Rdy". This will avoid hardware accepting next transaction until user moves out the current data in the related endpoint buffer. Once users have processed this transaction, the specified USBD_MXPLD0~24 register needs to be written by firmware to assert the signal "Out_Rdy" again to accept the next transaction.

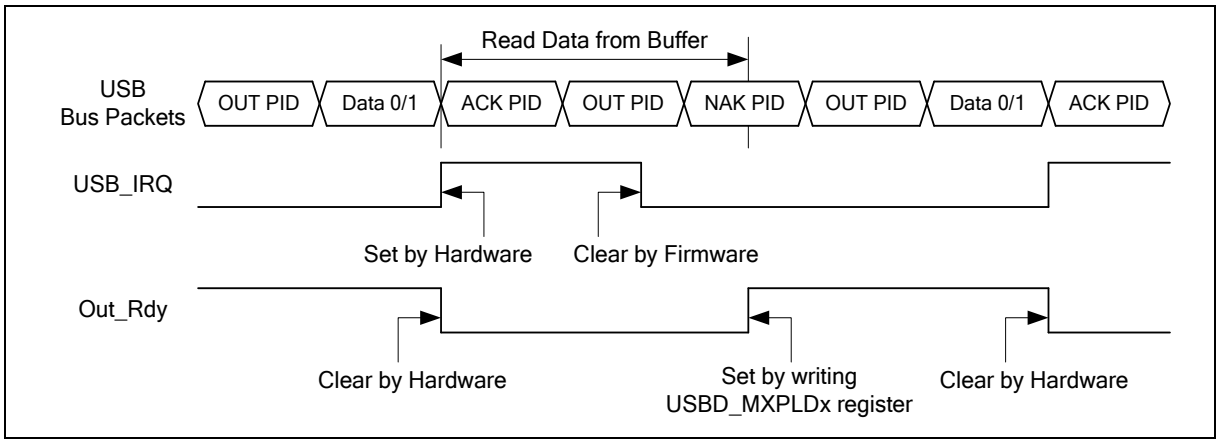


Figure 6.33-5 Data Out Transfer

6.33.5.9 Link Power Management (LPM)

Link Power Management (LPM) which is similar to the suspend/resume function, but has transitional

latencies of tens of microseconds between power states (instead of three to greater than 20 milliseconds latencies of the USB2.0 suspend/resume)

Through the LPM mechanism, Host lets the device state quickly change from an enabled state (L0) to a new sleep state (L1). For detailed definition of L0 and L1 state, see Table 6.33-1. The register USBD_ATTR and USBD_LPMATTR can let user know the current power state for LPM mechanism.

LPM State	Description
L0 (On)	In this state, the port is enabled for propagation of transaction signaling traffic. A port in L0 is either actively transmitting or receiving data (L0-Active) or able to do so but not currently transmitting or receiving information (L0-Idle). While in this state, Start-of-Frame (SOF) packets are issued by the host at a rate corresponding to the speed of the client device.
L1 (Sleep)	L1 is similar to L2 (below) but supports finer granularity in use. When in L1, the line state is identical to L2. Entry to L1 is started by a request to a hub or host port to transition to L1. A LPM transaction is sent to the downstream device. The requested transition can only occur if the device response with an ACK handshake. Exit from L1 is via remote wake, resume signaling, reset signaling or disconnect. L1 does not impose any specific power draw requirements (from V_{BUS}) on the attached device as L2 does. Either the host or device can initiate resume signaling when in L1. Although the signaling levels of resume are the same as L2, the duration of the signaling and transitional latencies associated with the L1 to L0 transition are much shorter.
L2 (Suspend)	This is the formalized name for USB 2.0 Suspend, Entry to L2 is nominally triggered by a command to a hub or host port to transition to suspend. The device discovers the suspend condition via observing 3ms of inactivity. The resultant line state is either Low or Full-speed idle. L2 also imposes power draw requirements (from V_{BUS}) on the attached device. Exit from this state is via remote wake, resume signaling, reset signaling or disconnect.
L3 (Off)	In this state, the port is not capable of performing any data signaling. It corresponds to the powered-off, disconnected, and disabled states.

Table 6.33-1 USB Link Power Management (Lx) States

For the state transaction process please refer to Figure 6.33-6, and for more information on the USB Link Power Management (LPM), please refer to USB2.0 Link Power Management ECN (Engineering Change Notice).

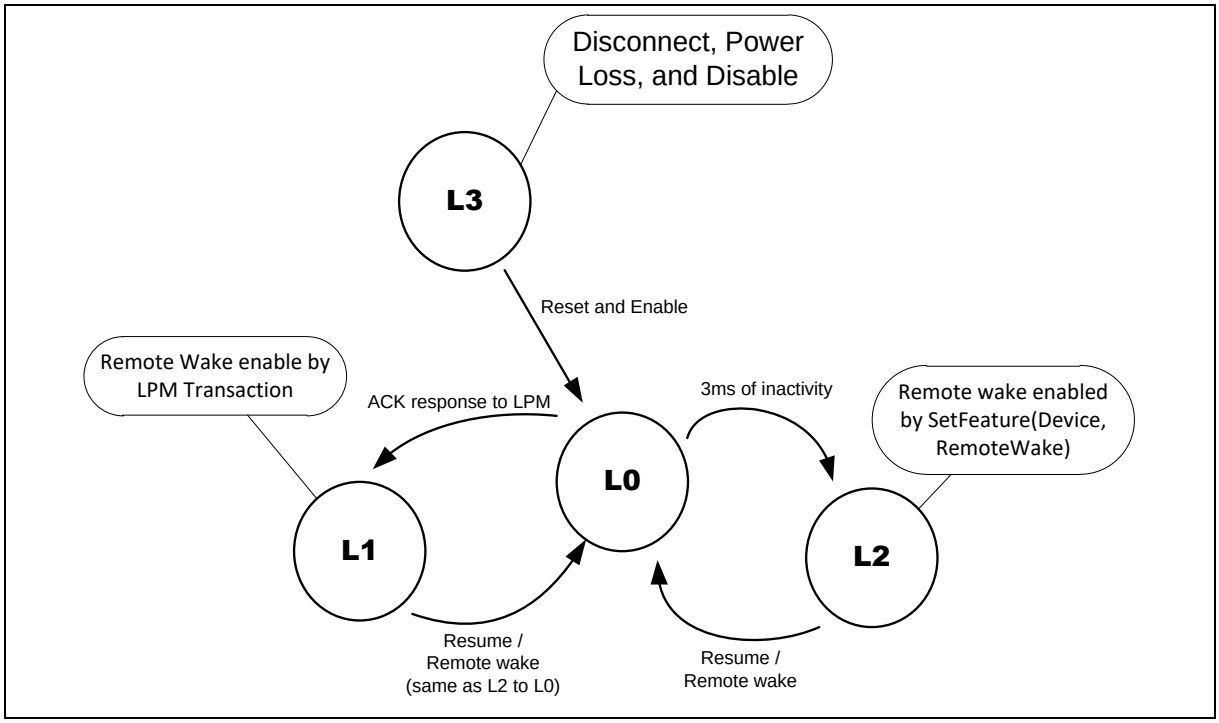


Figure 6.33-6 LPM State Transition Diagram

6.33.5.10 Double Buffer Transfer

When user sets double buffer to continuously transfer or receive data, hardware will automatically toggle endpoint buffer to access data after transaction is finished. User has much time to process the last transaction data and set the next transaction data to reduce the chances that the device replies NAK to the host.

There are 25 endpoints buffer. User can set single buffer + double buffer * 2 = 25, and set double buffer by “DBEN”, ” DBTGACTIVE” in USB_CFG0~24. The double buffer function supports IN, OUT and ISO transaction.

The USB device receives endpoint number and token to decide which endpoint buffer will be used. If setting double buffer (DBEN = 1), active endpoint (DBTGACTIVE =1) will transfer or receive data.

When double buffer is enabled, DBTGACTIVE will be toggled automatically by hardware after successful transfer, and DSQSYNC will remain stable.

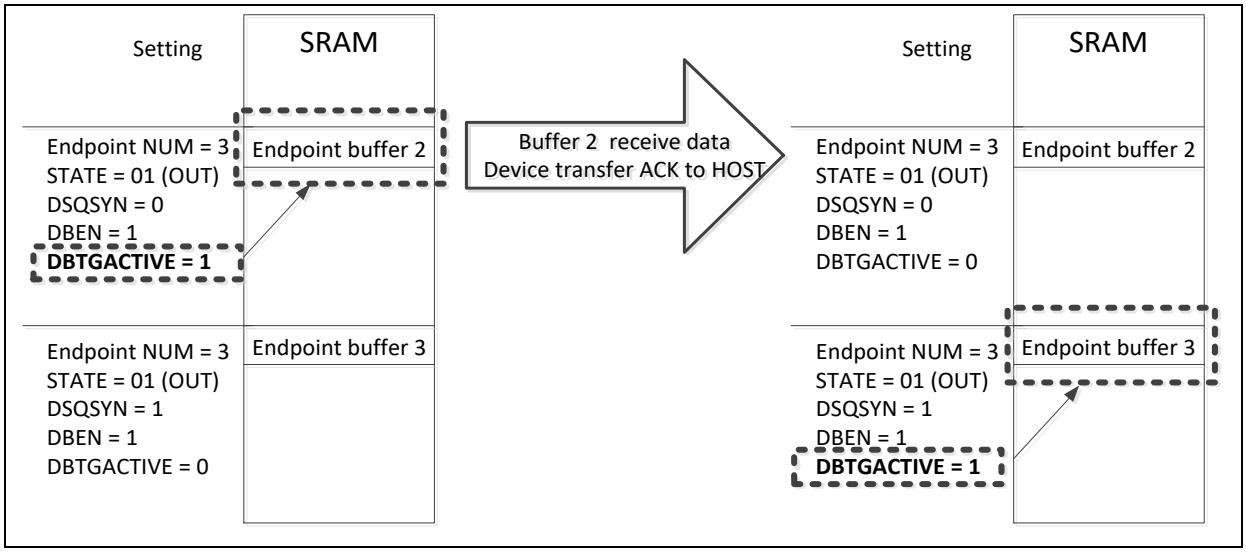


Figure 6.33-7 Double Buffer Change Process

6.33.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
USBD Base Address: **USBD_BA = 0x400C_0000**				
USBD_INTEN	USBD_BA+0x000	R/W	USB Device Interrupt Enable Register	0x0000_0000
USBD_INTSTS	USBD_BA+0x004	R/W	USB Device Interrupt Event Status Register	0x0000_0000
USBD_FADDR	USBD_BA+0x008	R/W	USB Device Function Address Register	0x0000_0000
USBD_EPSTS	USBD_BA+0x00C	R	USB Device Endpoint Status Register	0x0000_0000
USBD_ATTR	USBD_BA+0x010	R/W	USB Device Bus Status and Attribution Register	0x0000_0040
USBD_VBUSDET	USBD_BA+0x014	R	USB Device V_{BUS} Detection Register	0x0000_0000
USBD_STBUFSEG	USBD_BA+0x018	R/W	SETUP Token Buffer Segmentation Register	0x0000_0000
USBD_EPSTS0	USBD_BA+0x020	R	USB Device Endpoint Status Register 0	0x0000_0000
USBD_EPSTS1	USBD_BA+0x024	R	USB Device Endpoint Status Register 1	0x0000_0000
USBD_EPSTS2	USBD_BA+0x028	R	USB Device Endpoint Status Register 2	0x0000_0000
USBD_EPSTS3	USBD_BA+0x02C	R	USB Device Endpoint Status Register 3	0x0000_0000
USBD_EPINTSTS	USBD_BA+0x030	R/W	USB Device Endpoint Interrupt Event Status Register	0x0000_0000
USBD_LPMATTR	USBD_BA+0x088	R	USB LPM Attribution Register	0x0000_0000
USBD_FN	USBD_BA+0x08C	R	USB Frame Number Register	0x0000_0XXX
USBD_SE0	USBD_BA+0x090	R/W	USB Device Drive SE0 Control Register	0x0000_0001
USBD_BUFSEG0	USBD_BA+0x500	R/W	Endpoint 0 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD0	USBD_BA+0x504	R/W	Endpoint 0 Maximal Payload Register	0x0000_0000
USBD_CFG0	USBD_BA+0x508	R/W	Endpoint 0 Configuration Register	0x0000_0000
USBD_CFGP0	USBD_BA+0x50C	R/W	Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG1	USBD_BA+0x510	R/W	Endpoint 1 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD1	USBD_BA+0x514	R/W	Endpoint 1 Maximal Payload Register	0x0000_0000
USBD_CFG1	USBD_BA+0x518	R/W	Endpoint 1 Configuration Register	0x0000_0000
USBD_CFGP1	USBD_BA+0x51C	R/W	Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG2	USBD_BA+0x520	R/W	Endpoint 2 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD2	USBD_BA+0x524	R/W	Endpoint 2 Maximal Payload Register	0x0000_0000
USBD_CFG2	USBD_BA+0x528	R/W	Endpoint 2 Configuration Register	0x0000_0000
USBD_CFGP2	USBD_BA+0x52C	R/W	Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000

USBD_BUFSEG3	USBD_BA+0x530	R/W	Endpoint 3 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD3	USBD_BA+0x534	R/W	Endpoint 3 Maximal Payload Register	0x0000_0000
USBD_CFG3	USBD_BA+0x538	R/W	Endpoint 3 Configuration Register	0x0000_0000
USBD_CFGP3	USBD_BA+0x53C	R/W	Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG4	USBD_BA+0x540	R/W	Endpoint 4 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD4	USBD_BA+0x544	R/W	Endpoint 4 Maximal Payload Register	0x0000_0000
USBD_CFG4	USBD_BA+0x548	R/W	Endpoint 4 Configuration Register	0x0000_0000
USBD_CFGP4	USBD_BA+0x54C	R/W	Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG5	USBD_BA+0x550	R/W	Endpoint 5 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD5	USBD_BA+0x554	R/W	Endpoint 5 Maximal Payload Register	0x0000_0000
USBD_CFG5	USBD_BA+0x558	R/W	Endpoint 5 Configuration Register	0x0000_0000
USBD_CFGP5	USBD_BA+0x55C	R/W	Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG6	USBD_BA+0x560	R/W	Endpoint 6 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD6	USBD_BA+0x564	R/W	Endpoint 6 Maximal Payload Register	0x0000_0000
USBD_CFG6	USBD_BA+0x568	R/W	Endpoint 6 Configuration Register	0x0000_0000
USBD_CFGP6	USBD_BA+0x56C	R/W	Endpoint 6 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG7	USBD_BA+0x570	R/W	Endpoint 7 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD7	USBD_BA+0x574	R/W	Endpoint 7 Maximal Payload Register	0x0000_0000
USBD_CFG7	USBD_BA+0x578	R/W	Endpoint 7 Configuration Register	0x0000_0000
USBD_CFGP7	USBD_BA+0x57C	R/W	Endpoint 7 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG8	USBD_BA+0x580	R/W	Endpoint 8 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD8	USBD_BA+0x584	R/W	Endpoint 8 Maximal Payload Register	0x0000_0000
USBD_CFG8	USBD_BA+0x588	R/W	Endpoint 8 Configuration Register	0x0000_0000
USBD_CFGP8	USBD_BA+0x58C	R/W	Endpoint 8 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG9	USBD_BA+0x590	R/W	Endpoint 9 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD9	USBD_BA+0x594	R/W	Endpoint 9 Maximal Payload Register	0x0000_0000
USBD_CFG9	USBD_BA+0x598	R/W	Endpoint 9 Configuration Register	0x0000_0000
USBD_CFGP9	USBD_BA+0x59C	R/W	Endpoint 9 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG10	USBD_BA+0x5A0	R/W	Endpoint 10 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD10	USBD_BA+0x5A4	R/W	Endpoint 10 Maximal Payload Register	0x0000_0000
USBD_CFG10	USBD_BA+0x5A8	R/W	Endpoint 10 Configuration Register	0x0000_0000

USBD_CFGP10	USBD_BA+0x5AC	R/W	Endpoint 10 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG11	USBD_BA+0x5B0	R/W	Endpoint 11 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD11	USBD_BA+0x5B4	R/W	Endpoint 11 Maximal Payload Register	0x0000_0000
USBD_CFG11	USBD_BA+0x5B8	R/W	Endpoint 11 Configuration Register	0x0000_0000
USBD_CFGP11	USBD_BA+0x5BC	R/W	Endpoint 11 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG12	USBD_BA+0x5C0	R/W	Endpoint 12 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD12	USBD_BA+0x5C4	R/W	Endpoint 12 Maximal Payload Register	0x0000_0000
USBD_CFG12	USBD_BA+0x5C8	R/W	Endpoint 12 Configuration Register	0x0000_0000
USBD_CFGP12	USBD_BA+0x5CC	R/W	Endpoint 12 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG13	USBD_BA+0x5D0	R/W	Endpoint 13 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD13	USBD_BA+0x5D4	R/W	Endpoint 13 Maximal Payload Register	0x0000_0000
USBD_CFG13	USBD_BA+0x5D8	R/W	Endpoint 13 Configuration Register	0x0000_0000
USBD_CFGP13	USBD_BA+0x5DC	R/W	Endpoint 13 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG14	USBD_BA+0x5E0	R/W	Endpoint 14 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD14	USBD_BA+0x5E4	R/W	Endpoint 14 Maximal Payload Register	0x0000_0000
USBD_CFG14	USBD_BA+0x5E8	R/W	Endpoint 14 Configuration Register	0x0000_0000
USBD_CFGP14	USBD_BA+0x5EC	R/W	Endpoint 14 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG15	USBD_BA+0x5F0	R/W	Endpoint 15 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD15	USBD_BA+0x5F4	R/W	Endpoint 15 Maximal Payload Register	0x0000_0000
USBD_CFG15	USBD_BA+0x5F8	R/W	Endpoint 15 Configuration Register	0x0000_0000
USBD_CFGP15	USBD_BA+0x5FC	R/W	Endpoint 15 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG16	USBD_BA+0x600	R/W	Endpoint 16 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD16	USBD_BA+0x604	R/W	Endpoint 16 Maximal Payload Register	0x0000_0000
USBD_CFG16	USBD_BA+0x608	R/W	Endpoint 16 Configuration Register	0x0000_0000
USBD_CFGP16	USBD_BA+0x60C	R/W	Endpoint 16 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG17	USBD_BA+0x610	R/W	Endpoint 17 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD17	USBD_BA+0x614	R/W	Endpoint 17 Maximal Payload Register	0x0000_0000
USBD_CFG17	USBD_BA+0x618	R/W	Endpoint 17 Configuration Register	0x0000_0000
USBD_CFGP17	USBD_BA+0x61C	R/W	Endpoint 17 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG18	USBD_BA+0x620	R/W	Endpoint 18 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD18	USBD_BA+0x624	R/W	Endpoint 18 Maximal Payload Register	0x0000_0000

USBD_CFG18	USBD_BA+0x628	R/W	Endpoint 18 Configuration Register	0x0000_0000
USBD_CFGP18	USBD_BA+0x62C	R/W	Endpoint 18 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG19	USBD_BA+0x630	R/W	Endpoint 19 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD19	USBD_BA+0x634	R/W	Endpoint 19 Maximal Payload Register	0x0000_0000
USBD_CFG19	USBD_BA+0x638	R/W	Endpoint 19 Configuration Register	0x0000_0000
USBD_CFGP19	USBD_BA+0x63C	R/W	Endpoint 19 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG20	USBD_BA+0x640	R/W	Endpoint 20 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD20	USBD_BA+0x644	R/W	Endpoint 20 Maximal Payload Register	0x0000_0000
USBD_CFG20	USBD_BA+0x648	R/W	Endpoint 20 Configuration Register	0x0000_0000
USBD_CFGP20	USBD_BA+0x64C	R/W	Endpoint 20 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG21	USBD_BA+0x650	R/W	Endpoint 21 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD21	USBD_BA+0x654	R/W	Endpoint 21 Maximal Payload Register	0x0000_0000
USBD_CFG21	USBD_BA+0x658	R/W	Endpoint 21 Configuration Register	0x0000_0000
USBD_CFGP21	USBD_BA+0x65C	R/W	Endpoint 21 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG22	USBD_BA+0x660	R/W	Endpoint 22 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD22	USBD_BA+0x664	R/W	Endpoint 22 Maximal Payload Register	0x0000_0000
USBD_CFG22	USBD_BA+0x668	R/W	Endpoint 22 Configuration Register	0x0000_0000
USBD_CFGP22	USBD_BA+0x66C	R/W	Endpoint 22 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG23	USBD_BA+0x670	R/W	Endpoint 23 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD23	USBD_BA+0x674	R/W	Endpoint 23 Maximal Payload Register	0x0000_0000
USBD_CFG23	USBD_BA+0x678	R/W	Endpoint 23 Configuration Register	0x0000_0000
USBD_CFGP23	USBD_BA+0x67C	R/W	Endpoint 23 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG24	USBD_BA+0x680	R/W	Endpoint 24 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD24	USBD_BA+0x684	R/W	Endpoint 24 Maximal Payload Register	0x0000_0000
USBD_CFG24	USBD_BA+0x688	R/W	Endpoint 24 Configuration Register	0x0000_0000
USBD_CFGP24	USBD_BA+0x68C	R/W	Endpoint 24 Set Stall and Clear In/Out Ready Control Register	0x0000_0000

Memory Type	**Address**	**Size**	**Description**
USBD_BA = 0x400C_0000			
USBD_SRAM	USBD_BA+0x800 ~ USBD_BA+0xDFF	1536 Bytes	The SRAM is used for the entire endpoints buffer. Refer to section 6.33.5.7 for the endpoint SRAM structure and its description.

6.33.7 Register Description

USB Interrupt Enable Register (USBD_INTEN)

Register	Offset	R/W	Description	Reset Value
USBD_INTEN	USBD_BA+0x000	R/W	USB Device Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
INNAKEN	**Reserved**						**WKEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**SOFIEN**	**NEVWKIEN**	**VBDETIEN**	**USBIEN**	**BUSIEN**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**INNAKEN**	**Active NAK Function and Its Status in IN Token** 0 = When device responds NAK after receiving IN token, IN NAK status will not be updated to USBD_EPSTS0 and USBD_EPSTS1register, so that the USB interrupt event will not be asserted. 1 = IN NAK status will be updated to USBD_EPSTS0 and USBD_EPSTS1 register and the USB interrupt event will be asserted, when the device responds NAK after receiving IN token.
[14:9]	**Reserved**	Reserved.
[8]	**WKEN**	**Wake-up Function Enable Bit** 0 = USB wake-up function Disabled. 1 = USB wake-up function Enabled.
[7:5]	**Reserved**	Reserved.
[4]	**SOFIEN**	**Start of Frame Interrupt Enable Bit** 0 = SOF Interrupt Disabled. 1 = SOF Interrupt Enabled.
[3]	**NEVWKIEN**	**USB No-event-wake-up Interrupt Enable Bit** 0 = No-event-wake-up Interrupt Disabled. 1 = No-event-wake-up Interrupt Enabled.
[2]	**VBDETIEN**	**V_{BUS} Detection Interrupt Enable Bit** 0 = V_{BUS} detection Interrupt Disabled. 1 = V_{BUS} detection Interrupt Enabled.
[1]	**USBIEN**	**USB Event Interrupt Enable Bit**

		0 = USB event interrupt Disabled. 1 = USB event interrupt Enabled.
[0]	**BUSIEN**	**Bus Event Interrupt Enable Bit** 0 = BUS event interrupt Disabled. 1 = BUS event interrupt Enabled.

USB Interrupt Event Status Register (USBD_INTSTS)

Register	Offset	R/W	Description	Reset Value
USBD_INTSTS	USBD_BA+0x004	R/W	USB Device Interrupt Event Status Register	0x0000_0000

31	30	29	28	27	26	25	24
SETUP	Reserved			EPEVT11	EPEVT10	EPEVT9	EPEVT8
23	22	21	20	19	18	17	16
EPEVT7	EPEVT6	EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			SOFIF	NEVWKIF	VBDETIF	USBIF	BUSIF

Bits	Description	
[31]	**SETUP**	**Setup Event Status** 0 = No Setup event. 1 = Setup event occurred, cleared by writing 1 to USBD_INTSTS[31].
[30:28]	**Reserved**	Reserved.
[27]	**EPEVT11**	**Endpoint 11's USB Event Status** 0 = No event occurred in endpoint 11. 1 = USB event occurred on Endpoint 11. Check USBD_EPSTS1[15:12] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[27] or USBD_EPINTSTS[11] or USBD_INTSTS[1].
[26]	**EPEVT10**	**Endpoint 10's USB Event Status** 0 = No event occurred in endpoint 10. 1 = USB event occurred on Endpoint 10. Check USBD_EPSTS1[11:8] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[26] or USBD_EPINTSTS[10] or USBD_INTSTS[1].
[25]	**EPEVT9**	**Endpoint 9's USB Event Status** 0 = No event occurred in endpoint 9. 1 = USB event occurred on Endpoint 9. Check USBD_EPSTS1[7:4] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[25] or USBD_EPINTSTS[9] or USBD_INTSTS[1].
[24]	**EPEVT8**	**Endpoint 8's USB Event Status** 0 = No event occurred in endpoint 8. 1 = USB event occurred on Endpoint 8. Check USBD_EPSTS1[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[24] or USBD_EPINTSTS[8] or USBD_INTSTS[1].
[23]	**EPEVT7**	**Endpoint 7's USB Event Status** 0 = No event occurred in endpoint 7. 1 = USB event occurred on Endpoint 7. Check USBD_EPSTS0[31:28] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[23] or USBD_EPINTSTS[7] or USBD_INTSTS[1].

[22]	**EPEVT6**	**Endpoint 6's USB Event Status** 0 = No event occurred in endpoint 6. 1 = USB event occurred on Endpoint 6. Check USBD_EPSTS0[27:24] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[22] or USBD_EPINTSTS[6] or USBD_INTSTS[1].
[21]	**EPEVT5**	**Endpoint 5's USB Event Status** 0 = No event occurred in endpoint 5. 1 = USB event occurred on Endpoint 5. Check USBD_EPSTS0[23:20] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[21] or USBD_EPINTSTS[5] or USBD_INTSTS[1].
[20]	**EPEVT4**	**Endpoint 4's USB Event Status** 0 = No event occurred in endpoint 4. 1 = USB event occurred on Endpoint 4. Check USBD_EPSTS0[19:16] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[20] or USBD_EPINTSTS[4] or USBD_INTSTS[1].
[19]	**EPEVT3**	**Endpoint 3's USB Event Status** 0 = No event occurred in endpoint 3. 1 = USB event occurred on Endpoint 3. Check USBD_EPSTS0[15:12] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[19] or USBD_EPINTSTS[3] or USBD_INTSTS[1].
[18]	**EPEVT2**	**Endpoint 2's USB Event Status** 0 = No event occurred in endpoint 2. 1 = USB event occurred on Endpoint 2. Check USBD_EPSTS0[11:8] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[18] or USBD_EPINTSTS[2] or USBD_INTSTS[1].
[17]	**EPEVT1**	**Endpoint 1's USB Event Status** 0 = No event occurred in endpoint 1. 1 = USB event occurred on Endpoint 1. Check USBD_EPSTS0[7:4] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[17] or USBD_EPINTSTS[1] or USBD_INTSTS[1].
[16]	**EPEVT0**	**Endpoint 0's USB Event Status** 0 = No event occurred in endpoint 0. 1 = USB event occurred on Endpoint 0. Check USBD_EPSTS0[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[16] or USBD_EPINTSTS[0] or USBD_INTSTS[1].
[15:5]	**Reserved**	Reserved.
[4]	**SOFIF**	**Start of Frame Interrupt Status** 0 = SOF event did not occur. 1 = SOF event occurred, cleared by writing 1 to USBD_INTSTS[4].
[3]	**NEVWKIF**	**No-event-wake-up Interrupt Status** 0 = NEVWK event did not occur. 1 = No-event-wake-up event occurred, cleared by writing 1 to USBD_INTSTS[3].
[2]	**VBDETIF**	**V_{BUS} Detection Interrupt Status** 0 = There is not attached/detached event in the USB. 1 = There is attached/detached event in the USB bus and it is cleared by writing 1 to USBD_INTSTS[2].
[1]	**USBIF**	**USB Event Interrupt Status** The USB event includes the SETUP Token, IN Token, OUT ACK, ISO IN, or ISO OUT events in the bus.

		0 = No USB event occurred. 1 = USB event occurred. Check EPSTS0~25 in USBD_EPSTS0~3 to know which kind of USB event was occurred, cleared by writing 1 to USBD_INTSTS[1] or USBD_INTSTS[27:16] or USBD_EPINTSTS[24:0] and SETUP (USBD_INTSTS[31]).
[0]	**BUSIF**	**BUS Interrupt Status** The BUS event means that there is one of the suspense or the resume function in the bus. 0 = No BUS event occurred. 1 = Bus event occurred; check USBD_ATTR[3:0] to know which kind of bus event was occurred, cleared by writing 1 to USBD_INTSTS[0].

USB Device Function Address Register (USBD_FADDR)

A 7-bit value is used as the address of a device on the USB BUS.

Register	Offset	R/W	Description	Reset Value
USBD_FADDR	USBD_BA+0x008	R/W	USB Device Function Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**FADDR**						

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:0]	**FADDR**	USB device function address

USB Endpoint Status Register (USBD_EPSTS)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS	USBD_BA+0x00C	R	USB Device Endpoint Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OV	**Reserved**						

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**OV**	**Overrun** It indicates that the received data is over the maximum payload number or not. 0 = No overrun. 1 = Out Data is more than the Max Payload in MXPLD register or the Setup Data is more than 8 bytes.
[6:0]	**Reserved**	Reserved.

USB Bus Status and Attribution Register (USBD_ATTR)

Register	Offset	R/W	Description	Reset Value
USBD_ATTR	USBD_BA+0x010	R/W	USB Device Bus Status and Attribution Register	0x0000_0040

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		L1RESUME	L1SUSPEND	LPMACK	BYTEM	PWRDN	DPPUEN
7	6	5	4	3	2	1	0
USBEN	Reserved	RWAKEUP	PHYEN	TOUT	RESUME	SUSPEND	USBRST

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**L1RESUME**	**LPM L1 Resume (Read Only)** 0 = Bus no LPM L1 state resume. 1 = LPM L1 state resume from LPM L1 state suspend.
[12]	**L1SUSPEND**	**LPM L1 Suspend (Read Only)** 0 = Bus no L1 state suspend. 1 = This bit is set by the hardware when LPM command to enter the L1 state is successfully received and acknowledged.
[11]	**LPMACK**	**LPM Token Acknowledge Enable Bit** The NYET/ACK will be returned only on a successful LPM transaction if no errors in both the EXT token and the LPM token and a valid bLinkState = 0001 (L1) is received, else ERROR and STALL will be returned automatically, respectively. 0= The valid LPM Token will be NYET. 1= The valid LPM Token will be ACK.
[10]	**BYTEM**	**CPU Access USB SRAM Size Mode Selection** 0 = Word mode: The size of the transfer from CPU to USB SRAM can be Word only. 1 = Byte mode: The size of the transfer from CPU to USB SRAM can be Byte only.
[9]	**PWRDN**	**Power-down PHY Transceiver, Low Active** 0 = Power-down related circuit of PHY transceiver. 1 = Turn-on related circuit of PHY transceiver.
[8]	**DPPUEN**	**Pull-up Resistor on USB_DP Enable Bit** 0 = Pull-up resistor in USB_D+ bus Disabled. 1 = Pull-up resistor in USB_D+ bus Active.
[7]	**USBEN**	**USB Controller Enable Bit** 0 = USB Controller Disabled. 1 = USB Controller Enabled.
[6]	**Reserved**	Reserved.

[5]	**RWAKEUP**	**Remote Wake-up** 0 = Release the USB bus from K state. 1 = Force USB bus to K (USB_D+ low, USB_D- high) state, used for remote wake-up.
[4]	**PHYEN**	**PHY Transceiver Function Enable Bit** 0 = PHY transceiver function Disabled. 1 = PHY transceiver function Enabled.
[3]	**TOUT**	**Time-out Status (Read Only)** 0 = No time-out. 1 = No Bus response more than 18 bits time.
[2]	**RESUME**	**Resume Status (Read Only)** 0 = No bus resume. 1 = Resume from suspend.
[1]	**SUSPEND**	**Suspend Status (Read Only)** 0 = Bus no suspend. 1 = Bus idle more than 3ms, either cable is plugged-out or host is sleeping.
[0]	**USBRST**	**USB Reset Status (Read Only)** 0 = Bus no reset. 1 = Bus reset when SE0 (single-ended 0) more than 2.5us.

USB Device V_{BUS} Detection Register (USBD_VBUSDET)

Register	Offset	R/W	Description	Reset Value
USBD_VBUSDET	USBD_BA+0x014	R	USB Device V_{BUS} Detection Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**VBUSDET**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**VBUSDET**	**Device V_{BUS} Detection** 0 = Controller is not attached to the USB host. 1 = Controller is attached to the USB host.

USB SETUP Token Buffer Segmentation Register (USBD_STBUFSEG)

Register	Offset	R/W	Description	Reset Value
USBD_STBUFSEG	USBD_BA+0x018	R/W	SETUP Token Buffer Segmentation Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**STBUFSEG**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STBUFSEG					**Reserved**		

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:3]	**STBUFSEG**	**SETUP Token Buffer Segmentation** It is used to indicate the offset address for the SETUP token with the USB Device SRAM starting address. The effective starting address is USBD_SRAM address + {STBUFSE.G. 3'b000} Where the USBD_SRAM address = USBD_BA+0x800h. **Note:** It is used for SETUP token only.
[2:0]	**Reserved**	Reserved.

USB Endpoint Status Register 0 (USBD_EPSTS0)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS0	USBD_BA+0x020	R	USB Device Endpoint Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
EPSTS7				EPSTS6			
23	22	21	20	19	18	17	16
EPSTS5				EPSTS4			
15	14	13	12	11	10	9	8
EPSTS3				EPSTS2			
7	6	5	4	3	2	1	0
EPSTS1				EPSTS0			

Bits	Description	
[31:28]	**EPSTS7**	**Endpoint 7 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[27:24]	**EPSTS6**	**Endpoint 6 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[23:20]	**EPSTS5**	**Endpoint 5 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

[19:16]	**EPSTS4**	**Endpoint 4 Status** **These Bits Are Used To Indicate The Current Status Of This Endpoint.** **0000 = In ACK.** **0001 = In NAK.** **0010 = Out Packet Data0 ACK.** **0110 = Out Packet Data1 ACK.** **0111 = Isochronous Transfer End.**
[15:12]	EPSTS3	Endpoint 3 Status These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[11:8]	EPSTS2	Endpoint 2 Status These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[7:4]	EPSTS1	Endpoint 1 Status These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[3:0]	EPSTS0	Endpoint 0 Status These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

USB Endpoint Status Register 1 (USBD_EPSTS1)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS1	USBD_BA+0x024	R	USB Device Endpoint Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
EPSTS15				EPSTS14			
23	22	21	20	19	18	17	16
EPSTS13				EPSTS12			
15	14	13	12	11	10	9	8
EPSTS11				EPSTS10			
7	6	5	4	3	2	1	0
EPSTS9				EPSTS8			

Bits	Description	
[31:28]	**EPSTS15**	**Endpoint 15 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[27:24]	**EPSTS14**	**Endpoint 14 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[23:20]	**EPSTS13**	**Endpoint 13 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[19:16]	**EPSTS12**	**Endpoint 12 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

[15:12]	**EPSTS11**	**Endpoint 11 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[11:8]	**EPSTS10**	**Endpoint 10 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[7:4]	**EPSTS9**	**Endpoint 9 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[3:0]	**EPSTS8**	**Endpoint 8 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

USB Endpoint Status Register 2 (USBD_EPSTS2)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS2	USBD_BA+0x028	R	USB Device Endpoint Status Register 2	0x0000_0000

31	30	29	28	27	26	25	24
EPSTS23				EPSTS22			
23	22	21	20	19	18	17	16
EPSTS21				EPSTS20			
15	14	13	12	11	10	9	8
EPSTS19				EPSTS18			
7	6	5	4	3	2	1	0
EPSTS17				EPSTS16			

Bits	Description	
[31:28]	**EPSTS23**	**Endpoint 23 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[27:24]	**EPSTS22**	**Endpoint 22 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[23:20]	**EPSTS21**	**Endpoint 21 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[19:16]	**EPSTS20**	**Endpoint 20 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

[15:12]	**EPSTS19**	**Endpoint 19 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[11:8]	**EPSTS18**	**Endpoint 18 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[7:4]	**EPSTS17**	**Endpoint 17 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.
[3:0]	**EPSTS16**	**Endpoint 16 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

USB Endpoint Status Register 3 (USBD_EPSTS3)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS3	USBD_BA+0x02C	R	USB Device Endpoint Status Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**EPSTS24**			

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**EPSTS24**	**Endpoint 24 Status** These bits are used to indicate the current status of this endpoint. 0000 = In ACK. 0001 = In NAK. 0010 = Out Packet Data0 ACK. 0110 = Out Packet Data1 ACK. 0111 = Isochronous transfer end.

USB Endpoint Interrupt Event Status Register (USBD_EPINTSTS)

Register	Offset	R/W	Description	Reset Value
USBD_EPINTSTS	USBD_BA+0x030	R/W	USB Device Endpoint Interrupt Event Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							EPEVT24
23	22	21	20	19	18	17	16
EPEVT23	EPEVT22	EPEVT21	EPEVT20	EPEVT19	EPEVT18	EPEVT17	EPEVT16
15	14	13	12	11	10	9	8
EPEVT15	EPEVT14	EPEVT13	EPEVT12	EPEVT11	EPEVT10	EPEVT9	EPEVT8
7	6	5	4	3	2	1	0
EPEVT7	EPEVT6	EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**EPEVT24**	**Endpoint 24's USB Event Status** 0 = No event occurred in endpoint 24. 1 = USB event occurred on Endpoint 24. Check USBD_EPSTS3[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[24] or USBD_INTSTS[1].
[23]	**EPEVT23**	**Endpoint 23's USB Event Status** 0 = No event occurred in endpoint 23. 1 = USB event occurred on Endpoint 23. Check USBD_EPSTS2[31:28] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[23] or USBD_INTSTS[1].
[22]	**EPEVT22**	**Endpoint 22's USB Event Status** 0 = No event occurred in endpoint 22. 1 = USB event occurred on Endpoint 22. Check USBD_EPSTS2[27:24] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[22] or USBD_INTSTS[1].
[21]	**EPEVT21**	**Endpoint 21's USB Event Status** 0 = No event occurred in endpoint 21. 1 = USB event occurred on Endpoint 21. Check USBD_EPSTS2[23:20] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[21] or USBD_INTSTS[1].
[20]	**EPEVT20**	**Endpoint 20's USB Event Status** 0 = No event occurred in endpoint 20. 1 = USB event occurred on Endpoint 20. Check USBD_EPSTS2[19:16] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[20] or USBD_INTSTS[1].
[19]	**EPEVT19**	**Endpoint 19's USB Event Status** 0 = No event occurred in endpoint 19. 1 = USB event occurred on Endpoint 19. Check USBD_EPSTS2[15:12] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[19] or

		USBD_INTSTS[1].
[18]	**EPEVT18**	**Endpoint 18's USB Event Status** 0 = No event occurred in endpoint 18. 1 = USB event occurred on Endpoint 18. Check USBD_EPSTS2[11:8] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[18] or USBD_INTSTS[1].
[17]	**EPEVT17**	**Endpoint 17's USB Event Status** 0 = No event occurred in endpoint 17. 1 = USB event occurred on Endpoint 17. Check USBD_EPSTS2[7:4] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[17] or USBD_INTSTS[1].
[16]	**EPEVT16**	**Endpoint 16's USB Event Status** 0 = No event occurred in endpoint 16. 1 = USB event occurred on Endpoint 16. Check USBD_EPSTS2[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[16] or USBD_INTSTS[1].
[15]	**EPEVT15**	**Endpoint 15's USB Event Status** 0 = No event occurred in endpoint 15. 1 = USB event occurred on Endpoint 15. Check USBD_EPSTS1[27:24] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[15] or USBD_INTSTS[1].
[14]	**EPEVT14**	**Endpoint 14's USB Event Status** 0 = No event occurred in endpoint 14. 1 = USB event occurred on Endpoint 14. Check USBD_EPSTS1[23:20] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[14] or USBD_INTSTS[1].
[13]	**EPEVT13**	**Endpoint 13's USB Event Status** 0 = No event occurred in endpoint 13. 1 = USB event occurred on Endpoint 13. Check USBD_EPSTS1[19:16] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[13] or USBD_INTSTS[1].
[12]	**EPEVT12**	**Endpoint 12's USB Event Status** 0 = No event occurred in endpoint 12. 1 = USB event occurred on Endpoint 12. Check USBD_EPSTS1[15:12] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[12] or USBD_INTSTS[1].
[11]	**EPEVT11**	**Endpoint 11's USB Event Status** 0 = No event occurred in endpoint 11. 1 = USB event occurred on Endpoint 11. Check USBD_EPSTS1[15:12] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[11] or USBD_INTSTS[27] or USBD_INTSTS[1].
[10]	**EPEVT10**	**Endpoint 10's USB Event Status** 0 = No event occurred in endpoint 10. 1 = USB event occurred on Endpoint 10. Check USBD_EPSTS1[11:8] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[10] or USBD_INTSTS[26] or USBD_INTSTS[1].
[9]	**EPEVT9**	**Endpoint 9's USB Event Status** 0 = No event occurred in endpoint 9. 1 = USB event occurred on Endpoint 9. Check USBD_EPSTS1[7:4] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[9] or USBD_INTSTS[25] or USBD_INTSTS[1].

[8]	**EPEVT8**	**Endpoint 8's USB Event Status** 0 = No event occurred in endpoint 8. 1 = USB event occurred on Endpoint 8. Check USBD_EPSTS1[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[8] or USBD_INTSTS[24] or USBD_INTSTS[1].
[7]	**EPEVT7**	**Endpoint 7's USB Event Status** 0 = No event occurred in endpoint 7. 1 = USB event occurred on Endpoint 7. Check USBD_EPSTS0[31:28] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[7] or USBD_INTSTS[23] or USBD_INTSTS[1].
[6]	**EPEVT6**	**Endpoint 6's USB Event Status** 0 = No event occurred in endpoint 6. 1 = USB event occurred on Endpoint 6. Check USBD_EPSTS0[27:24] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[6] or USBD_INTSTS[22] or USBD_INTSTS[1].
[5]	**EPEVT5**	**Endpoint 5's USB Event Status** 0 = No event occurred in endpoint 5. 1 = USB event occurred on Endpoint 5. Check USBD_EPSTS0[23:20] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[5] or USBD_INTSTS[21] or USBD_INTSTS[1].
[4]	**EPEVT4**	**Endpoint 4's USB Event Status** 0 = No event occurred in endpoint 4. 1 = USB event occurred on Endpoint 4. Check USBD_EPSTS0[19:16] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[4] or USBD_INTSTS[20] or USBD_INTSTS[1].
[3]	**EPEVT3**	**Endpoint 3's USB Event Status** 0 = No event occurred in endpoint 3. 1 = USB event occurred on Endpoint 3. Check USBD_EPSTS0[15:12] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[3] or USBD_INTSTS[19] or USBD_INTSTS[1].
[2]	**EPEVT2**	**Endpoint 2's USB Event Status** 0 = No event occurred in endpoint 2. 1 = USB event occurred on Endpoint 2. Check USBD_EPSTS0[11:8] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[2] or USBD_EPINTSTS[2] or USBD_INTSTS[18] or USBD_INTSTS[1].
[1]	**EPEVT1**	**Endpoint 1's USB Event Status** 0 = No event occurred in endpoint 1. 1 = USB event occurred on Endpoint 1. Check USBD_EPSTS0[7:4] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[1] or USBD_INTSTS[17] or USBD_INTSTS[1].
[0]	**EPEVT0**	**Endpoint 0's USB Event Status** 0 = No event occurred in endpoint 0. 1 = USB event occurred on Endpoint 0. Check USBD_EPSTS0[3:0] to know which kind of USB event was occurred, cleared by writing 1 to USBD_EPINTSTS[0] or USBD_INTSTS[16] or USBD_INTSTS[1].

USB LPM Attribution Register (USBD_LPMATTR)

Register	Offset	R/W	Description	Reset Value
USBD_LPMATTR	USBD_BA+0x088	R	USB LPM Attribution Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**LPMRWAKUP**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LPMBESL				**LPMLINKSTS**			

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**LPMRWAKUP**	**LPM Remote Wakeup** This bit contains the bRemoteWake value received with last ACK LPM Token
[7:4]	**LPMBESL**	**LPM Best Effort Service Latency** These bits contain the BESL value received with last ACK LPM Token
[3:0]	**LPMLINKSTS**	**LPM Link State** These bits contain the bLinkState received with last ACK LPM Token

USB Frame Number Register (USBD_FN)

Register	Offset	R/W	Description	Reset Value
USBD_FN	USBD_BA+0x08C	R	USB Frame Number Register	0x0000_0XXX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**FN**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FN							

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:0]	**FN**	**Frame Number** These bits contain the 11 bits frame number in the last received SOF packet.

USB Drive SE0 Register (USBD_SE0)

Register	Offset	R/W	Description	Reset Value
USBD_SE0	USBD_BA+0x090	R/W	USB Device Drive SE0 Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**SE0**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SE0**	**Drive Single Ended Zero in USB Bus** The Single Ended Zero (SE0) is when both lines (USB_D+ and USB_D-) are being pulled low. 0 = Normal operation. 1 = Force USB PHY transceiver to drive SE0.

USB Buffer Segmentation Register (USBD_BUFSEG0~24)

Register	Offset	R/W	Description	Reset Value
USBD_BUFSEG0	USBD_BA+0x500	R/W	Endpoint 0 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG1	USBD_BA+0x510	R/W	Endpoint 1 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG2	USBD_BA+0x520	R/W	Endpoint 2 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG3	USBD_BA+0x530	R/W	Endpoint 3 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG4	USBD_BA+0x540	R/W	Endpoint 4 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG5	USBD_BA+0x550	R/W	Endpoint 5 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG6	USBD_BA+0x560	R/W	Endpoint 6 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG7	USBD_BA+0x570	R/W	Endpoint 7 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG8	USBD_BA+0x580	R/W	Endpoint 8 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG9	USBD_BA+0x590	R/W	Endpoint 9 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG10	USBD_BA+0x5A0	R/W	Endpoint 10 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG11	USBD_BA+0x5B0	R/W	Endpoint 11 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG12	USBD_BA+0x5C0	R/W	Endpoint 12 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG13	USBD_BA+0x5D0	R/W	Endpoint 13 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG14	USBD_BA+0x5E0	R/W	Endpoint 14 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG15	USBD_BA+0x5F0	R/W	Endpoint 15 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG16	USBD_BA+0x600	R/W	Endpoint 16 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG17	USBD_BA+0x610	R/W	Endpoint 17 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG18	USBD_BA+0x620	R/W	Endpoint 18 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG19	USBD_BA+0x630	R/W	Endpoint 19 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG20	USBD_BA+0x640	R/W	Endpoint 20 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG21	USBD_BA+0x650	R/W	Endpoint 21 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG22	USBD_BA+0x660	R/W	Endpoint 22 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG23	USBD_BA+0x670	R/W	Endpoint 23 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG24	USBD_BA+0x680	R/W	Endpoint 24 Buffer Segmentation Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							

15	14	13	12	11	10	9	8
Reserved					BUFSEG		
7	6	5	4	3	2	1	0
BUFSEG					Reserved		

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:3]	**BUFSEG**	**Endpoint Buffer Segmentation** It is used to indicate the offset address for each endpoint with the USB SRAM starting address The effective starting address of the endpoint is USBD_SRAM address + {BUFSE.G. 3'b000} Where the USBD_SRAM address = USBD_BA+0x800h. Refer to the section 6.33.5.7 for the endpoint SRAM structure and its description.
[2:0]	**Reserved**	Reserved.

USB Maximal Payload Register (USBD_MXPLD0~24)

Register	Offset	R/W	Description	Reset Value
USBD_MXPLD0	USBD_BA+0x504	R/W	Endpoint 0 Maximal Payload Register	0x0000_0000
USBD_MXPLD1	USBD_BA+0x514	R/W	Endpoint 1 Maximal Payload Register	0x0000_0000
USBD_MXPLD2	USBD_BA+0x524	R/W	Endpoint 2 Maximal Payload Register	0x0000_0000
USBD_MXPLD3	USBD_BA+0x534	R/W	Endpoint 3 Maximal Payload Register	0x0000_0000
USBD_MXPLD4	USBD_BA+0x544	R/W	Endpoint 4 Maximal Payload Register	0x0000_0000
USBD_MXPLD5	USBD_BA+0x554	R/W	Endpoint 5 Maximal Payload Register	0x0000_0000
USBD_MXPLD6	USBD_BA+0x564	R/W	Endpoint 6 Maximal Payload Register	0x0000_0000
USBD_MXPLD7	USBD_BA+0x574	R/W	Endpoint 7 Maximal Payload Register	0x0000_0000
USBD_MXPLD8	USBD_BA+0x584	R/W	Endpoint 8 Maximal Payload Register	0x0000_0000
USBD_MXPLD9	USBD_BA+0x594	R/W	Endpoint 9 Maximal Payload Register	0x0000_0000
USBD_MXPLD10	USBD_BA+0x5A4	R/W	Endpoint 10 Maximal Payload Register	0x0000_0000
USBD_MXPLD11	USBD_BA+0x5B4	R/W	Endpoint 11 Maximal Payload Register	0x0000_0000
USBD_MXPLD12	USBD_BA+0x5C4	R/W	Endpoint 12 Maximal Payload Register	0x0000_0000
USBD_MXPLD13	USBD_BA+0x5D4	R/W	Endpoint 13 Maximal Payload Register	0x0000_0000
USBD_MXPLD14	USBD_BA+0x5E4	R/W	Endpoint 14 Maximal Payload Register	0x0000_0000
USBD_MXPLD15	USBD_BA+0x5F4	R/W	Endpoint 15 Maximal Payload Register	0x0000_0000
USBD_MXPLD16	USBD_BA+0x604	R/W	Endpoint 16 Maximal Payload Register	0x0000_0000
USBD_MXPLD17	USBD_BA+0x614	R/W	Endpoint 17 Maximal Payload Register	0x0000_0000
USBD_MXPLD18	USBD_BA+0x624	R/W	Endpoint 18 Maximal Payload Register	0x0000_0000
USBD_MXPLD19	USBD_BA+0x634	R/W	Endpoint 19 Maximal Payload Register	0x0000_0000
USBD_MXPLD20	USBD_BA+0x644	R/W	Endpoint 20 Maximal Payload Register	0x0000_0000
USBD_MXPLD21	USBD_BA+0x654	R/W	Endpoint 21 Maximal Payload Register	0x0000_0000
USBD_MXPLD22	USBD_BA+0x664	R/W	Endpoint 22 Maximal Payload Register	0x0000_0000
USBD_MXPLD23	USBD_BA+0x674	R/W	Endpoint 23 Maximal Payload Register	0x0000_0000
USBD_MXPLD24	USBD_BA+0x684	R/W	Endpoint 24 Maximal Payload Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							

15	14	13	12	11	10	9	8
Reserved					MXPLD		
7	6	5	4	3	2	1	0
MXPLD							

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:0]	**MXPLD**	**Maximal Payload** Define the data length which is transmitted to host (IN token) or the actual data length which is received from the host (OUT token). It also used to indicate that the endpoint is ready to be transmitted in IN token or received in OUT token. (1) When the register is written by CPU, For IN token, the value of MXPLD is used to define the data length to be transmitted and indicate the data buffer is ready. For OUT token, it means that the controller is ready to receive data from the host and the value of MXPLD is the maximal data length comes from host. (2) When the register is read by CPU, For IN token, the value of MXPLD is indicated by the data length be transmitted to host For OUT token, the value of MXPLD is indicated the actual data length receiving from host. **Note:** Once MXPLD is written, the data packets will be transmitted/received immediately after IN/OUT token arrived.

USB Configuration Register (USBD_CFG0~24)

Register	Offset	R/W	Description	Reset Value
USBD_CFG0	USBD_BA+0x508	R/W	Endpoint 0 Configuration Register	0x0000_0000
USBD_CFG1	USBD_BA+0x518	R/W	Endpoint 1 Configuration Register	0x0000_0000
USBD_CFG2	USBD_BA+0x528	R/W	Endpoint 2 Configuration Register	0x0000_0000
USBD_CFG3	USBD_BA+0x538	R/W	Endpoint 3 Configuration Register	0x0000_0000
USBD_CFG4	USBD_BA+0x548	R/W	Endpoint 4 Configuration Register	0x0000_0000
USBD_CFG5	USBD_BA+0x558	R/W	Endpoint 5 Configuration Register	0x0000_0000
USBD_CFG6	USBD_BA+0x568	R/W	Endpoint 6 Configuration Register	0x0000_0000
USBD_CFG7	USBD_BA+0x578	R/W	Endpoint 7 Configuration Register	0x0000_0000
USBD_CFG8	USBD_BA+0x588	R/W	Endpoint 8 Configuration Register	0x0000_0000
USBD_CFG9	USBD_BA+0x598	R/W	Endpoint 9 Configuration Register	0x0000_0000
USBD_CFG10	USBD_BA+0x5A8	R/W	Endpoint 10 Configuration Register	0x0000_0000
USBD_CFG11	USBD_BA+0x5B8	R/W	Endpoint 11 Configuration Register	0x0000_0000
USBD_CFG12	USBD_BA+0x5C8	R/W	Endpoint 12 Configuration Register	0x0000_0000
USBD_CFG13	USBD_BA+0x5D8	R/W	Endpoint 13 Configuration Register	0x0000_0000
USBD_CFG14	USBD_BA+0x5E8	R/W	Endpoint 14 Configuration Register	0x0000_0000
USBD_CFG15	USBD_BA+0x5F8	R/W	Endpoint 15 Configuration Register	0x0000_0000
USBD_CFG16	USBD_BA+0x608	R/W	Endpoint 16 Configuration Register	0x0000_0000
USBD_CFG17	USBD_BA+0x618	R/W	Endpoint 17 Configuration Register	0x0000_0000
USBD_CFG18	USBD_BA+0x628	R/W	Endpoint 18 Configuration Register	0x0000_0000
USBD_CFG19	USBD_BA+0x638	R/W	Endpoint 19 Configuration Register	0x0000_0000
USBD_CFG20	USBD_BA+0x648	R/W	Endpoint 20 Configuration Register	0x0000_0000
USBD_CFG21	USBD_BA+0x658	R/W	Endpoint 21 Configuration Register	0x0000_0000
USBD_CFG22	USBD_BA+0x668	R/W	Endpoint 22 Configuration Register	0x0000_0000
USBD_CFG23	USBD_BA+0x678	R/W	Endpoint 23 Configuration Register	0x0000_0000
USBD_CFG24	USBD_BA+0x688	R/W	Endpoint 24 Configuration Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							

15	14	13	12	11	10	9	8
Reserved				DBEN	DBTGACTIVE	CSTALL	Reserved
7	6	5	4	3	2	1	0
DSQSYNC	STATE		ISOCH	EPNUM			

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11]	**DBEN**	**Double Buffer Enable** 0 = Single buffer mode. 1 = Double buffer mode.
[10]	**DBTGACTIVE**	**Double Buffer Toggle Active Bit** 0 = Inactive in double buffer mode. 1 = Active in double buffer mode. When DBEN = 1 and endpoint successful transfer, DBTGACTIVE of each double buffer will be toggled automatically by hardware.
[9]	**CSTALL**	**Clear STALL Response** 0 = Disable the device to clear the STALL handshake in setup stage. 1 = Clear the device to response STALL handshake in setup stage.
[8]	**Reserved**	Reserved.
[7]	**DSQSYNC**	**Data Sequence Synchronization** 0 = DATA0 PID. 1 = DATA1 PID. For IN token, DSQSYNC specify DATA0 or DATA1 PID in transfer data packet For OUT token, DSQSYNC specify DATA0 or DATA1 PID in received data packet. DSQSYNC will be toggled automatically by hardware when IN or OUT token transfer successfully in single buffer mode, but won't be toggled in double buffer mode. **Note 1:** When double buffer is enabled, hardware will automatically write 0 to DSQSYNC with active double buffer and write 1 to DSQSYNC with inactive double buffer. **Note 2:** It won't be toggled by hardware when DBEN = 1. USB data toggle will be guaranteed by changing endpoint.
[6:5]	**STATE**	**Endpoint State** 00 = Endpoint is Disabled. 01 = Out endpoint. 10 = IN endpoint. 11 = Undefined.
[4]	**ISOCH**	**Isochronous Endpoint** This bit is used to set the endpoint as Isochronous endpoint, no handshake. 0 = No Isochronous endpoint. 1 = Isochronous endpoint.
[3:0]	**EPNUM**	**Endpoint Number** These bits are used to define the endpoint number of the current endpoint.

USB Extra Configuration Register (USBD_CFGP0~24)

Register	Offset	R/W	Description	Reset Value
USBD_CFGP0	USBD_BA+0x50C	R/W	Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP1	USBD_BA+0x51C	R/W	Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP2	USBD_BA+0x52C	R/W	Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP3	USBD_BA+0x53C	R/W	Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP4	USBD_BA+0x54C	R/W	Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP5	USBD_BA+0x55C	R/W	Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP6	USBD_BA+0x56C	R/W	Endpoint 6 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP7	USBD_BA+0x57C	R/W	Endpoint 7 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP8	USBD_BA+0x58C	R/W	Endpoint 8 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP9	USBD_BA+0x59C	R/W	Endpoint 9 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP10	USBD_BA+0x5AC	R/W	Endpoint 10 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP11	USBD_BA+0x5BC	R/W	Endpoint 11 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP12	USBD_BA+0x5CC	R/W	Endpoint 12 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP13	USBD_BA+0x5DC	R/W	Endpoint 13 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP14	USBD_BA+0x5EC	R/W	Endpoint 14 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP15	USBD_BA+0x5FC	R/W	Endpoint 15 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP16	USBD_BA+0x60C	R/W	Endpoint 16 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP17	USBD_BA+0x61C	R/W	Endpoint 17 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP18	USBD_BA+0x62C	R/W	Endpoint 18 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP19	USBD_BA+0x63C	R/W	Endpoint 19 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP20	USBD_BA+0x64C	R/W	Endpoint 20 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP21	USBD_BA+0x65C	R/W	Endpoint 21 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP22	USBD_BA+0x66C	R/W	Endpoint 22 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP23	USBD_BA+0x67C	R/W	Endpoint 23 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP24	USBD_BA+0x68C	R/W	Endpoint 24 Set Stall and Clear In/Out Ready Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							

15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						SSTALL	CLRRDY

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**SSTALL**	**Set STALL** 0 = Disable the device to response STALL. 1 = Set the device to respond STALL automatically.
[0]	**CLRRDY**	**Clear Ready** When the USBD_MXPLD0~24 register is set by user, it means that the endpoint is ready to transmit or receive data. If the user wants to disable this transaction before the transaction start, users can set this bit to 1 to disable it and it is automatically cleared to 0. For IN token, write '1' to clear the IN token had ready to transmit the data to USB. For OUT token, write '1' to clear the OUT token had ready to receive the data from USB. This bit is written 1 only and is always 0 when it is read back.

6.34 High Speed USB 2.0 Device Controller (HSUSBD)

6.34.1 Overview

The USB device controller interfaces the AHB bus and the UTMI bus. The USB controller contains both the AHB master interface and AHB slave interface. CPU programs the USB controller registers through the AHB slave interface. For IN or OUT transfer, the USB device controller needs to write data to memory or read data from memory through the AHB master interface. The USB device controller is compliant with USB 2.0 specification and it contains 12 configurable endpoints in addition to control endpoint. These endpoints could be configured to BULK, INTERRUPT or ISO. The USB device controller has a built-in DMA to relieve the load of CPU.

6.34.2 Features

- USB Specification revision 2.0 compliant
- Supports 12 configurable endpoints in addition to Control Endpoint
- Each of the endpoints can be Isochronous, Bulk or Interrupt and either IN or OUT direction
- Three different operation modes of an in-endpoint – Auto Validation mode, Manual Validation mode, Fly mode
- Supports DMA operation
- 4096 Bytes Configurable RAM used as endpoint buffer
- Supports Endpoint Maximum Packet Size up to 1024 bytes
- Supports V_{BUS} / Resume wakeup from system power-down mode
- Supports Link Power Management (LPM) feature
- Supports Battery Charging 1.2 (BC1.2) feature

6.34.3 Block Diagram

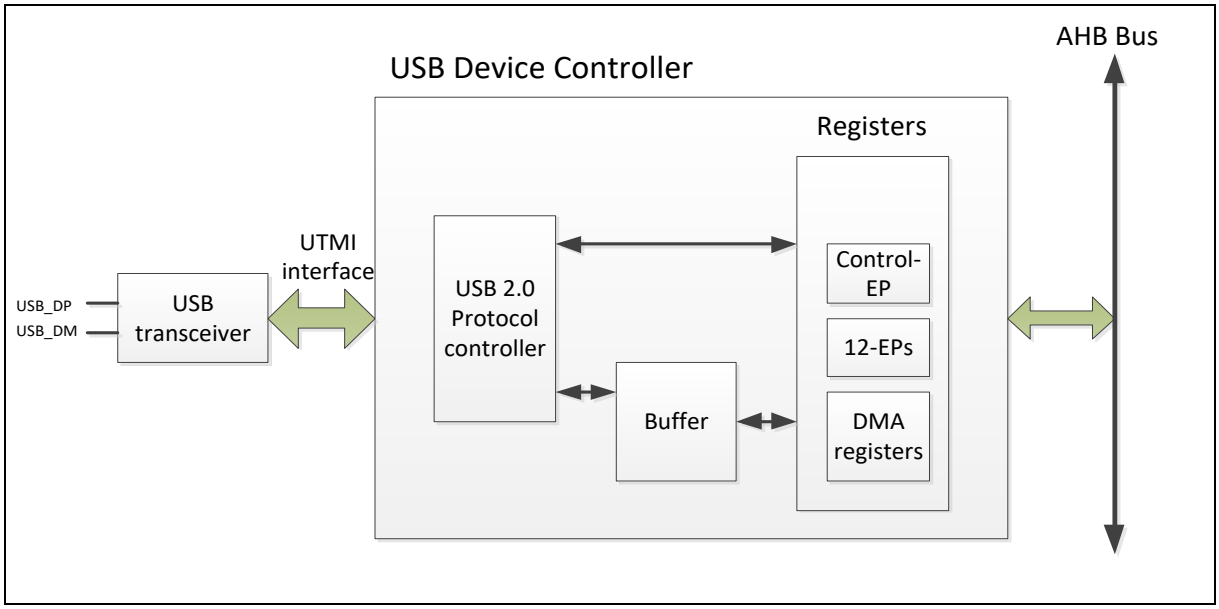


Figure 6.34-1 USB Device Controller Block Diagram

6.34.4 Basic Configuration

- Clock Source Configuration

- Enable HSUSBD clock in HSUSBDCKEN (CLK_AHBCLK0[10]).

- Reset Configuration
  - Reset HSUSBD controller in HSUSBDRST (SYS_IPRST0[10]).

6.34.5 Functional Description

6.34.5.1 Operation of Different In-transfer Modes

The data for any in-transfer is written into the internal buffer when in turn is sent to the host on receipt of an in-token. There are three different modes by which the data sent to the host is validated by CPU.

- Auto-Validation Mode
- Manual-Validation Mode
- Fly Mode

6.34.5.2 Auto-Validation Mode

If an endpoint is selected to be operating in auto-validation mode, the endpoint responds only with data payload to be equal to EPMPS (HSUSBD_EPxMPS[10:0], x = A~L) register. The endpoint controller wait until the amount of data is equal to EPMPS value and then validates the data. If CPU needs to send a short-packet at the end of a transfer, the SHORTTXEN bit (HSUSBD_EPxRSPCTL[6], x = A~L) should be set. When this bit set, any remaining data in the buffer is validated and is sent to the host, for the forthcoming in-token.

This mode requires least intervention of CPU, as most of the work is done by the USB device controller. The mode can be selected, when the data payload sent to host is always equal to EPMPS.

SHORTTXEN	Data Availability In Buffer	Data Sent/NAK Sent
0	< Max. Packet Size	NAK sent
0	>= Max. Packet Size	Data payload of max. packet size sent
1	< Max. Packet Size	Available data of < max. packet size sent
1	>= Max. Packet Size	Data payload of max. packet size sent

6.34.5.3 Manual-Validation Mode

If the endpoint is selected to be operating in manual-validation mode, the endpoint responds only when the data in the buffer is validated by CPU every time. The CPU has to write data into the buffer and then write the count of the data into TXCNT (HSUSBD_EPxTXCNT, x = A~L) register. Once the data is validated by writing a count into the TXCNT register, the data is sent to the host on receipt of an in-token.

This mode requires intervention of CPU for each transfer. But this would be useful if the data-count to be sent each time is not fixed, and it is being decided by CPU.

TXCNT Written	Data Availability In Buffer	Data Sent/NAK Sent
NO	-	NAK sent
YES	TXCNT	Data payload of TXCNT sent

6.34.5.4 Fly Mode

The fly mode of operation is simplest mode of operation, where there is no validation procedure. The buffer is being filled by CPU. If an in-token is send from the host, the data in the buffer is automatically validated and sent to the host. If the data in the buffer spans more than one packet of maximum packet size, the controller automatically packs to equal to EPMPS and sends it to the host.

This mode requires the least intervention by CPU. This mode is best suited for isochronous data transfer, where the speed of data transfer is more important than the packet size.

Data Availability In Buffer	Data Sent
< Max. Packet Size	Data available sent
>= Max. Packet Size	Data payload of max. packet size sent

6.34.5.5 Scatter-Gather Function

User enables DMA scatter gather function by setting SGEN (HSUSBD_DMACTL[6]) = 1, and sets DMACNT (HSUSBD_DMACNT[19:0] as 8 bytes, and the memory address and length will be recorded in the descriptor. The format is shown in Table 6.34-1.

	Format			
	[31]	[30]	[29:0]	
Word0	MEM_ADDR[31:0]			
Word1	EOT	RD	Reserved	Count[19:0]

Table 6.34-1 Scatter Gather Descriptor Format

MEM_ADDR: It specifies the memory address (AHB address).

EOT: End of transfer. When this bit is set to high, it means this is the last descriptor.

RD: “1” means read from memory into buffer. “0” means read from buffer into memory.

6.34.5.6 V_{BUS} / Resume Wakeup Function

The HSUSBD supports V_{BUS} and resume wakeup if system enters Power-down mode, with two separate enable bits VBUSWKEN (HSUSBD_PHYCTL[24]) and LINESTATEWKEN (HSUSBD_PHYCTL[25]).

6.34.5.7 Link Power Management Feature

The HSUSBD supports USB LPM (Link Power Management) feature for shorter transitional latency between the power states.

The LPMEN (HSUSBD_LPMCSR[12]) bit can disable/enable the feature.

6.34.5.8 Battery Charging Detector

Battery Charging (BC) Revision 1.2 is supported. BC1.2 detects attached port in which class. Each class can charge different current. User can detect class of attached port, then charge adequate current to battery to reduce charging time.

After battery charge is detected, user can distinguish SDP (Standard Downstream Port), DCP (Dedicated Charging Port), and CDP (Charging Downstream Port). SDP is standard downstream port in USB2.0 definition that can draw current up to 0.5A after configured. DCP is a charging port in USB that cannot enumerate and can draw current range to 0.5A~5A. CDP can enumerate and draw current range to 1.5A~5A.

User can refer to Figure 6.34-2 to control HSUSBD_BCDC, or refer to BC1.2 demo code. The flow includes V_{BUS} detect, DCD, PD, and SD. First check V_{BUS} attach status, check data pin contact status, and then distinguish SDP, CDP, and DCP by PD and SD.

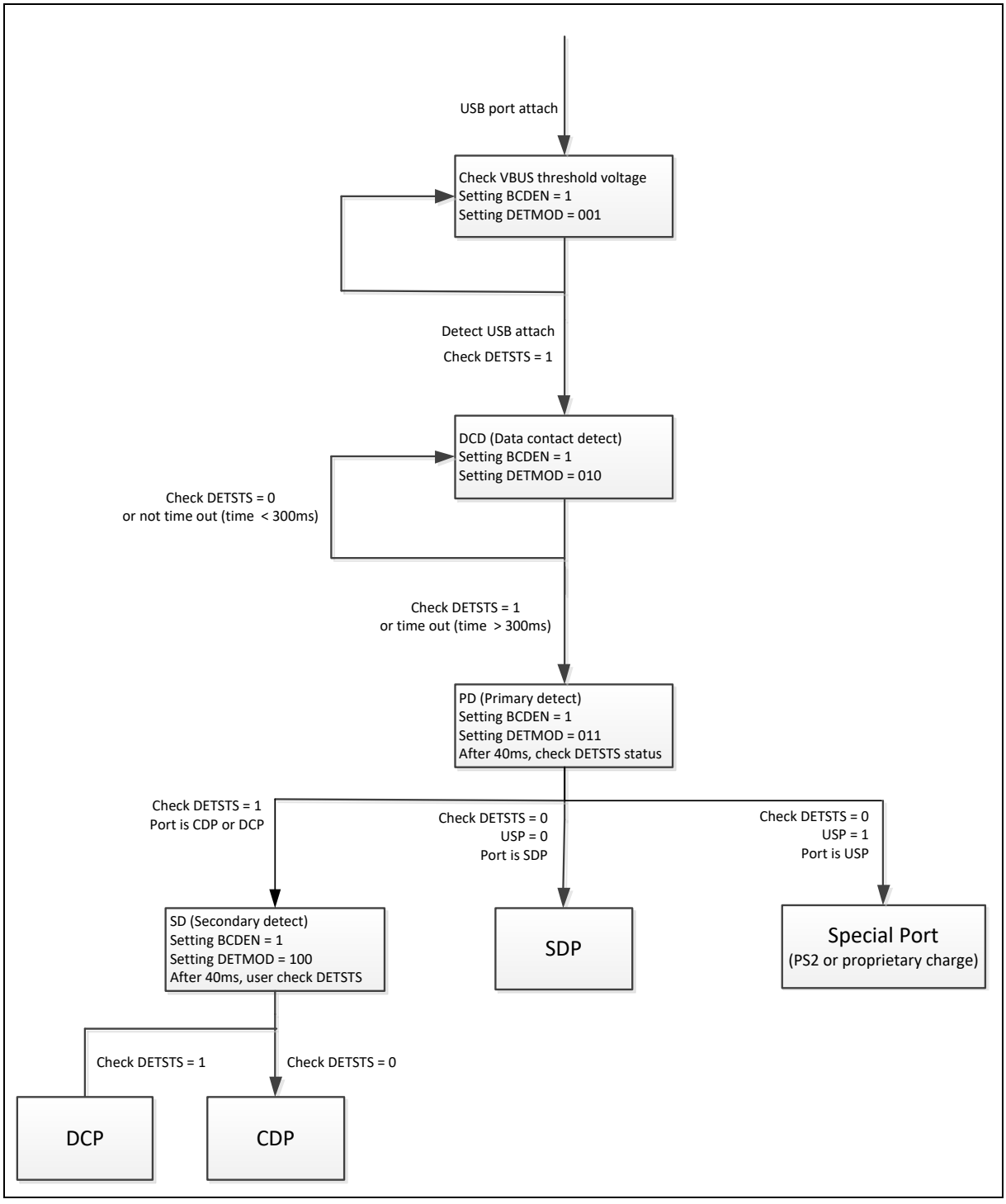


Figure 6.34-2 USB BC 1.2 Detection Flow Block Diagram

6.34.6 Registers Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
USBD Base Address: **HSUSBD_BA = 0x4001_9000**				
HSUSBD_GINTSTS	HSUSBD_BA+0x000	R	Global Interrupt Status Register	0x0000_0000
HSUSBD_GINTEN	HSUSBD_BA+0x008	R/W	Global Interrupt Enable Register	0x0000_0001
HSUSBD_BUSINTSTS	HSUSBD_BA+0x010	R/W	USB Bus Interrupt Status Register	0x0000_0000
HSUSBD_BUSINTEN	HSUSBD_BA+0x014	R/W	USB Bus Interrupt Enable Register	0x0000_0040
HSUSBD_OPER	HSUSBD_BA+0x018	R/W	USB Operational Register	0x0000_0002
HSUSBD_FRAMECNT	HSUSBD_BA+0x01C	R	USB Frame Count Register	0x0000_0000
HSUSBD_FADDR	HSUSBD_BA+0x020	R/W	USB Function Address Register	0x0000_0000
HSUSBD_TEST	HSUSBD_BA+0x024	R/W	USB Test Mode Register	0x0000_0000
HSUSBD_CEPDAT	HSUSBD_BA+0x028	R/W	Control-endpoint Data Buffer	0x0000_0000
HSUSBD_CEPCTL	HSUSBD_BA+0x02C	R/W	Control-endpoint Control Register	0x0000_0000
HSUSBD_CEPINTEN	HSUSBD_BA+0x030	R/W	Control-endpoint Interrupt Enable	0x0000_0000
HSUSBD_CEPINTSTS	HSUSBD_BA+0x034	R/W	Control-endpoint Interrupt Status	0x0000_1800
HSUSBD_CEPTXCNT	HSUSBD_BA+0x038	R/W	Control-endpoint In-transfer Data Count	0x0000_0000
HSUSBD_CEPRXCNT	HSUSBD_BA+0x03C	R	Control-endpoint Out-transfer Data Count	0x0000_0000
HSUSBD_CEPDATCNT	HSUSBD_BA+0x040	R	Control-endpoint Data Count	0x0000_0000
HSUSBD_SETUP1_0	HSUSBD_BA+0x044	R	Setup1 & Setup0 Bytes	0x0000_0000
HSUSBD_SETUP3_2	HSUSBD_BA+0x048	R	Setup3 & Setup2 Bytes	0x0000_0000
HSUSBD_SETUP5_4	HSUSBD_BA+0x04C	R	Setup5 & Setup4 Bytes	0x0000_0000
HSUSBD_SETUP7_6	HSUSBD_BA+0x050	R	Setup7 & Setup6 Bytes	0x0000_0000
HSUSBD_CEPBUFSTART	HSUSBD_BA+0x054	R/W	Control Endpoint RAM Start Address Register	0x0000_0000
HSUSBD_CEPBUFEND	HSUSBD_BA+0x058	R/W	Control Endpoint RAM End Address Register	0x0000_0000
HSUSBD_DMACTL	HSUSBD_BA+0x05C	R/W	DMA Control Status Register	0x0000_0000
HSUSBD_DMACNT	HSUSBD_BA+0x060	R/W	DMA Count Register	0x0000_0000
HSUSBD_EPADAT	HSUSBD_BA+0x064	R/W	Endpoint A Data Register	0x0000_0000
HSUSBD_EPAINTSTS	HSUSBD_BA+0x068	R/W	Endpoint A Interrupt Status Register	0x0000_0003
HSUSBD_EPAINTEN	HSUSBD_BA+0x06C	R/W	Endpoint A Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPADATCNT	HSUSBD_BA+0x070	R	Endpoint A Data Available Count Register	0x0000_0000
HSUSBD_EPARSPCTL	HSUSBD_BA+0x074	R/W	Endpoint A Response Control Register	0x0000_0000
HSUSBD_EPAMPS	HSUSBD_BA+0x078	R/W	Endpoint A Maximum Packet Size Register	0x0000_0000
HSUSBD_EPATXCNT	HSUSBD_BA+0x07C	R/W	Endpoint A Transfer Count Register	0x0000_0000
HSUSBD_EPACFG	HSUSBD_BA+0x080	R/W	Endpoint A Configuration Register	0x0000_0012

HSUSBD_EPABUFSTART	HSUSBD_BA+0x084	R/W	Endpoint A RAM Start Address Register	0x0000_0000
HSUSBD_EPABUFEND	HSUSBD_BA+0x088	R/W	Endpoint A RAM End Address Register	0x0000_0000
HSUSBD_EPBDAT	HSUSBD_BA+0x08C	R/W	Endpoint B Data Register	0x0000_0000
HSUSBD_EPBINTSTS	HSUSBD_BA+0x090	R/W	Endpoint B Interrupt Status Register	0x0000_0003
HSUSBD_EPBINTEN	HSUSBD_BA+0x094	R/W	Endpoint B Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPBDATCNT	HSUSBD_BA+0x098	R	Endpoint B Data Available Count Register	0x0000_0000
HSUSBD_EPBRSPCTL	HSUSBD_BA+0x09C	R/W	Endpoint B Response Control Register	0x0000_0000
HSUSBD_EPBMPS	HSUSBD_BA+0x0A0	R/W	Endpoint B Maximum Packet Size Register	0x0000_0000
HSUSBD_EPBTXCNT	HSUSBD_BA+0x0A4	R/W	Endpoint B Transfer Count Register	0x0000_0000
HSUSBD_EPBCFG	HSUSBD_BA+0x0A8	R/W	Endpoint B Configuration Register	0x0000_0022
HSUSBD_EPBBUFSTART	HSUSBD_BA+0x0AC	R/W	Endpoint B RAM Start Address Register	0x0000_0000
HSUSBD_EPBBUFEND	HSUSBD_BA+0x0B0	R/W	Endpoint B RAM End Address Register	0x0000_0000
HSUSBD_EPCDAT	HSUSBD_BA+0x0B4	R/W	Endpoint C Data Register	0x0000_0000
HSUSBD_EPCINTSTS	HSUSBD_BA+0x0B8	R/W	Endpoint C Interrupt Status Register	0x0000_0003
HSUSBD_EPCINTEN	HSUSBD_BA+0x0BC	R/W	Endpoint C Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPCDATCNT	HSUSBD_BA+0x0C0	R	Endpoint C Data Available Count Register	0x0000_0000
HSUSBD_EPCRSPCTL	HSUSBD_BA+0x0C4	R/W	Endpoint C Response Control Register	0x0000_0000
HSUSBD_EPCMPS	HSUSBD_BA+0x0C8	R/W	Endpoint C Maximum Packet Size Register	0x0000_0000
HSUSBD_EPCTXCNT	HSUSBD_BA+0x0CC	R/W	Endpoint C Transfer Count Register	0x0000_0000
HSUSBD_EPCCFG	HSUSBD_BA+0x0D0	R/W	Endpoint C Configuration Register	0x0000_0032
HSUSBD_EPCBUFSTART	HSUSBD_BA+0x0D4	R/W	Endpoint C RAM Start Address Register	0x0000_0000
HSUSBD_EPCBUFEND	HSUSBD_BA+0x0D8	R/W	Endpoint C RAM End Address Register	0x0000_0000
HSUSBD_EPDDAT	HSUSBD_BA+0x0DC	R/W	Endpoint D Data Register	0x0000_0000
HSUSBD_EPDINTSTS	HSUSBD_BA+0x0E0	R/W	Endpoint D Interrupt Status Register	0x0000_0003
HSUSBD_EPDINTEN	HSUSBD_BA+0x0E4	R/W	Endpoint D Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPDDATCNT	HSUSBD_BA+0x0E8	R	Endpoint D Data Available Count Register	0x0000_0000
HSUSBD_EPDRSPCTL	HSUSBD_BA+0x0EC	R/W	Endpoint D Response Control Register	0x0000_0000
HSUSBD_EPDMPS	HSUSBD_BA+0x0F0	R/W	Endpoint D Maximum Packet Size Register	0x0000_0000
HSUSBD_EPDTXCNT	HSUSBD_BA+0x0F4	R/W	Endpoint D Transfer Count Register	0x0000_0000
HSUSBD_EPDCFG	HSUSBD_BA+0x0F8	R/W	Endpoint D Configuration Register	0x0000_0042
HSUSBD_EPDBUFSTART	HSUSBD_BA+0x0FC	R/W	Endpoint D RAM Start Address Register	0x0000_0000
HSUSBD_EPDBUFEND	HSUSBD_BA+0x100	R/W	Endpoint D RAM End Address Register	0x0000_0000
HSUSBD_EPEDAT	HSUSBD_BA+0x104	R/W	Endpoint E Data Register	0x0000_0000
HSUSBD_EPEINTSTS	HSUSBD_BA+0x108	R/W	Endpoint E Interrupt Status Register	0x0000_0003
HSUSBD_EPEINTEN	HSUSBD_BA+0x10C	R/W	Endpoint E Interrupt Enable Control Register	0x0000_0000

HSUSBD_EPEDATCNT	HSUSBD_BA+0x110	R	Endpoint E Data Available Count Register	0x0000_0000
HSUSBD_EPERSPCTL	HSUSBD_BA+0x114	R/W	Endpoint E Response Control Register	0x0000_0000
HSUSBD_EPEMPS	HSUSBD_BA+0x118	R/W	Endpoint E Maximum Packet Size Register	0x0000_0000
HSUSBD_EPETXCNT	HSUSBD_BA+0x11C	R/W	Endpoint E Transfer Count Register	0x0000_0000
HSUSBD_EPECFG	HSUSBD_BA+0x120	R/W	Endpoint E Configuration Register	0x0000_0052
HSUSBD_EPEBUFSTART	HSUSBD_BA+0x124	R/W	Endpoint E RAM Start Address Register	0x0000_0000
HSUSBD_EPEBUFEND	HSUSBD_BA+0x128	R/W	Endpoint E RAM End Address Register	0x0000_0000
HSUSBD_EPFDAT	HSUSBD_BA+0x12C	R/W	Endpoint F Data Register	0x0000_0000
HSUSBD_EPFINTSTS	HSUSBD_BA+0x130	R/W	Endpoint F Interrupt Status Register	0x0000_0003
HSUSBD_EPFINTEN	HSUSBD_BA+0x134	R/W	Endpoint F Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPFDATCNT	HSUSBD_BA+0x138	R	Endpoint F Data Available Count Register	0x0000_0000
HSUSBD_EPFRSPCTL	HSUSBD_BA+0x13C	R/W	Endpoint F Response Control Register	0x0000_0000
HSUSBD_EPFMPS	HSUSBD_BA+0x140	R/W	Endpoint F Maximum Packet Size Register	0x0000_0000
HSUSBD_EPFTXCNT	HSUSBD_BA+0x144	R/W	Endpoint F Transfer Count Register	0x0000_0000
HSUSBD_EPFCFG	HSUSBD_BA+0x148	R/W	Endpoint F Configuration Register	0x0000_0062
HSUSBD_EPFBUFSTART	HSUSBD_BA+0x14C	R/W	Endpoint F RAM Start Address Register	0x0000_0000
HSUSBD_EPFBUFEND	HSUSBD_BA+0x150	R/W	Endpoint F RAM End Address Register	0x0000_0000
HSUSBD_EPGDAT	HSUSBD_BA+0x154	R/W	Endpoint G Data Register	0x0000_0000
HSUSBD_EPGINTSTS	HSUSBD_BA+0x158	R/W	Endpoint G Interrupt Status Register	0x0000_0003
HSUSBD_EPGINTEN	HSUSBD_BA+0x15C	R/W	Endpoint G Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPGDATCNT	HSUSBD_BA+0x160	R	Endpoint G Data Available Count Register	0x0000_0000
HSUSBD_EPGRSPCTL	HSUSBD_BA+0x164	R/W	Endpoint G Response Control Register	0x0000_0000
HSUSBD_EPGMPS	HSUSBD_BA+0x168	R/W	Endpoint G Maximum Packet Size Register	0x0000_0000
HSUSBD_EPGTXCNT	HSUSBD_BA+0x16C	R/W	Endpoint G Transfer Count Register	0x0000_0000
HSUSBD_EPGCFG	HSUSBD_BA+0x170	R/W	Endpoint G Configuration Register	0x0000_0072
HSUSBD_EPGBUFSTART	HSUSBD_BA+0x174	R/W	Endpoint G RAM Start Address Register	0x0000_0000
HSUSBD_EPGBUFEND	HSUSBD_BA+0x178	R/W	Endpoint G RAM End Address Register	0x0000_0000
HSUSBD_EPHDAT	HSUSBD_BA+0x17C	R/W	Endpoint H Data Register	0x0000_0000
HSUSBD_EPHINTSTS	HSUSBD_BA+0x180	R/W	Endpoint H Interrupt Status Register	0x0000_0003
HSUSBD_EPHINTEN	HSUSBD_BA+0x184	R/W	Endpoint H Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPHDATCNT	HSUSBD_BA+0x188	R	Endpoint H Data Available Count Register	0x0000_0000
HSUSBD_EPHRSPCTL	HSUSBD_BA+0x18C	R/W	Endpoint H Response Control Register	0x0000_0000
HSUSBD_EPHMPS	HSUSBD_BA+0x190	R/W	Endpoint H Maximum Packet Size Register	0x0000_0000
HSUSBD_EPHTXCNT	HSUSBD_BA+0x194	R/W	Endpoint H Transfer Count Register	0x0000_0000
HSUSBD_EPHCFG	HSUSBD_BA+0x198	R/W	Endpoint H Configuration Register	0x0000_0082

HSUSBD_EPHBUFSTART	HSUSBD_BA+0x19C	R/W	Endpoint H RAM Start Address Register	0x0000_0000
HSUSBD_EPHBUFEND	HSUSBD_BA+0x1A0	R/W	Endpoint H RAM End Address Register	0x0000_0000
HSUSBD_EPIDAT	HSUSBD_BA+0x1A4	R/W	Endpoint I Data Register	0x0000_0000
HSUSBD_EPIINTSTS	HSUSBD_BA+0x1A8	R/W	Endpoint I Interrupt Status Register	0x0000_0003
HSUSBD_EPIINTEN	HSUSBD_BA+0x1AC	R/W	Endpoint I Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPIDATCNT	HSUSBD_BA+0x1B0	R	Endpoint I Data Available Count Register	0x0000_0000
HSUSBD_EPIRSPCTL	HSUSBD_BA+0x1B4	R/W	Endpoint I Response Control Register	0x0000_0000
HSUSBD_EPIMPS	HSUSBD_BA+0x1B8	R/W	Endpoint I Maximum Packet Size Register	0x0000_0000
HSUSBD_EPITXCNT	HSUSBD_BA+0x1BC	R/W	Endpoint I Transfer Count Register	0x0000_0000
HSUSBD_EPICFG	HSUSBD_BA+0x1C0	R/W	Endpoint I Configuration Register	0x0000_0092
HSUSBD_EPIBUFSTART	HSUSBD_BA+0x1C4	R/W	Endpoint I RAM Start Address Register	0x0000_0000
HSUSBD_EPIBUFEND	HSUSBD_BA+0x1C8	R/W	Endpoint I RAM End Address Register	0x0000_0000
HSUSBD_EPJDAT	HSUSBD_BA+0x1CC	R/W	Endpoint J Data Register	0x0000_0000
HSUSBD_EPJINTSTS	HSUSBD_BA+0x1D0	R/W	Endpoint J Interrupt Status Register	0x0000_0003
HSUSBD_EPJINTEN	HSUSBD_BA+0x1D4	R/W	Endpoint J Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPJDATCNT	HSUSBD_BA+0x1D8	R	Endpoint J Data Available Count Register	0x0000_0000
HSUSBD_EPJRSPCTL	HSUSBD_BA+0x1DC	R/W	Endpoint J Response Control Register	0x0000_0000
HSUSBD_EPJMPS	HSUSBD_BA+0x1E0	R/W	Endpoint J Maximum Packet Size Register	0x0000_0000
HSUSBD_EPJTXCNT	HSUSBD_BA+0x1E4	R/W	Endpoint J Transfer Count Register	0x0000_0000
HSUSBD_EPJCFG	HSUSBD_BA+0x1E8	R/W	Endpoint J Configuration Register	0x0000_00A2
HSUSBD_EPJBUFSTART	HSUSBD_BA+0x1EC	R/W	Endpoint J RAM Start Address Register	0x0000_0000
HSUSBD_EPJBUFEND	HSUSBD_BA+0x1F0	R/W	Endpoint J RAM End Address Register	0x0000_0000
HSUSBD_EPKDAT	HSUSBD_BA+0x1F4	R/W	Endpoint K Data Register	0x0000_0000
HSUSBD_EPKINTSTS	HSUSBD_BA+0x1F8	R/W	Endpoint K Interrupt Status Register	0x0000_0003
HSUSBD_EPKINTEN	HSUSBD_BA+0x1FC	R/W	Endpoint K Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPKDATCNT	HSUSBD_BA+0x200	R	Endpoint K Data Available Count Register	0x0000_0000
HSUSBD_EPKRSPCTL	HSUSBD_BA+0x204	R/W	Endpoint K Response Control Register	0x0000_0000
HSUSBD_EPKMPS	HSUSBD_BA+0x208	R/W	Endpoint K Maximum Packet Size Register	0x0000_0000
HSUSBD_EPKTXCNT	HSUSBD_BA+0x20C	R/W	Endpoint K Transfer Count Register	0x0000_0000
HSUSBD_EPKCFG	HSUSBD_BA+0x210	R/W	Endpoint K Configuration Register	0x0000_00B2
HSUSBD_EPKBUFSTART	HSUSBD_BA+0x214	R/W	Endpoint K RAM Start Address Register	0x0000_0000
HSUSBD_EPKBUFEND	HSUSBD_BA+0x218	R/W	Endpoint K RAM End Address Register	0x0000_0000
HSUSBD_EPLDAT	HSUSBD_BA+0x21C	R/W	Endpoint L Data Register	0x0000_0000
HSUSBD_EPLINTSTS	HSUSBD_BA+0x220	R/W	Endpoint L Interrupt Status Register	0x0000_0003
HSUSBD_EPLINTEN	HSUSBD_BA+0x224	R/W	Endpoint L Interrupt Enable Control Register	0x0000_0000

HSUSBD_EPLDATCNT	HSUSBD_BA+0x228	R	Endpoint L Data Available Count Register	0x0000_0000
HSUSBD_EPLRSPCTL	HSUSBD_BA+0x22C	R/W	Endpoint L Response Control Register	0x0000_0000
HSUSBD_EPLMPS	HSUSBD_BA+0x230	R/W	Endpoint L Maximum Packet Size Register	0x0000_0000
HSUSBD_EPLTXCNT	HSUSBD_BA+0x234	R/W	Endpoint L Transfer Count Register	0x0000_0000
HSUSBD_EPLCFG	HSUSBD_BA+0x238	R/W	Endpoint L Configuration Register	0x0000_00C2
HSUSBD_EPLBUFSTART	HSUSBD_BA+0x23C	R/W	Endpoint L RAM Start Address Register	0x0000_0000
HSUSBD_EPLBUFEND	HSUSBD_BA+0x240	R/W	Endpoint L RAM End Address Register	0x0000_0000
HSUSBD_BCDC	HSUSBD_BA+0x6F8	R/W	Battery Charge Detect Control Register	0x0000_0000
HSUSBD_LPMCSR	HSUSBD_BA+0x6FC	R/W	LPM Control and Status Register	0x0000_0000
HSUSBD_DMAADDR	HSUSBD_BA+0x700	R/W	AHB DMA Address Register	0x0000_0000
HSUSBD_PHYCTL	HSUSBD_BA+0x704	R/W	USB PHY Control Register	0x0000_0000

6.34.7 Register Description

Global Interrupt Status Register (HSUSBD_GINTSTS)

Register	Offset	R/W	Description	Reset Value
HSUSBD_GINTSTS	HSUSBD_BA+0x000	R	Global Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		EPLIF	EPKIF	EPJIF	EPIIF	EPHIF	EPGIF
7	6	5	4	3	2	1	0
EPFIF	EPEIF	EPDIF	EPCIF	EPBIF	EPAIF	CEPIF	USBIF

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**EPLIF**	**Endpoint L Interrupt** When set, the corresponding Endpoint L's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[12]	**EPKIF**	**Endpoint K Interrupt** When set, the corresponding Endpoint K's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[11]	**EPJIF**	**Endpoint J Interrupt** When set, the corresponding Endpoint J's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[10]	**EPIIF**	**Endpoint I Interrupt** When set, the corresponding Endpoint I's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[9]	**EPHIF**	**Endpoint H Interrupt** When set, the corresponding Endpoint H's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.

[8]	**EPGIF**	**Endpoint G Interrupt** When set, the corresponding Endpoint G's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[7]	**EPFIF**	**Endpoint F Interrupt** When set, the corresponding Endpoint F's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[6]	**EPEIF**	**Endpoint E Interrupt** When set, the corresponding Endpoint E's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[5]	**EPDIF**	**Endpoint D Interrupt** When set, the corresponding Endpoint D's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[4]	**EPCIF**	**Endpoint C Interrupt** When set, the corresponding Endpoint C's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[3]	**EPBIF**	**Endpoint B Interrupt** When set, the corresponding Endpoint B's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[2]	**EPAIF**	**Endpoint A Interrupt** When set, the corresponding Endpoint A's interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[1]	**CEPIF**	**Control Endpoint Interrupt** This bit conveys the interrupt status for control endpoint. When set, Control-ep’s interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.
[0]	**USBIF**	**USB Interrupt** This bit conveys the interrupt status for USB specific events endpoint. When set, USB interrupt status register should be read to determine the cause of the interrupt. 0 = No interrupt event occurred. 1 = The related interrupt event occurred.

Global Interrupt Enable Register (HSUSBD_GINTEN)

Register	Offset	R/W	Description	Reset Value
HSUSBD_GINTEN	HSUSBD_BA+0x008	R/W	Global Interrupt Enable Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**EPLIEN**	**EPKIEN**	**EPJIEN**	**EPIIEN**	**EPHIEN**	**EPGIEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EPFIEN	**EPEIEN**	**EPDIEN**	**EPCIEN**	**EPBIEN**	**EPAIEN**	**CEPIEN**	**USBIEN**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**EPLIEN**	**Interrupt Enable Control for Endpoint L** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint L 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[12]	**EPKIEN**	**Interrupt Enable Control for Endpoint K** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint K 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[11]	**EPJIEN**	**Interrupt Enable Control for Endpoint J** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint J 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[10]	**EPIIEN**	**Interrupt Enable Control for Endpoint I** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint I 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[9]	**EPHIEN**	**Interrupt Enable Control for Endpoint H** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint H 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[8]	**EPGIEN**	**Interrupt Enable Control for Endpoint G** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint G 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.

[7]	**EPFIEN**	**Interrupt Enable Control for Endpoint F** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint F 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[6]	**EPEIEN**	**Interrupt Enable Control for Endpoint E** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint E 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[5]	**EPDIEN**	**Interrupt Enable Control for Endpoint D** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint D 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[4]	**EPCIEN**	**Interrupt Enable Control for Endpoint C** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint C 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[3]	**EPBIEN**	**Interrupt Enable Control for Endpoint B** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint B 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[2]	**EPAIEN**	**Interrupt Enable Control for Endpoint A** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint A. 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[1]	**CEPIEN**	**Control Endpoint Interrupt Enable Bit** When set, this bit enables a local interrupt to be generated when an interrupt is pending for the control endpoint. 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.
[0]	**USBIEN**	**USB Interrupt Enable Bit** When set, this bit enables a local interrupt to be generated when a USB event occurs on the bus. 0 = The related interrupt Disabled. 1 = The related interrupt Enabled.

USB Bus Interrupt Status Register (HSUSBD_BUSINTSTS)

Register	Offset	R/W	Description	Reset Value
HSUSBD_BUSINTSTS	HSUSBD_BA+0x010	R/W	USB Bus Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						LPMTKNIF	VBUSDETIF
7	6	5	4	3	2	1	0
Reserved	PHYCLKVLDIF	DMADONEIF	HISPDIF	SUSPENDIF	RESUMEIF	RSTIF	SOFIF

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**LPMTKNIF**	**LPM Token Interrupt** 0 = No LPM token has been received from the host. 1 = A LPM token has been received from the host. **Note** : Write 1 to clear this bit to 0.
[8]	**VBUSDETIF**	**V_{BUS} Detection Interrupt** 0 = No V_{BUS} is plug-in. 1 = V_{BUS} is plug-in. **Note:** Write 1 to clear this bit to 0.
[7]	**Reserved**	Reserved.
[6]	**PHYCLKVLDIF**	**Usable Clock Interrupt** 0 = Usable clock is not available. 1 = Usable clock is available from the transceiver. **Note:** Write 1 to clear this bit to 0.
[5]	**DMADONEIF**	**DMA Completion Interrupt** 0 = No DMA transfer over. 1 = DMA transfer is over. **Note:** Write 1 to clear this bit to 0.
[4]	**HISPDIF**	**High-speed Settle Interrupt** 0 = No valid high-speed reset protocol is detected. 1 = Valid high-speed reset protocol is over and the device has settled in high-speed. **Note:** Write 1 to clear this bit to 0.

[3]	**SUSPENDIF**	**Suspend Request Interrupt** This bit is set as default and it has to be cleared by writing '1' before the USB reset. This bit is also set when a USB Suspend request is detected from the host. 0 = No USB Suspend request is detected from the host. 1= USB Suspend request is detected from the host. **Note:** Write 1 to clear this bit to 0.
[2]	**RESUMEIF**	**Resume Interrupt** When set, this bit indicates that a device resume has occurred. 0 = No device resume has occurred. 1 = Device resume has occurred. **Note:** Write 1 to clear this bit to 0.
[1]	**RSTIF**	**Reset Interrupt** When set, this bit indicates that either the USB root port reset is end. 0 = No USB root port reset is end. 1 = USB root port reset is end. **Note:** Write 1 to clear this bit to 0.
[0]	**SOFIF**	**SOF Receive Control Interrupt** This bit indicates when a start-of-frame packet has been received. 0 = No start-of-frame packet has been received. 1 = Start-of-frame packet has been received. **Note:** Write 1 to clear this bit to 0.

USB Bus Interrupt Enable Register (HSUSBD_BUSINTEN)

Register	Offset	R/W	Description	Reset Value
HSUSBD_BUSINTEN	HSUSBD_BA+0x014	R/W	USB Bus Interrupt Enable Register	0x0000_0040

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						LPMTKNIEN	VBUSDETIEN
7	6	5	4	3	2	1	0
Reserved	PHYCLKVLDIEN	DMADONEIEN	HISPDIEN	SUSPENDIEN	RESUMEIEN	RSTIEN	SOFIEN

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**LPMTKNIEN**	**LPM Token Interrupt Enable Bit** 0 = The LPM token interrupt Disabled. 1 = The LPM token interrupt Enabled.
[8]	**VBUSDETIEN**	**V_{BUS} Detection Interrupt Enable Bit** This bit enables the V_{BUS} floating detection interrupt. 0 = V_{BUS} floating detection interrupt Disabled. 1 = V_{BUS} floating detection interrupt Enabled.
[7]	**Reserved**	Reserved.
[6]	**PHYCLKVLDIEN**	**Usable Clock Interrupt Enable Bit** This bit enables the usable clock interrupt. 0 = Usable clock interrupt Disabled. 1 = Usable clock interrupt Enabled.
[5]	**DMADONEIEN**	**DMA Completion Interrupt Enable Bit** This bit enables the DMA completion interrupt 0 = DMA completion interrupt Disabled. 1 = DMA completion interrupt Enabled.
[4]	**HISPDIEN**	**High-speed Settle Interrupt Enable Bit** This bit enables the high-speed settle interrupt. 0 = High-speed settle interrupt Disabled. 1 = High-speed settle interrupt Enabled.
[3]	**SUSPENDIEN**	**Suspend Request Interrupt Enable Bit** This bit enables the Suspend interrupt. 0 = Suspend interrupt Disabled. 1 = Suspend interrupt Enabled.

[2]	**RESUMEIEN**	**Resume Interrupt Enable Bit** This bit enables the Resume interrupt. 0 = Resume interrupt Disabled. 1 = Resume interrupt Enabled.
[1]	**RSTIEN**	**Reset Interrupt Enable Bit** This bit enables the USB-Reset interrupt. 0 = USB-Reset interrupt Disabled. 1 = USB-Reset interrupt Enabled.
[0]	**SOFIEN**	**SOF Interrupt Enable Bit** This bit enables the SOF interrupt. 0 = SOF interrupt Disabled. 1 = SOF interrupt Enabled.

USB Operational Register (HSUSBD_OPER)

Register	Offset	R/W	Description	Reset Value
HSUSBD_OPER	HSUSBD_BA+0x018	R/W	USB Operational Register	0x0000_0002

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**CURSPD**	**HISPDEN**	**RESUMEEN**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**CURSPD**	**USB Current Speed** 0 = The device has settled in full-speed. 1 = The USB device controller has settled in high-speed.
[1]	**HISPDEN**	**USB High-speed** 0 = The USB device controller to suppress the chirp-sequence during reset protocol, thereby allowing the USB device controller to settle in full-speed, even though it is connected to a USB2.0 Host. 1 = The USB device controller to initiate a chirp-sequence during reset protocol.
[0]	**RESUMEEN**	**Generate Resume** 0 = No Resume sequence to be initiated to the host. 1 = A Resume sequence to be initiated to the host if device remote wakeup is enabled. This bit is self-clearing.

USB Frame Count Register (HSUSBD_FRAMECNT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_FRAMECNT	HSUSBD_BA+0x01C	R	USB Frame Count Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		FRAMECNT					
7	6	5	4	3	2	1	0
FRAMECNT					MFRAMECNT		

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:3]	**FRAMECNT**	**Frame Counter** This field contains the frame count from the most recent start-of-frame packet.
[2:0]	**MFRAMECNT**	**Micro-frame Counter** This field contains the micro-frame number for the frame number in the frame counter field.

USB Function Address Register (HSUSBD_FADDR)

Register	Offset	R/W	Description	Reset Value
HSUSBD_FADDR	HSUSBD_BA+0x020	R/W	USB Function Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**FADDR**						

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:0]	**FADDR**	**USB Function Address** This field contains the current USB address of the device. This field is cleared when a root port reset is detected.

USB Test Mode Register (HSUSBD_TEST)

Register	Offset	R/W	Description	Reset Value
HSUSBD_TEST	HSUSBD_BA+0x024	R/W	USB Test Mode Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**TESTMODE**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**TESTMODE**	**Test Mode Selection** 000 = Normal Operation. 001 = Test_J. 010 = Test_K. 011 = Test_SE0_NAK. 100 = Test_Packet. 101 = Test_Force_Enable. 110 = Reserved. 111 = Reserved. **Note:** This field is cleared when root port reset is detected.

Control-endpoint Data Buffer (HSUSBD_CEPDAT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPDAT	HSUSBD_BA+0x028	R/W	Control-endpoint Data Buffer	0x0000_0000

31	30	29	28	27	26	25	24
DAT							
23	22	21	20	19	18	17	16
DAT							
15	14	13	12	11	10	9	8
DAT							
7	6	5	4	3	2	1	0
DAT							

Bits	Description	
[31:0]	**DAT**	**Control-endpoint Data Buffer** Control endpoint data buffer for the buffer transaction (read or write). **Note:** Only word or byte access are supported.

Control-endpoint Control Register (HSUSBD_CEPCTL)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPCTL	HSUSBD_BA+0x02C	R/W	Control-endpoint Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				FLUSH	ZEROLEN	STALLEN	NAKCLR

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**FLUSH**	**CEP-flush Bit** 0 = No the packet buffer and its corresponding HSUSBD_CEPDATCNT register to be cleared. 1 = The packet buffer and its corresponding HSUSBD_CEPDATCNT register to be cleared. This bit is self-cleaning.
[2]	**ZEROLEN**	**Zero Packet Length** This bit is valid for Auto Validation mode only. 0 = No zero length packet to the host during Data stage to an IN token. 1 = USB device controller can send a zero length packet to the host during Data stage to an IN token. This bit gets cleared once the zero length data packet is sent. So, the local CPU need not write again to clear this bit.
[1]	**STALLEN**	**Stall Enable Bit** When this stall bit is set, the control endpoint sends a stall handshake in response to any in or out token thereafter. This is typically used for response to invalid/unsupported requests. When this bit is being set the NAK clear bit has to be cleared at the same time since the NAK clear bit has highest priority than STALL. It is automatically cleared on receipt of a next setup-token. So, the local CPU need not write again to clear this bit. 0 = No sends a stall handshake in response to any in or out token thereafter. 1 = The control endpoint sends a stall handshake in response to any in or out token thereafter. **Note:** Only when CPU writes data[1:0] is 2'b10 or 2'b00, this bit can be updated.
[0]	**NAKCLR**	**No Acknowledge Control** This bit plays a crucial role in any control transfer. 0 = The bit is being cleared by the local CPU by writing zero, the USB device controller will be responding with NAKs for the subsequent status phase. This mechanism holds the host from moving to the next request, until the local CPU is also ready to process the next request. 1 = This bit is set to one by the USB device controller, whenever a setup token is received. The local CPU can take its own time to finish off any house-keeping work based on the request and then clear this bit. **Note:** Only when CPU writes data[1:0] is 2'b10 or 2'b00, this bit can be updated.

Control-endpoint Interrupt Enable (HSUSBD_CEPINTEN)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPINTEN	HSUSBD_BA+0x030	R/W	Control-endpoint Interrupt Enable	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			BUFEMPTYIEN	BUFFULLIEN	STSDONEIEN	ERRIEN	STALLIEN
7	6	5	4	3	2	1	0
NAKIEN	RXPKIEN	TXPKIEN	PINGIEN	INTKIEN	OUTTKIEN	SETUPPKIEN	SETUPTKIEN

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**BUFEMPTYIEN**	**Buffer Empty Interrupt Enable Bit** 0 = The buffer empty interrupt in Control Endpoint Disabled. 1= The buffer empty interrupt in Control Endpoint Enabled.
[11]	**BUFFULLIEN**	**Buffer Full Interrupt Enable Bit** 0 = The buffer full interrupt in Control Endpoint Disabled. 1 = The buffer full interrupt in Control Endpoint Enabled.
[10]	**STSDONEIEN**	**Status Completion Interrupt Enable Bit** 0 = The Status Completion interrupt in Control Endpoint Disabled. 1 = The Status Completion interrupt in Control Endpoint Enabled.
[9]	**ERRIEN**	**USB Error Interrupt Enable Bit** 0 = The USB Error interrupt in Control Endpoint Disabled. 1 = The USB Error interrupt in Control Endpoint Enabled.
[8]	**STALLIEN**	**STALL Sent Interrupt Enable Bit** 0 = The STALL sent interrupt in Control Endpoint Disabled. 1 = The STALL sent interrupt in Control Endpoint Enabled.
[7]	**NAKIEN**	**NAK Sent Interrupt Enable Bit** 0 = The NAK sent interrupt in Control Endpoint Disabled. 1 = The NAK sent interrupt in Control Endpoint Enabled.
[6]	**RXPKIEN**	**Data Packet Received Interrupt Enable Bit** 0 = The data received interrupt in Control Endpoint Disabled. 1 = The data received interrupt in Control Endpoint Enabled.
[5]	**TXPKIEN**	**Data Packet Transmitted Interrupt Enable Bit** 0 = The data packet transmitted interrupt in Control Endpoint Disabled. 1 = The data packet transmitted interrupt in Control Endpoint Enabled.

[4]	**PINGIEN**	**Ping Token Interrupt Enable Bit** 0 = The ping token interrupt in Control Endpoint Disabled. 1 = The ping token interrupt Control Endpoint Enabled.
[3]	**INTKIEN**	**In Token Interrupt Enable Bit** 0 = The IN token interrupt in Control Endpoint Disabled. 1 = The IN token interrupt in Control Endpoint Enabled.
[2]	**OUTTKIEN**	**Out Token Interrupt Enable Bit** 0 = The OUT token interrupt in Control Endpoint Disabled. 1 = The OUT token interrupt in Control Endpoint Enabled.
[1]	**SETUPPKIEN**	**Setup Packet Interrupt Enable Bit** 0 = The SETUP packet interrupt in Control Endpoint Disabled. 1 = The SETUP packet interrupt in Control Endpoint Enabled.
[0]	**SETUPTKIEN**	**Setup Token Interrupt Enable Bit** 0 = The SETUP token interrupt in Control Endpoint Disabled. 1 = The SETUP token interrupt in Control Endpoint Enabled.

Control-endpoint Interrupt Status (HSUSBD_CEPINTSTS)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPINTSTS	HSUSBD_BA+0x034	R/W	Control-endpoint Interrupt Status	0x0000_1800

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			BUFEMPTYIF	BUFFULLIF	STSDONEIF	ERRIF	STALLIF
7	6	5	4	3	2	1	0
NAKIF	RXPKIF	TXPKIF	PINGIF	INTKIF	OUTTKIF	SETUPPKIF	SETUPTKIF

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**BUFEMPTYIF**	**Buffer Empty Interrupt** 0 = The control-endpoint buffer is not empty. 1 = The control-endpoint buffer is empty. **Note:** Write 1 to clear this bit to 0.
[11]	**BUFFULLIF**	**Buffer Full Interrupt** 0 = The control-endpoint buffer is not full. 1 = The control-endpoint buffer is full. **Note:** Write 1 to clear this bit to 0.
[10]	**STSDONEIF**	**Status Completion Interrupt** 0 = Not a USB transaction has completed successfully. 1 = The status stage of a USB transaction has completed successfully. **Note:** Write 1 to clear this bit to 0.
[9]	**ERRIF**	**USB Error Interrupt** 0 = No error had occurred during the transaction. 1 = An error had occurred during the transaction. **Note:** Write 1 to clear this bit to 0.
[8]	**STALLIF**	**STALL Sent Interrupt** 0 = Not a stall-token is sent in response to an IN/OUT token. 1 = A stall-token is sent in response to an IN/OUT token. **Note:** Write 1 to clear this bit to 0.
[7]	**NAKIF**	**NAK Sent Interrupt** 0 = Not a NAK-token is sent in response to an IN/OUT token. 1 = A NAK-token is sent in response to an IN/OUT token. **Note:** Write 1 to clear this bit to 0.

[6]	**RXPKIF**	**Data Packet Received Interrupt** 0 = Not a data packet is successfully received from the host for an OUT-token and an ACK is sent to the host. 1 = A data packet is successfully received from the host for an OUT-token and an ACK is sent to the host. **Note:** Write 1 to clear this bit to 0.
[5]	**TXPKIF**	**Data Packet Transmitted Interrupt** 0 = Not a data packet is successfully transmitted to the host in response to an IN-token and an ACK-token is received for the same. 1 = A data packet is successfully transmitted to the host in response to an IN-token and an ACK-token is received for the same. **Note:** Write 1 to clear this bit to 0.
[4]	**PINGIF**	**Ping Token Interrupt** 0 = The control-endpoint does not receive a ping token from the host. 1 = The control-endpoint receives a ping token from the host. **Note:** Write 1 to clear this bit to 0.
[3]	**INTKIF**	**In Token Interrupt** 0 = The control-endpoint does not receive an IN token from the host. 1 = The control-endpoint receives an IN token from the host. **Note:** Write 1 to clear this bit to 0.
[2]	**OUTTKIF**	**Out Token Interrupt** 0 = The control-endpoint does not receive an OUT token from the host. 1 = The control-endpoint receives an OUT token from the host. **Note:** Write 1 to clear this bit to 0.
[1]	**SETUPPKIF**	**Setup Packet Interrupt** This bit must be cleared (by writing 1) before the next setup packet can be received. If the bit is not cleared, then the successive setup packets will be overwritten in the setup packet buffer. 0 = Not a Setup packet has been received from the host. 1 = A Setup packet has been received from the host. **Note:** Write 1 to clear this bit to 0.
[0]	**SETUPTKIF**	**Setup Token Interrupt** 0 = Not a Setup token is received. 1 = A Setup token is received. Writing 1 clears this status bit **Note:** Write 1 to clear this bit to 0.

Control-endpoint In-transfer Data Count (HSUSBD_CEPTXCNT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPTXCNT	HSUSBD_BA+0x038	R/W	Control-endpoint In-transfer Data Count	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
TXCNT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**TXCNT**	**In-transfer Data Count** There is no mode selection for the control endpoint (but it operates like manual mode). The local-CPU has to fill the control-endpoint buffer with the data to be sent for an in-token and to write the count of bytes in this register. When zero is written into this field, a zero length packet is sent to the host. When the count written in the register is more than the MPS, the data sent will be of only MPS.

Control-endpoint Out-transfer Data Count (HSUSBD_CEPRXCNT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPRXCNT	HSUSBD_BA+0x03C	R	Control-endpoint Out-transfer Data Count	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RXCNT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**RXCNT**	**Out-transfer Data Count** The USB device controller maintains the count of the data received in case of an out transfer, during the control transfer.

Control-endpoint Data Count (HSUSBD_CEPDATCNT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPDATCNT	HSUSBD_BA+0x040	R	Control-endpoint Data Count	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DATCNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATCNT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATCNT**	**Control-endpoint Data Count** The USB device controller maintains the count of the data of control-endpoint.

Setup1 & Setup0 Bytes (HSUSBD_SETUP1_0)

Register	Offset	R/W	Description	Reset Value
HSUSBD_SETUP1_0	HSUSBD_BA+0x044	R	Setup1 & Setup0 Bytes	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
SETUP1							
7	6	5	4	3	2	1	0
SETUP0							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**SETUP1**	**Setup Byte 1[15:8]** This register provides byte 1 of the last setup packet received. For a Standard Device Request, the following bRequest Code information is returned. 00000000 = Get Status. 00000001 = Clear Feature. 00000010 = Reserved. 00000011 = Set Feature. 00000100 = Reserved. 00000101 = Set Address. 00000110 = Get Descriptor. 00000111 = Set Descriptor. 00001000 = Get Configuration. 00001001 = Set Configuration. 00001010 = Get Interface. 00001011 = Set Interface. 00001100 = Sync Frame.

[7:0]	**SETUP0**	**Setup Byte 0[7:0]** This register provides byte 0 of the last setup packet received. For a Standard Device Request, the following bmRequestType information is returned. Bit 7 (Direction): 0 = Host to dev..ice 1 = Device to h..ost Bit 6-5 (Type): 00 = Stand..ard 01 = Cl..ass 10 = Ven..dor 11 = Reser..ved Bit 4-0 (Recipient) 00000 = Dev..ice 00001 = Interf..ace 00010 = Endpo..int 00011 = Ot..her Others = Reser..ved

Setup3 & Setup2 Bytes (HSUSBD_SETUP3_2)

Register	Offset	R/W	Description	Reset Value
HSUSBD_SETUP3_2	HSUSBD_BA+0x048	R	Setup3 & Setup2 Bytes	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
SETUP3							
7	6	5	4	3	2	1	0
SETUP2							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:8]	SETUP3	**Setup Byte 3[15:8]** This register provides byte 3 of the last setup packet received. For a Standard Device Request, the most significant byte of the wValue field is returned.
[7:0]	SETUP2	**Setup Byte 2[7:0]** This register provides byte 2 of the last setup packet received. For a Standard Device Request, the least significant byte of the wValue field is returned.

Setup5 & Setup4 Bytes (HSUSBD_SETUP5_4)

Register	Offset	R/W	Description	Reset Value
HSUSBD_SETUP5_4	HSUSBD_BA+0x04C	R	Setup5 & Setup4 Bytes	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
SETUP5							
7	6	5	4	3	2	1	0
SETUP4							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**SETUP5**	**Setup Byte 5[15:8]** This register provides byte 5 of the last setup packet received. For a Standard Device Request, the most significant byte of the wIndex field is returned.
[7:0]	**SETUP4**	**Setup Byte 4[7:0]** This register provides byte 4 of the last setup packet received. For a Standard Device Request, the least significant byte of the wIndex field is returned.

Setup7 & Setup6 Bytes (HSUSBD_SETUP7_6)

Register	Offset	R/W	Description	Reset Value
HSUSBD_SETUP7_6	HSUSBD_BA+0x050	R	Setup7 & Setup6 Bytes	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETUP7							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETUP6							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**SETUP7**	**Setup Byte 7[15:8]** This register provides byte 7 of the last setup packet received. For a Standard Device Request, the most significant byte of the wLength field is returned.
[7:0]	**SETUP6**	**Setup Byte 6[7:0]** This register provides byte 6 of the last setup packet received. For a Standard Device Request, the least significant byte of the wLength field is returned.

Control Endpoint RAM Start Address Register (HSUSBD_CEPBUFSTART)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPBUFSTART	HSUSBD_BA+0x054	R/W	Control Endpoint RAM Start Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				SADDR			
7	6	5	4	3	2	1	0
SADDR							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**SADDR**	**Control-endpoint Start Address** This is the start-address of the RAM space allocated for the control-endpoint.

Control Endpoint RAM End Address Register (HSUSBD_CEPBUFEND)

Register	Offset	R/W	Description	Reset Value
HSUSBD_CEPBUFEND	HSUSBD_BA+0x058	R/W	Control Endpoint RAM End Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**EADDR**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EADDR							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**EADDR**	**Control-endpoint End Address** This is the end-address of the RAM space allocated for the control-endpoint.

DMA Control Status Register (HSUSBD_DMACTL)

Register	Offset	R/W	Description	Reset Value
HSUSBD_DMACTL	HSUSBD_BA+0x05C	R/W	DMA Control Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							SVINEP
7	6	5	4	3	2	1	0
DMARST	SGEN	DMAEN	DMARD	EPNUM			

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**SVINEP**	**Serve IN Endpoint** This bit is used to specify DMA serving endpoint-IN endpoint or OUT endpoint. 0 = DMA serves OUT endpoint. 1 = DMA serves IN endpoint.
[7]	**DMARST**	**Reset DMA State Machine** 0 = No reset the DMA state machine. 1 = Reset the DMA state machine.
[6]	**SGEN**	**Scatter Gather Function Enable Bit** 0 = Scatter gather function Disabled. 1 = Scatter gather function Enabled.
[5]	**DMAEN**	**DMA Enable Bit** 0 = DMA function Disabled. 1 = DMA function Enabled.
[4]	**DMARD**	**DMA Operation** 0 = The operation is a DMA write (read from USB buffer). DMA will check endpoint data available count (HSUSBD_EPxDATCNT) according to EPNUM setting before to perform DMA write operation. 1 = The operation is a DMA read (write to USB buffer).
[3:0]	**EPNUM**	**DMA Endpoint Address Bits** Used to define the Endpoint Address

DMA Count Register (HSUSBD_DMACNT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_DMACNT	HSUSBD_BA+0x060	R/W	DMA Count Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**DMACNT**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DMACNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMACNT							

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:0]	**DMACNT**	**DMA Transfer Count** The transfer count of the DMA operation to be performed is written to this register.

Endpoint A~L Data Register (HSUSBD_EPADAT~ HSUSBD_EPLDAT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPADAT	HSUSBD_BA+0x064	R/W	Endpoint A Data Register	0x0000_0000
HSUSBD_EPBDAT	HSUSBD_BA+0x08C	R/W	Endpoint B Data Register	0x0000_0000
HSUSBD_EPCDAT	HSUSBD_BA+0x0B4	R/W	Endpoint C Data Register	0x0000_0000
HSUSBD_EPDDAT	HSUSBD_BA+0x0DC	R/W	Endpoint D Data Register	0x0000_0000
HSUSBD_EPEDAT	HSUSBD_BA+0x104	R/W	Endpoint E Data Register	0x0000_0000
HSUSBD_EPFDAT	HSUSBD_BA+0x12C	R/W	Endpoint F Data Register	0x0000_0000
HSUSBD_EPGDAT	HSUSBD_BA+0x154	R/W	Endpoint G Data Register	0x0000_0000
HSUSBD_EPHDAT	HSUSBD_BA+0x17C	R/W	Endpoint H Data Register	0x0000_0000
HSUSBD_EPIDAT	HSUSBD_BA+0x1A4	R/W	Endpoint I Data Register	0x0000_0000
HSUSBD_EPJDAT	HSUSBD_BA+0x1CC	R/W	Endpoint J Data Register	0x0000_0000
HSUSBD_EPKDAT	HSUSBD_BA+0x1F4	R/W	Endpoint K Data Register	0x0000_0000
HSUSBD_EPLDAT	HSUSBD_BA+0x21C	R/W	Endpoint L Data Register	0x0000_0000

31	30	29	28	27	26	25	24
EPDAT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
EPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
EPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EPDAT							

Bits	Description	
[31:0]	**EPDAT**	**Endpoint A~L Data Register** Endpoint A~L data buffer for the buffer transaction (read or write). **Note:** Only word or byte access are supported.

Endpoint A~L Interrupt Status Register (HSUSBD_EPAINTSTS~ HSUSBD_EPLINTSTS)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPAINTSTS	HSUSBD_BA+0x068	R/W	Endpoint A Interrupt Status Register	0x0000_0003
HSUSBD_EPBINTSTS	HSUSBD_BA+0x090	R/W	Endpoint B Interrupt Status Register	0x0000_0003
HSUSBD_EPCINTSTS	HSUSBD_BA+0x0B8	R/W	Endpoint C Interrupt Status Register	0x0000_0003
HSUSBD_EPDINTSTS	HSUSBD_BA+0x0E0	R/W	Endpoint D Interrupt Status Register	0x0000_0003
HSUSBD_EPEINTSTS	HSUSBD_BA+0x108	R/W	Endpoint E Interrupt Status Register	0x0000_0003
HSUSBD_EPFINTSTS	HSUSBD_BA+0x130	R/W	Endpoint F Interrupt Status Register	0x0000_0003
HSUSBD_EPGINTSTS	HSUSBD_BA+0x158	R/W	Endpoint G Interrupt Status Register	0x0000_0003
HSUSBD_EPHINTSTS	HSUSBD_BA+0x180	R/W	Endpoint H Interrupt Status Register	0x0000_0003
HSUSBD_EPIINTSTS	HSUSBD_BA+0x1A8	R/W	Endpoint I Interrupt Status Register	0x0000_0003
HSUSBD_EPJINTSTS	HSUSBD_BA+0x1D0	R/W	Endpoint J Interrupt Status Register	0x0000_0003
HSUSBD_EPKINTSTS	HSUSBD_BA+0x1F8	R/W	Endpoint K Interrupt Status Register	0x0000_0003
HSUSBD_EPLINTSTS	HSUSBD_BA+0x220	R/W	Endpoint L Interrupt Status Register	0x0000_0003

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			SHORTRXIF	ERRIF	NYETIF	STALLIF	NAKIF
7	6	5	4	3	2	1	0
PINGIF	INTKIF	OUTTKIF	RXPKIF	TXPKIF	SHORTTXIF	BUFEMPTYIF	BUFFULLIF

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**SHORTRXIF**	**Bulk Out Short Packet Received Interrupt** 0 = No bulk out short packet is received. 1 = Received bulk out short packet (including zero length packet). **Note:** Write 1 to clear this bit to 0.
[11]	**ERRIF**	**ERR Sent Interrupt** 0 = No any error in the transaction. 1 = There occurs any error in the transaction. **Note:** Write 1 to clear this bit to 0.

[10]	**NYETIF**	**NYET Sent Interrupt** 0 = The space available in the RAM is sufficient to accommodate the next on coming data packet. 1 = The space available in the RAM is not sufficient to accommodate the next on coming data packet. **Note:** Write 1 to clear this bit to 0.
[9]	**STALLIF**	**USB STALL Sent Interrupt** 0 = The last USB packet could be accepted or provided because the endpoint was stalled, and was acknowledged with a STALL. 1 = The last USB packet could not be accepted or provided because the endpoint was stalled, and was acknowledged with a STALL. **Note:** Write 1 to clear this bit to 0.
[8]	**NAKIF**	**USB NAK Sent Interrupt** 0 = The last USB IN packet could be provided, and was acknowledged with an ACK. 1 = The last USB IN packet could not be provided, and was acknowledged with a NAK. **Note:** Write 1 to clear this bit to 0.
[7]	**PINGIF**	**PING Token Interrupt** 0 = A Data PING token has not been received from the host. 1 = A Data PING token has been received from the host. **Note:** Write 1 to clear this bit to 0.
[6]	**INTKIF**	**Data IN Token Interrupt** 0 = Not Data IN token has been received from the host. 1 = A Data IN token has been received from the host. **Note:** Write 1 to clear this bit to 0.
[5]	**OUTTKIF**	**Data OUT Token Interrupt** 0 = A Data OUT token has not been received from the host. 1 = A Data OUT token has been received from the host. This bit also set by PING token (in high-speed only). **Note:** Write 1 to clear this bit to 0.
[4]	**RXPKIF**	**Data Packet Received Interrupt** 0 = No data packet is received from the host by the endpoint. 1 = A data packet is received from the host by the endpoint. **Note:** Write 1 to clear this bit to 0.
[3]	**TXPKIF**	**Data Packet Transmitted Interrupt** 0 = Not a data packet is transmitted from the endpoint to the host. 1 = A data packet is transmitted from the endpoint to the host. **Note:** Write 1 to clear this bit to 0.
[2]	**SHORTTXIF**	**Short Packet Transferred Interrupt** 0 = The length of the last packet was not less than the Maximum Packet Size (EPMPS). 1 = The length of the last packet was less than the Maximum Packet Size (EPMPS). **Note:** Write 1 to clear this bit to 0.

[1]	**BUFEMPTYIF**	**Buffer Empty Interrupt** For an IN endpoint, a buffer is available to the local side for writing up to FIFO full of bytes. 0 = The endpoint buffer is not empty. 1 = The endpoint buffer is empty. For an OUT endpoint: 0 = The currently selected buffer has not a count of 0. 1 = The currently selected buffer has a count of 0, or no buffer is available on the local side (nothing to read). **Note:** This bit is read-only.
[0]	**BUFFULLIF**	**Buffer Full Interrupt** For an IN endpoint, the currently selected buffer is full, or no buffer is available to the local side for writing (no space to write). For an OUT endpoint, there is a buffer available on the local side, and there are FIFO full of bytes available to be read (entire packet is available for reading). 0 = The endpoint packet buffer is not full. 1 = The endpoint packet buffer is full. **Note:** This bit is read-only.

Endpoint A~L Interrupt Enable Control Register (HSUSBD_EPAINTEN~ HSUSBD_EPLINTEN)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPAINTEN	HSUSBD_BA+0x06C	R/W	Endpoint A Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPBINTEN	HSUSBD_BA+0x094	R/W	Endpoint B Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPCINTEN	HSUSBD_BA+0x0BC	R/W	Endpoint C Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPDINTEN	HSUSBD_BA+0x0E4	R/W	Endpoint D Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPEINTEN	HSUSBD_BA+0x10C	R/W	Endpoint E Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPFINTEN	HSUSBD_BA+0x134	R/W	Endpoint F Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPGINTEN	HSUSBD_BA+0x15C	R/W	Endpoint G Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPHINTEN	HSUSBD_BA+0x184	R/W	Endpoint H Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPIINTEN	HSUSBD_BA+0x1AC	R/W	Endpoint I Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPJINTEN	HSUSBD_BA+0x1D4	R/W	Endpoint J Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPKINTEN	HSUSBD_BA+0x1FC	R/W	Endpoint K Interrupt Enable Control Register	0x0000_0000
HSUSBD_EPLINTEN	HSUSBD_BA+0x224	R/W	Endpoint L Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			SHORTRXIEN	ERRIEN	NYETIEN	STALLIEN	NAKIEN
7	6	5	4	3	2	1	0
PINGIEN	INTKIEN	OUTTKIEN	RXPKIEN	TXPKIEN	SHORTTXIEN	BUFEMPTYIEN	BUFFULLIEN

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**SHORTRXIEN**	**Bulk Out Short Packet Received Interrupt Enable Bit** When set, this bit enables a local interrupt to be set whenever bulk out short packet occurs on the bus for this endpoint. 0 = Bulk out interrupt Disabled. 1 = Bulk out interrupt Enabled.
[11]	**ERRIEN**	**ERR Sent Interrupt Enable Bit** When set, this bit enables a local interrupt to be set whenever ERR condition occurs on the bus for this endpoint. 0 = Error event interrupt Disabled. 1 = Error event interrupt Enabled.

[10]	**NYETIEN**	**NYET Sent Interrupt Enable Bit** When set, this bit enables a local interrupt to be set whenever NYET condition occurs on the bus for this endpoint. 0 = NYET condition interrupt Disabled. 1 = NYET condition interrupt Enabled.
[9]	**STALLIEN**	**USB STALL Sent Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a stall token is sent to the host. 0 = STALL token interrupt Disabled. 1 = STALL token interrupt Enabled.
[8]	**NAKIEN**	**USB NAK Sent Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a NAK token is sent to the host. 0 = NAK token interrupt Disabled. 1 = NAK token interrupt Enabled.
[7]	**PINGIEN**	**PING Token Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a PING token has been received from the host. 0 = PING token interrupt Disabled. 1 = PING token interrupt Enabled.
[6]	**INTKIEN**	**Data IN Token Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a Data IN token has been received from the host. 0 = Data IN token interrupt Disabled. 1 = Data IN token interrupt Enabled.
[5]	**OUTTKIEN**	**Data OUT Token Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a Data OUT token has been received from the host. 0 = Data OUT token interrupt Disabled. 1 = Data OUT token interrupt Enabled.
[4]	**RXPKIEN**	**Data Packet Received Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a data packet has been received from the host. 0 = Data packet has been received from the host interrupt Disabled. 1 = Data packet has been received from the host interrupt Enabled.
[3]	**TXPKIEN**	**Data Packet Transmitted Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a data packet has been transmitted to the host. 0 = Data packet has been transmitted to the host interrupt Disabled. 1 = Data packet has been transmitted to the host interrupt Enabled.
[2]	**SHORTTXIEN**	**Short Packet Transferred Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a short data packet has been transferred to/from the host. 0 = Short data packet interrupt Disabled. 1 = Short data packet interrupt Enabled.
[1]	**BUFEMPTYIEN**	**Buffer Empty Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a buffer empty condition is detected on the bus. 0 = Buffer empty interrupt Disabled. 1 = Buffer empty interrupt Enabled.

[0]	**BUFFULLIEN**	**Buffer Full Interrupt Enable Bit** When set, this bit enables a local interrupt to be set when a buffer full condition is detected on the bus. 0 = Buffer full interrupt Disabled. 1 = Buffer full interrupt Enabled.

Endpoint A~L Data Available Count Register (HSUSBD_EPADATCNT~ HSUSBD_EPLDATCNT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPADATCNT	HSUSBD_BA+0x070	R	Endpoint A Data Available Count Register	0x0000_0000
HSUSBD_EPBDATCNT	HSUSBD_BA+0x098	R	Endpoint B Data Available Count Register	0x0000_0000
HSUSBD_EPCDATCNT	HSUSBD_BA+0x0C0	R	Endpoint C Data Available Count Register	0x0000_0000
HSUSBD_EPDDATCNT	HSUSBD_BA+0x0E8	R	Endpoint D Data Available Count Register	0x0000_0000
HSUSBD_EPEDATCNT	HSUSBD_BA+0x110	R	Endpoint E Data Available Count Register	0x0000_0000
HSUSBD_EPFDATCNT	HSUSBD_BA+0x138	R	Endpoint F Data Available Count Register	0x0000_0000
HSUSBD_EPGDATCNT	HSUSBD_BA+0x160	R	Endpoint G Data Available Count Register	0x0000_0000
HSUSBD_EPHDATCNT	HSUSBD_BA+0x188	R	Endpoint H Data Available Count Register	0x0000_0000
HSUSBD_EPIDATCNT	HSUSBD_BA+0x1B0	R	Endpoint I Data Available Count Register	0x0000_0000
HSUSBD_EPJDATCNT	HSUSBD_BA+0x1D8	R	Endpoint J Data Available Count Register	0x0000_0000
HSUSBD_EPKDATCNT	HSUSBD_BA+0x200	R	Endpoint K Data Available Count Register	0x0000_0000
HSUSBD_EPLDATCNT	HSUSBD_BA+0x228	R	Endpoint L Data Available Count Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	DMALOOP						
23	22	21	20	19	18	17	16
DMALOOP							
15	14	13	12	11	10	9	8
DATCNT							
7	6	5	4	3	2	1	0
DATCNT							

Bits	Description	
[31]	**Reserved**	Reserved.
[30:16]	**DMALOOP**	**DMA Loop** This register is the remaining DMA loop to complete. Each loop means 32-byte transfer.
[15:0]	**DATCNT**	**Data Count** For an IN endpoint (EPDIR (HSUSBD_EPxCFG[3] is high.), this register returns the number of valid bytes in the IN endpoint packet buffer. For an OUT endpoint (EPDIR (HSUSBD_EPxCFG[3] is low.), this register returns the number of received valid bytes in the Host OUT transfer.

Endpoint A~L Response Control Register (HSUSBD_EPARSPCTL~ HSUSBD_EPLRSPCTL)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPARSPCTL	HSUSBD_BA+0x074	R/W	Endpoint A Response Control Register	0x0000_0000
HSUSBD_EPBRSPCTL	HSUSBD_BA+0x09C	R/W	Endpoint B Response Control Register	0x0000_0000
HSUSBD_EPCRSPCTL	HSUSBD_BA+0x0C4	R/W	Endpoint C Response Control Register	0x0000_0000
HSUSBD_EPDRSPCTL	HSUSBD_BA+0x0EC	R/W	Endpoint D Response Control Register	0x0000_0000
HSUSBD_EPERSPCTL	HSUSBD_BA+0x114	R/W	Endpoint E Response Control Register	0x0000_0000
HSUSBD_EPFRSPCTL	HSUSBD_BA+0x13C	R/W	Endpoint F Response Control Register	0x0000_0000
HSUSBD_EPGRSPCTL	HSUSBD_BA+0x164	R/W	Endpoint G Response Control Register	0x0000_0000
HSUSBD_EPHRSPCTL	HSUSBD_BA+0x18C	R/W	Endpoint H Response Control Register	0x0000_0000
HSUSBD_EPIRSPCTL	HSUSBD_BA+0x1B4	R/W	Endpoint I Response Control Register	0x0000_0000
HSUSBD_EPJRSPCTL	HSUSBD_BA+0x1DC	R/W	Endpoint J Response Control Register	0x0000_0000
HSUSBD_EPKRSPCTL	HSUSBD_BA+0x204	R/W	Endpoint K Response Control Register	0x0000_0000
HSUSBD_EPLRSPCTL	HSUSBD_BA+0x22C	R/W	Endpoint L Response Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DISBUF	**SHORTTXEN**	**ZEROLEN**	**HALT**	**TOGGLE**	**MODE**		**FLUSH**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**DISBUF**	**Buffer Disable Bit** This bit is used to receive unknown size OUT short packet. The received packet size is reference HSUSBD_EPxDATCNT register. 0 = Buffer not disabled when Bulk-OUT short packet is received. 1 = Buffer disabled when Bulk-OUT short packet is received.

[6]	**SHORTTXEN**	**Short Packet Transfer Enable** This bit is applicable only in case of Auto-Validate Method. This bit is set to validate any remaining data in the buffer which is not equal to the MPS of the endpoint, and happens to be the last transfer. This bit gets cleared once the data packet is sent. 0 = Not validate any remaining data in the buffer which is not equal to the MPS of the endpoint. 1 = Validate any remaining data in the buffer which is not equal to the MPS of the endpoint.
[5]	**ZEROLEN**	**Zero Length** This bit is used to send a zero-length packet response to an IN-token. When this bit is set, a zero packet is sent to the host on reception of an IN-token. This bit gets cleared once the zero length data packet is sent. 0 = A zero packet is not sent to the host on reception of an IN-token. 1 = A zero packet is sent to the host on reception of an IN-token.
[4]	**HALT**	**Endpoint Halt** This bit is used to send a STALL handshake as response to the token from the host. When an Endpoint Set Feature (ep_halt) is detected by the local CPU, it must write a '1' to this bit. 0 = Not send a STALL handshake as response to the token from the host. 1 = Send a STALL handshake as response to the token from the host.
[3]	**TOGGLE**	**Endpoint Toggle** This bit is used to clear the endpoint data toggle bit. Reading this bit returns the current state of the endpoint data toggle bit. The local CPU may use this bit to initialize the end-point's toggle in case of reception of a Set Interface request or a Clear Feature (ep_halt) request from the host. Only when toggle bit is "1", this bit can be written into the inversed write data bit[3]. 0 = Not clear the endpoint data toggle bit. 1 = Clear the endpoint data toggle bit.
[2:1]	**MODE**	**Mode Control** The two bits decide the operation mode of the in-endpoint. 00 = Auto-Validate Mode. 01 = Manual-Validate Mode. 10 = Fly Mode. 11 = Reserved. These bits are not valid for an out-endpoint. The auto validate mode will be activated when the reserved mode is selected.
[0]	**FLUSH**	**Buffer Flush** Writing 1 to this bit causes the packet buffer to be flushed and the corresponding EP_AVAIL register to be cleared. This bit is self-clearing. This bit should always be written after a configuration event. 0 = The packet buffer is not flushed. 1 = The packet buffer is flushed by user.

Endpoint A~L Maximum Packet Size Register (HSUSBD_EPAMPS~ HSUSBD_EPLMPS)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPAMPS	HSUSBD_BA+0x078	R/W	Endpoint A Maximum Packet Size Register	0x0000_0000
HSUSBD_EPBMPS	HSUSBD_BA+0x0A0	R/W	Endpoint B Maximum Packet Size Register	0x0000_0000
HSUSBD_EPCMPS	HSUSBD_BA+0x0C8	R/W	Endpoint C Maximum Packet Size Register	0x0000_0000
HSUSBD_EPDMPS	HSUSBD_BA+0x0F0	R/W	Endpoint D Maximum Packet Size Register	0x0000_0000
HSUSBD_EPEMPS	HSUSBD_BA+0x118	R/W	Endpoint E Maximum Packet Size Register	0x0000_0000
HSUSBD_EPFMPS	HSUSBD_BA+0x140	R/W	Endpoint F Maximum Packet Size Register	0x0000_0000
HSUSBD_EPGMPS	HSUSBD_BA+0x168	R/W	Endpoint G Maximum Packet Size Register	0x0000_0000
HSUSBD_EPHMPS	HSUSBD_BA+0x190	R/W	Endpoint H Maximum Packet Size Register	0x0000_0000
HSUSBD_EPIMPS	HSUSBD_BA+0x1B8	R/W	Endpoint I Maximum Packet Size Register	0x0000_0000
HSUSBD_EPJMPS	HSUSBD_BA+0x1E0	R/W	Endpoint J Maximum Packet Size Register	0x0000_0000
HSUSBD_EPKMPS	HSUSBD_BA+0x208	R/W	Endpoint K Maximum Packet Size Register	0x0000_0000
HSUSBD_EPLMPS	HSUSBD_BA+0x230	R/W	Endpoint L Maximum Packet Size Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**EPMPS**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EPMPS							

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:0]	**EPMPS**	**Endpoint Maximum Packet Size** This field determines the Maximum Packet Size of the Endpoint.

Endpoint A~L Transfer Count Register (HSUSBD_EPATXCNT~ HSUSBD_EPLTXCNT)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPATXCNT	HSUSBD_BA+0x07C	R/W	Endpoint A Transfer Count Register	0x0000_0000
HSUSBD_EPBTXCNT	HSUSBD_BA+0x0A4	R/W	Endpoint B Transfer Count Register	0x0000_0000
HSUSBD_EPCTXCNT	HSUSBD_BA+0x0CC	R/W	Endpoint C Transfer Count Register	0x0000_0000
HSUSBD_EPDTXCNT	HSUSBD_BA+0x0F4	R/W	Endpoint D Transfer Count Register	0x0000_0000
HSUSBD_EPETXCNT	HSUSBD_BA+0x11C	R/W	Endpoint E Transfer Count Register	0x0000_0000
HSUSBD_EPFTXCNT	HSUSBD_BA+0x144	R/W	Endpoint F Transfer Count Register	0x0000_0000
HSUSBD_EPGTXCNT	HSUSBD_BA+0x16C	R/W	Endpoint G Transfer Count Register	0x0000_0000
HSUSBD_EPHTXCNT	HSUSBD_BA+0x194	R/W	Endpoint H Transfer Count Register	0x0000_0000
HSUSBD_EPITXCNT	HSUSBD_BA+0x1BC	R/W	Endpoint I Transfer Count Register	0x0000_0000
HSUSBD_EPJTXCNT	HSUSBD_BA+0x1E4	R/W	Endpoint J Transfer Count Register	0x0000_0000
HSUSBD_EPKTXCNT	HSUSBD_BA+0x20C	R/W	Endpoint K Transfer Count Register	0x0000_0000
HSUSBD_EPLTXCNT	HSUSBD_BA+0x234	R/W	Endpoint L Transfer Count Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**TXCNT**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXCNT							

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:0]	**TXCNT**	**Endpoint Transfer Count** For IN endpoints, this field determines the total number of bytes to be sent to the host in case of manual validation method. For OUT endpoints, this field has no effect.

Endpoint A~L Configuration Register (HSUSBD_EPACFG~ HSUSBD_EPLCFG)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPACFG	HSUSBD_BA+0x080	R/W	Endpoint A Configuration Register	0x0000_0012
HSUSBD_EPBCFG	HSUSBD_BA+0x0A8	R/W	Endpoint B Configuration Register	0x0000_0022
HSUSBD_EPCCFG	HSUSBD_BA+0x0D0	R/W	Endpoint C Configuration Register	0x0000_0032
HSUSBD_EPDCFG	HSUSBD_BA+0x0F8	R/W	Endpoint D Configuration Register	0x0000_0042
HSUSBD_EPECFG	HSUSBD_BA+0x120	R/W	Endpoint E Configuration Register	0x0000_0052
HSUSBD_EPFCFG	HSUSBD_BA+0x148	R/W	Endpoint F Configuration Register	0x0000_0062
HSUSBD_EPGCFG	HSUSBD_BA+0x170	R/W	Endpoint G Configuration Register	0x0000_0072
HSUSBD_EPHCFG	HSUSBD_BA+0x198	R/W	Endpoint H Configuration Register	0x0000_0082
HSUSBD_EPICFG	HSUSBD_BA+0x1C0	R/W	Endpoint I Configuration Register	0x0000_0092
HSUSBD_EPJCFG	HSUSBD_BA+0x1E8	R/W	Endpoint J Configuration Register	0x0000_00A2
HSUSBD_EPKCFG	HSUSBD_BA+0x210	R/W	Endpoint K Configuration Register	0x0000_00B2
HSUSBD_EPLCFG	HSUSBD_BA+0x238	R/W	Endpoint L Configuration Register	0x0000_00C2

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EPNUM				**EPDIR**	**EPTYPE**		**EPEN**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[7:4]	**EPNUM**	**Endpoint Number** This field selects the number of the endpoint. Valid numbers 1 to 15.
[3]	**EPDIR**	**Endpoint Direction** 0 = out-endpoint (Host OUT to Device). 1 = in-endpoint (Host IN to Device). **Note:** A maximum of one OUT and IN endpoint is allowed for each endpoint number.

[2:1]	**EPTYPE**	**Endpoint Type** This field selects the type of this endpoint. Endpoint 0 is forced to a Control type. 00 = Reserved. 01 = Bulk. 10 = Interrupt. 11 = Isochronous.
[0]	**EPEN**	**Endpoint Valid** When set, this bit enables this endpoint. This bit has no effect on Endpoint 0, which is always enabled. 0 = The endpoint Disabled. 1 = The endpoint Enabled.

Endpoint A~L RAM Start Address Register (HSUSBD_EPABUFSTART~ HSUSBD_EPLBUFSTART)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPABUFSTART	HSUSBD_BA+0x084	R/W	Endpoint A RAM Start Address Register	0x0000_0000
HSUSBD_EPBBUFSTART	HSUSBD_BA+0x0AC	R/W	Endpoint B RAM Start Address Register	0x0000_0000
HSUSBD_EPCBUFSTART	HSUSBD_BA+0x0D4	R/W	Endpoint C RAM Start Address Register	0x0000_0000
HSUSBD_EPDBUFSTART	HSUSBD_BA+0x0FC	R/W	Endpoint D RAM Start Address Register	0x0000_0000
HSUSBD_EPEBUFSTART	HSUSBD_BA+0x124	R/W	Endpoint E RAM Start Address Register	0x0000_0000
HSUSBD_EPFBUFSTART	HSUSBD_BA+0x14C	R/W	Endpoint F RAM Start Address Register	0x0000_0000
HSUSBD_EPGBUFSTART	HSUSBD_BA+0x174	R/W	Endpoint G RAM Start Address Register	0x0000_0000
HSUSBD_EPHBUFSTART	HSUSBD_BA+0x19C	R/W	Endpoint H RAM Start Address Register	0x0000_0000
HSUSBD_EPIBUFSTART	HSUSBD_BA+0x1C4	R/W	Endpoint I RAM Start Address Register	0x0000_0000
HSUSBD_EPJBUFSTART	HSUSBD_BA+0x1EC	R/W	Endpoint J RAM Start Address Register	0x0000_0000
HSUSBD_EPKBUFSTART	HSUSBD_BA+0x214	R/W	Endpoint K RAM Start Address Register	0x0000_0000
HSUSBD_EPLBUFSTART	HSUSBD_BA+0x23C	R/W	Endpoint L RAM Start Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			SADDR				
7	6	5	4	3	2	1	0
SADDR							

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12:0]	**SADDR**	**Endpoint Start Address** This is the start-address of the RAM space allocated for the endpoint A~L.

Endpoint A~L RAM End Address Register (HSUSBD_EPABUFEND~ HSUSBD_EPLBUFEND)

Register	Offset	R/W	Description	Reset Value
HSUSBD_EPABUFEND	HSUSBD_BA+0x088	R/W	Endpoint A RAM End Address Register	0x0000_0000
HSUSBD_EPBBUFEND	HSUSBD_BA+0x0B0	R/W	Endpoint B RAM End Address Register	0x0000_0000
HSUSBD_EPCBUFEND	HSUSBD_BA+0x0D8	R/W	Endpoint C RAM End Address Register	0x0000_0000
HSUSBD_EPDBUFEND	HSUSBD_BA+0x100	R/W	Endpoint D RAM End Address Register	0x0000_0000
HSUSBD_EPEBUFEND	HSUSBD_BA+0x128	R/W	Endpoint E RAM End Address Register	0x0000_0000
HSUSBD_EPFBUFEND	HSUSBD_BA+0x150	R/W	Endpoint F RAM End Address Register	0x0000_0000
HSUSBD_EPGBUFEND	HSUSBD_BA+0x178	R/W	Endpoint G RAM End Address Register	0x0000_0000
HSUSBD_EPHBUFEND	HSUSBD_BA+0x1A0	R/W	Endpoint H RAM End Address Register	0x0000_0000
HSUSBD_EPIBUFEND	HSUSBD_BA+0x1C8	R/W	Endpoint I RAM End Address Register	0x0000_0000
HSUSBD_EPJBUFEND	HSUSBD_BA+0x1F0	R/W	Endpoint J RAM End Address Register	0x0000_0000
HSUSBD_EPKBUFEND	HSUSBD_BA+0x218	R/W	Endpoint K RAM End Address Register	0x0000_0000
HSUSBD_EPLBUFEND	HSUSBD_BA+0x240	R/W	Endpoint L RAM End Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**EADDR**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EADDR							

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12:0]	**EADDR**	**Endpoint End Address** This is the end-address of the RAM space allocated for the endpoint A~L.

USB Battery Charge Detect Control Register (HSUSBD_BCDC)

Register	Offset	R/W	Description	Reset Value
HSUSBD_BCDC	HSUSBD_BA+0x6F8	R/W	Battery Charge Detect Control Register	0x0000_0000

31	30	29	28	27	26	25	24
BCDIF	BCDIEN	Reserved					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		USP	DETSTS	DETMOD			BCDEN

Bits	Description	
[31]	**BCDIF**	**Battery Charge Detect Interrupt Status** It supports VBUSOK and DCD interrupt status When HSUSBD_BCDC[0] = 1, HSUSBD_BCDC[3:1] = 001, V_{BUS} is detected. When HSUSBD_BCDC[0] = 1, HSUSBD_BCDC[3:1] = 010, DCD is detected. 0 = BCD event did not occur. 1 = BCD event occurred. **Note:** Write 1 to clear this bit to 0.
[30]	**BCDIEN**	**Battery Charge Detect Interrupt Enable Bit** 0 = BCD Interrupt Disabled. 1 = BCD Interrupt Enabled.
[29:6]	**Reserved**	Reserved.
[5]	**USP**	**USB Special Port (Read Only)** When DETMOD = 011(PD), detect DM be pulled logic high, it means contact port not USB support port. 0 = USB support port. 1 = Special port. (PS/2 or proprietary charger)
[4]	**DETSTS**	**Detect Status (Read Only)** When DETMOD = 000 (IDLE). DETSTS = 0. When DETMOD = 001 (V_{BUS} detect). 0 = V_{BUS} is less than threshold voltage. 1 = V_{BUS} is greater than threshold voltage. When DETMOD = 010 (DCD detect). 0 = Data pin not contacted. 1 = Data pin contacted. When DETMOD = 011 (PD). 0 = SDP port or not USB support port. If it is not USB support port, USP is 1. 1 = DCP or CDP.

		When DETMOD = 100 (SD). 0 = CDP. 1 = DCP.
[3:1]	**DETMOD**	**Detect Mode** When BCDEN = 1, select detect mode to perform. 000 = Idle, nothing to detect. 001 = V_{BUS} detect, detect USB V_{BUS} whether great than threshold voltage. 010 = Data contact detect (DCD), detect data pin contact status. 011 = Primary detect (PD), distinguish between (SDP or USP) and (CDP or DCP). 100 = Secondary detect (SD), distinguish between CDP and DCP. 101~111 = Reserved.
[0]	**BCDEN**	**Battery Charge Detect Enable** Enable battery charge detect, select DETMOD and then observer DETSTS to decide contact port. PHY can be used for BCD, but cannot be used for communication when BCDEN = 1. 0 = Normal operation. 1 = Battery charge detect operation.

LPM Control and Status Register (HSUSBD_LPMCSR)

Register	Offset	R/W	Description	Reset Value
HSUSBD_LPMCSR	HSUSBD_BA+0x6FC	R/W	LPM Control and Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	LPMSENDNYET	LPMSLEEPEN	LPMEN	Reserved	bmAttributes		
7	6	5	4	3	2	1	0
bmAttributes							

Bits	Description	
[31:15]	**Reserved**	Reserved.
[14]	**LPMSENDNYET**	**LPM Send NYET Response** 0 = Not send a NYET handshake as response to the LPM token. 1 = Send a NYET handshake as response to the LPM token.
[13]	**LPMSLEEPEN**	**LPM Sleep Function Enable Bit** 0 = The LPM sleep function Disabled. 1 = The LPM sleep function Enabled.
[12]	**LPMEN**	**LPM Function Enable Bit** 0 = The LPM function Disabled. 1 = The LPM function Enabled.
[11]	**Reserved**	Reserved.
[10:0]	**bmAttributes**	**The LPM Token bmAttributes Field Definition (Read Only)** Bit 10-9 (Reserved) Bit 8 (bRemoteWake) Bit 7-4 (HIRD) Bit 3-0 (bLinkState): 0000 = Reser..ved 0001 = L1 (Sle..ep) 0010 – 1111 = Reser..ved

AHB DMA Address Register (HSUSBD_DMAADDR)

Register	Offset	R/W	Description	Reset Value
HSUSBD_DMAADDR	HSUSBD_BA+0x700	R/W	AHB DMA Address Register	0x0000_0000

31	30	29	28	27	26	25	24
DMAADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DMAADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DMAADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMAADDR							

Bits	Description	
[31:0]	**DMAADDR**	**DMAADDR** The register specifies the address from which the DMA has to read / write. The address must WORD (32-bit) aligned.

USB PHY Control Register (HSUSBD_PHYCTL)

Register	Offset	R/W	Description	Reset Value
HSUSBD_PHYCTL	HSUSBD_BA+0x704	R/W	USB PHY Control Register	0x0000_0000

31	30	29	28	27	26	25	24
VBUSDET	Reserved			PHYCLKSTB	STALLREVERT	LINESTATEWKEN	VBUSWKEN
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						PHYEN	DPPUEN
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31]	**VBUSDET**	**V_{BUS} Status** 0 = The V_{BUS} is not detected yet. 1 = The V_{BUS} is detected.
[30:28]	**Reserved**	Reserved.
[27]	**PHYCLKSTB**	**PHY Clock Stable Flag** 0 = The PHY clock is not stable for USB device usage. 1 = The PHY clock is stable for USB device usage.
[26]	**STALLREVERT**	**Stall Revert Write Pointer Enable Bit** 0 = The Stall revert write pointer function Disabled. 1 = The Stall revert write pointer function Enabled.
[25]	**LINESTATEWKEN**	**Line State Wake-up Enable Bit** 0 = The Line State wake-up function Disabled. 1 = The Line State wake-up function Enabled.
[24]	**VBUSWKEN**	**V_{BUS} Wake-up Enable Bit** 0 = The wake-up function Disabled. 1 = The wake-up function Enabled.
[23:10]	**Reserved**	Reserved.
[9]	**PHYEN**	**PHY Suspend Enable Bit** 0 = The USB PHY is suspend. 1 = The USB PHY is not suspend.
[8]	**DPPUEN**	**DP Pull-up Enable Bit** 0 = Pull-up resistor on D+ Disabled. 1 = Pull-up resistor on D+ Enabled.
[7:0]	**Reserved**	Reserved.

6.35 High Speed USB 2.0 and USB 1.1 Host Controller (HSUSB and USBH)

6.35.1 Overview

This chip is equipped with a USB 2.0 HS/FS Host Controller (USBH) that supports Enhanced Host Controller Interface (EHCI) and Open Host Controller Interface (OpenHCI, OHCI) Specification, a register-level description of a host controller, to manage the devices and data transfer of Universal Serial Bus (USB).

The USBH supports an integrated Root Hub with a USB port, a DMA for real-time data transfer between system memory and USB bus, port power control and port overcurrent detection.

The USBH is responsible for detecting the connect and disconnect of USB devices, managing data transfer, collecting status and activity of USB bus, providing power control and detecting overcurrent of attached USB devices.

6.35.2 Features

- Compliant with Universal Serial Bus (USB) Specification Revision 2.0.
- Supports Enhanced Host Controller Interface (EHCI) Specification Revision 1.0
- Supports Open Host Controller Interface (OpenHCI) Specification Revision 1.0.
- Supports high-speed (480Mbps), full-speed (12Mbps) and low-speed (1.5Mbps) USB devices.
- Supports Control, Bulk, Interrupt, Isochronous and Split transfers.
- Supports an integrated Root Hub.
- Supports a port routing logic to route full/low speed device to OHCI controller.
- Supports two USB host port shared with USB device (OTG function).
- Supports port power control and port overcurrent detection.
- Supports DMA for real-time data transfer.

6.35.3 Block Diagram

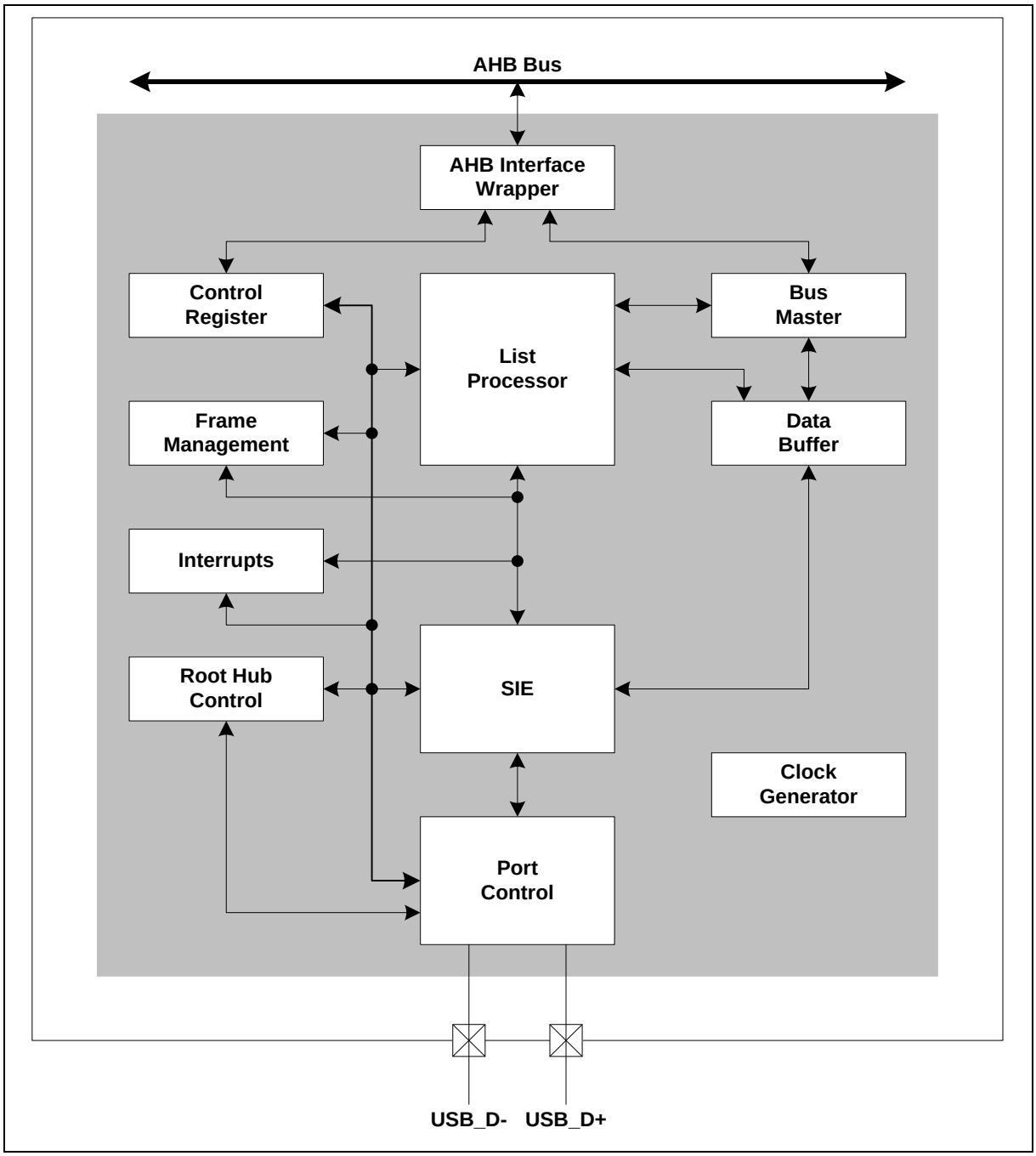


Figure 6.35-1 USB 2.0 HS/FS Host Controller Block Diagram

6.35.4 Basic Configuration

6.35.4.1 USBH EHCI Controller Basic Configuration

- Clock source Configuration
  - Select the source of USBH engine clock on USBSEL (CLK_CLKSEL0[8]).
  - Select the clock divider number of USBH engine clock on USBDIV (CLK_CLKDIV0[7:4]).
  - Enable USBH engine clock in USBHCKEN (CLK_AHBCLK0[16]).
- Reset Configuration
  - Reset USBH controller in HSUSBHRST (SYS_IPRST0[16]).

6.35.4.2 USBH OHCI Controller Basic Configuration

- Clock source Configuration
  - Select the source of USBH engine clock on USBSEL (CLK_CLKSEL0[8]).
  - Select the clock divider number of USBH engine clock on USBDIV (CLK_CLKDIV0[7:4]).
  - Enable USBH engine clock in USBHCKEN (CLK_AHBCLK0[16]).
- Reset Configuration
  - Reset USBH controller in HSUSBHRST (SYS_IPRST0[16]).

6.35.5 Functional Description

6.35.5.1 EHCI Controller

The EHCI is interfaced with the system through AHB interface. Whenever the CPU wants to initiate a register read or register write, it uses the AHB slave I/F signals and performs the necessary operation (register read writes). The CPU acts as a bus master, having initiated this transfer. At that time, EHCI acts as a target and responds to the transfer initiated by the system software. For example, if the CPU wants to write into one of the memory mapped registers of EHCI, it says the address and value to be written into that addressed register. EHCI targets the register by using that address and fills the register with the value specified by the software. If it is a register read, EHCI gets the value from the addressed register and puts it on the system bus.
Likewise, when the EHCI wants to perform a data transfer, it acts as a master and initiates a data transfer. At that time, the system memory acts as a bus target. EHCI, as a master can perform two types of data transfers, from EHCI to the system memory and from system memory to the EHCI. When the EHCI wants the data to be moved from the downstream USB2.0 device to the system memory, it initiates a memory write transfer by accessing the memory interfacing signals. EHCI writes the control word (write), data and data count to be moved to the system memory. The memory controller accepts the data and moves it to the memory. If the data has to be moved from memory to the downstream device, the EHCI issue a read transfer to system bus. The memory controller gives data through the memory interfacing signals. EHCI accepts the data and moves them to the downstream device.

To ensure the successful USB transfer and the correct data transmission between USBH engine clock domain and UTMI clock domain, the frequency of USBH engine clock should be higher than UTMI clock (30 MHz).

6.35.5.2 OHCI Controller

AHB Interface

The OpenHCI Host Controller is connected to the system by the AHB bus. The design requires both master and slave bus operations. As a master, the Host Controller is responsible for running cycles on the AHB bus to access EDs and TDs as well as transferring data between memory and the local data buffer. As a slave, the Host Controller monitors the cycles on the AHB bus and determines when to

respond to these cycles. Configuration and non-real-time control access to the Host Controller operational registers are through the AHB bus slave interface.

Host Controller

The host controller includes 5 functional blocks, including List Processing, Frame Management, Interrupt Processing, Host Controller Bus Master and Data Buffer.

The List Processor manages the data structures from the Host Controller Driver and coordinates all activity within the Host Controller.

The Frame Management is responsible for managing the frame specific tasks required by the USB specification and the OpenHCI specification. These tasks are:

- Management of the OpenHCI frame specific Operational Registers
- Operation of the Largest Data Packet Counter.
- Performing frame qualifications on USB Transaction requests to the SI.E.
- Generate SOF token requests to the SI.E.

Interrupts are the communication method for HC-initiated communication with the Host Controller Driver. There are several events that may trigger an interrupt from the Host Controller. Each specific event sets a specific bit in the HcInterruptStatus register.

The Host Controller Bus Master is the central block in the data path. The Host Controller Bus Master coordinates all access to the AHB Interface. There are two sources of bus mastering within Host Controller: the List Processor and the Data Buffer Engine.

The Data Buffer serves as the data interface between the Bus Master and the SI.E. It is a combination of a 64-byte latched based bi-directional asynchronous FIFO and a single DWORD AHB Holding Register.

USB Interface

The USB interface includes the integrated Root Hub with an USB port, Port 1 as well as the Serial Interface Engine (SIE) and USB clock generator. The interface combines responsibility for executing bus transactions requested by the HC as well as the hub and port management specified by USB.

The SIE is responsible for managing all transactions to the USB. It controls the bus protocol, packet generation/extraction, data parallel-to-serial conversion, CRC coding, bit stuffing, and NRZI encoding. All transactions on the USB are requested from the List Processor and Frame Manager.

The Root Hub is a collection of ports that are individually controlled and a hub that maintains control/status over functions common to all ports.

6.35.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
USBH Base Address: **USBH_BA = 0x4000_9000** **HSUSBH_BA = 0x4001_A000**				
HcRevision	USBH_BA+0x000	R	Host Controller Revision Register	0x0000_0110
HcControl	USBH_BA+0x004	R/W	Host Controller Control Register	0x0000_0000
HcCommandStatus	USBH_BA+0x008	R/W	Host Controller Command Status Register	0x0000_0000
HcInterruptStatus	USBH_BA+0x00C	R/W	Host Controller Interrupt Status Register	0x0000_0000
HcInterruptEnable	USBH_BA+0x010	R/W	Host Controller Interrupt Enable Register	0x0000_0000
HcInterruptDisable	USBH_BA+0x014	R/W	Host Controller Interrupt Disable Register	0x0000_0000
HcHCCA	USBH_BA+0x018	R/W	Host Controller Communication Area Register	0x0000_0000
HcPeriodCurrentED	USBH_BA+0x01C	R	Host Controller Period Current ED Register	0x0000_0000
HcControlHeadED	USBH_BA+0x020	R/W	Host Controller Control Head ED Register	0x0000_0000
HcControlCurrentED	USBH_BA+0x024	R/W	Host Controller Control Current ED Register	0x0000_0000
HcBulkHeadED	USBH_BA+0x028	R/W	Host Controller Bulk Head ED Register	0x0000_0000
HcBulkCurrentED	USBH_BA+0x02C	R/W	Host Controller Bulk Current ED Register	0x0000_0000
HcDoneHead	USBH_BA+0x030	R/W	Host Controller Done Head Register	0x0000_0000
HcFmInterval	USBH_BA+0x034	R/W	Host Controller Frame Interval Register	0x0000_2EDF
HcFmRemaining	USBH_BA+0x038	R	Host Controller Frame Remaining Register	0x0000_0000
HcFmNumber	USBH_BA+0x03C	R	Host Controller Frame Number Register	0x0000_0000
HcPeriodicStart	USBH_BA+0x040	R/W	Host Controller Periodic Start Register	0x0000_0000
HcLSThreshold	USBH_BA+0x044	R/W	Host Controller Low-speed Threshold Register	0x0000_0628
HcRhDescriptorA	USBH_BA+0x048	R/W	Host Controller Root Hub Descriptor A Register	0x0100_0902
HcRhDescriptorB	USBH_BA+0x04C	R/W	Host Controller Root Hub Descriptor B Register	0x0000_0000
HcRhStatus	USBH_BA+0x050	R/W	Host Controller Root Hub Status Register	0x0000_0000
HcRhPortStatus1	USBH_BA+0x058	R/W	Host Controller Root Hub Port Status [1]	0x0000_0000
HcPhyControl	USBH_BA+0x200	R/W	Host Controller PHY Control Register	0x080F_0000
HcMiscControl	USBH_BA+0x204	R/W	Host Controller Miscellaneous Control Register	0x0000_0000
EHCVNR	HSUSBH_BA+0x000	R	EHCI Version Number Register	0x0095_0020
EHCSPR	HSUSBH_BA+0x004	R	EHCI Structural Parameters Register	0x0000_0012

Register	Offset	R/W	Description	Reset Value
EHCCPR	HSUSBH_BA+0x008	R	EHCI Capability Parameters Register	0x0000_0000
UCMDR	HSUSBH_BA+0x020	R/W	USB Command Register	0x0008_0000
USTSR	HSUSBH_BA+0x024	R/W	USB Status Register	0x0000_1000
UIENR	HSUSBH_BA+0x028	R/W	USB Interrupt Enable Register	0x0000_0000
UFINDR	HSUSBH_BA+0x02C	R/W	USB Frame Index Register	0x0000_0000
UPFLBAR	HSUSBH_BA+0x034	R/W	USB Periodic Frame List Base Address Register	0x0000_0000
UCALAR	HSUSBH_BA+0x038	R/W	USB Current Asynchronous List Address Register	0x0000_0000
UASSTR	HSUSBH_BA+0x03C	R/W	USB Asynchronous Schedule Sleep Timer Register	0x0000_0000
UCFGR	HSUSBH_BA+0x060	R/W	USB Configure Flag Register	0x0000_0000
UPSCR0	HSUSBH_BA+0x064	R/W	USB Port 0 Status and Control Register	0x0000_2000
UPSCR1	HSUSBH_BA+0x068	R/W	USB Port 1 Status and Control Register	0x0000_2000
USBPCR0	HSUSBH_BA+0x0C4	R/W	USB PHY 0 Control Register	0x0000_0060
USBPCR1	HSUSBH_BA+0x0C8	R/W	USB PHY 1 Control Register	0x0000_0020

6.35.7 Register Description

Host Controller Revision Register (HcRevision)

Register	Offset	R/W	Description	Reset Value
HcRevision	USBH_BA+0x000	R	Host Controller Revision Register	0x0000_0110

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REV							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**REV**	**Revision Number** Indicates the Open HCI Specification revision number implemented by the Hardware. Host Controller supports 1.1 specification. (X.Y = XYh).

Host Controller Control Register (HcControl)

Register	Offset	R/W	Description	Reset Value
HcControl	USBH_BA+0x004	R/W	Host Controller Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HCFS		**BLE**	**CLE**	**IE**	**PLE**	**CBSR**	

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**HCFS**	**Host Controller Functional State** This field sets the Host Controller state. The Controller may force a state change from USBSUSPEND to USBRESUME after detecting resume signaling from a downstream port. States are: 00 = USBRESET. 01 = USBRESUME. 10 = USBOPERATIONAL. 11 = USBSUSPEND.
[5]	**BLE**	**Bulk List Enable Bit** 0 = Processing of the Bulk list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Bulk list in the next frame Enabled.
[4]	**CLE**	**Control List Enable Bit** 0 = Processing of the Control list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Control list in the next frame Enabled.
[3]	**IE**	**Isochronous List Enable Bit** Both IE and PLE (HcControl[2]) high enables Host Controller to process the Isochronous list. Either IE or PLE (HcControl[2]) is low disables Host Controller to process the Isochronous list. 0 = Processing of the Isochronous list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Isochronous list in the next frame Enabled, if the PLE (HcControl[2]) is high, too.

[2]	**PLE**	**Periodic List Enable Bit** When set, this bit enables processing of the Periodic (interrupt and isochronous) list. The Host Controller checks this bit prior to attempting any periodic transfers in a frame. 0 = Processing of the Periodic (Interrupt and Isochronous) list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Periodic (Interrupt and Isochronous) list in the next frame Enabled. **Note:** To enable the processing of the Isochronous list, user has to set both PLE and IE (HcControl[3]) high.
[1:0]	**CBSR**	**Control Bulk Service Ratio** This specifies the service ratio between Control and Bulk EDs. Before processing any of the non-periodic lists, HC must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this value. 00 = Number of Control EDs over Bulk EDs served is 1:1. 01 = Number of Control EDs over Bulk EDs served is 2:1. 10 = Number of Control EDs over Bulk EDs served is 3:1. 11 = Number of Control EDs over Bulk EDs served is 4:1.

Host Controller Command Status Register (HcCommandStatus)

Register	Offset	R/W	Description	Reset Value
HcCommandStatus	USBH_BA+0x008	R/W	Host Controller Command Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**SOC**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**BLF**	**CLF**	**HCR**

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**SOC**	**Schedule Overrun Count** These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SO (HcInterruptStatus[0]) has already been set.
[15:3]	**Reserved**	Reserved.
[2]	**BLF**	**Bulk List Filled** Set high to indicate there is an active TD on the Bulk list. This bit may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Bulk list. 0 = No active TD found or Host Controller begins to process the head of the Bulk list. 1 = An active TD added or found on the Bulk list.
[1]	**CLF**	**Control List Filled** Set high to indicate there is an active TD on the Control List. It may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Control List. 0 = No active TD found or Host Controller begins to process the head of the Control list. 1 = An active TD added or found on the Control list.
[0]	**HCR**	**Host Controller Reset** This bit is set to initiate the software reset of Host Controller. This bit is cleared by the Host Controller, upon completed of the reset operation. This bit, when set, didn't reset the Root Hub and no subsequent reset signaling be asserted to its downstream ports. 0 = Host Controller is not in software reset state. 1 = Host Controller is in software reset state.

Host Controller Interrupt Status Register (HcInterruptStatus)

Register	Offset	R/W	Description	Reset Value
HcInterruptStatus	USBH_BA+0x00C	R/W	Host Controller Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**RHSC**	**FNO**	**Reserved**	**RD**	**SF**	**WDH**	**SO**

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change** This bit is set when the content of HcRhStatus or the content of HcRhPortStatus1 register has changed. 0 = The content of HcRhStatus and the content of HcRhPortStatus1 register didn't change. 1 = The content of HcRhStatus or the content of HcRhPortStatus1 register has changed.
[5]	**FNO**	**Frame Number Overflow** This bit is set when bit 15 of Frame Number changes from 1 to 0 or from 0 to 1. 0 = The bit 15 of Frame Number didn't change. 1 = The bit 15 of Frame Number changes from 1 to 0 or from 0 to 1.
[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected** Set when Host Controller detects resume signaling on a downstream port. 0 = No resume signaling detected on a downstream port. 1 = Resume signaling detected on a downstream port.
[2]	**SF**	**Start of Frame** Set when the Frame Management functional block signals a 'Start of Frame' event. Host Control generates a SOF token at the same time. 0 = Not the start of a frame. 1 = Indicate the start of a frame and Host Controller generates a SOF token.
[1]	**WDH**	**Write Back Done Head** Set after the Host Controller has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared. 0 = Host Controller didn't update HccaDoneHead. 1 = Host Controller has written HcDoneHead to HccaDoneHead.

[0]	**SO**	**Scheduling Overrun** Set when the List Processor determines a Schedule Overrun has occurred. 0 = Schedule Overrun didn't occur. 1 = Schedule Overrun has occurred.

Host Controller Interrupt Enable Register (HcInterruptEnable)

Register	Offset	R/W	Description	Reset Value
HcInterruptEnable	USBH_BA+0x010	R/W	Host Controller Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
MIE	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	RHSC	FNO	Reserved	RD	SF	WDH	SO

Bits	Description	
[31]	**MIE**	**Master Interrupt Enable Bit** This bit is a global interrupt enable. A write of '1' allows interrupts to be enabled via the specific enable bits listed above. Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled even if the corresponding bit in HcInterruptEnable is high. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high.
[30:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled.
[5]	**FNO**	**Frame Number Overflow Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled. Read Operation: 0 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled.

[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled. Read Operation: 0 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled.
[2]	**SF**	**Start of Frame Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled. Read Operation: 0 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled.
[1]	**WDH**	**Write Back Done Head Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled. Read Operation: 0 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled.
[0]	**SO**	**Scheduling Overrun Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled. Read Operation: 0 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled.

Host Controller Interrupt Disable Register (HcInterruptDisable)

Register	Offset	R/W	Description	Reset Value
HcInterruptDisable	USBH_BA+0x014	R/W	Host Controller Interrupt Disable Register	0x0000_0000

31	30	29	28	27	26	25	24
MIE	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	RHSC	FNO	Reserved	RD	SF	WDH	SO

Bits	Description	
[31]	**MIE**	**Master Interrupt Disable Bit** Global interrupt disable. Writing '1' to disable all interrupts. Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled if the corresponding bit in HcInterruptEnable is high. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled even if the corresponding bit in HcInterruptEnable is high. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high.
[30:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled.
[5]	**FNO**	**Frame Number Overflow Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. Read Operation: 0 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled.

[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. Read Operation: 0 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled.
[2]	**SF**	**Start of Frame Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. Read Operation: 0 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled.
[1]	**WDH**	**Write Back Done Head Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. Read Operation: 0 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled.
[0]	**SO**	**Scheduling Overrun Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. Read Operation: 0 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled.

Host Controller Communication Area Register (HcHCCA)

Register	Offset	R/W	Description	Reset Value
HcHCCA	USBH_BA+0x018	R/W	Host Controller Communication Area Register	0x0000_0000

31	30	29	28	27	26	25	24
HCCA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
HCCA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
HCCA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:8]	**HCCA**	**Host Controller Communication Area** Pointer to indicate the base address of the Host Controller Communication Area (HCCA).
[7:0]	**Reserved**	Reserved.

Host Controller Period Current ED Register (HcPeriodCurrentED)

Register	Offset	R/W	Description	Reset Value
HcPeriodCurrentED	USBH_BA+0x01C	R	Host Controller Period Current ED Register	0x0000_0000

31	30	29	28	27	26	25	24
PCED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PCED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PCED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PCED				**Reserved**			

Bits	Description	
[31:4]	**PCED**	**Periodic Current ED** Pointer to indicate the physical address of the current Isochronous or Interrupt Endpoint Descriptor.
[3:0]	**Reserved**	Reserved.

Host Controller Control Head ED Register (HcControlHeadED)

Register	Offset	R/W	Description	Reset Value
HcControlHeadED	USBH_BA+0x020	R/W	Host Controller Control Head ED Register	0x0000_0000

31	30	29	28	27	26	25	24
CHED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CHED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CHED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CHED				**Reserved**			

Bits	Description	
[31:4]	**CHED**	**Control Head ED** Pointer to indicate the physical address of the first Endpoint Descriptor of the Control list.
[3:0]	**Reserved**	Reserved.

Host Controller Control Current ED Register (HcControlCurrentED)

Register	Offset	R/W	Description	Reset Value
HcControlCurrentED	USBH_BA+0x024	R/W	Host Controller Control Current ED Register	0x0000_0000

31	30	29	28	27	26	25	24
CCED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CCED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CCED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCED				**Reserved**			

Bits	Description	
[31:4]	**CCED**	**Control Current Head ED** Pointer to indicate the physical address of the current Endpoint Descriptor of the Control list.
[3:0]	**Reserved**	Reserved.

Host Controller Bulk Head ED Register (HcBulkHeadED)

Register	Offset	R/W	Description	Reset Value
HcBulkHeadED	USBH_BA+0x028	R/W	Host Controller Bulk Head ED Register	0x0000_0000

31	30	29	28	27	26	25	24
BHED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BHED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BHED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BHED				**Reserved**			

Bits	Description	
[31:4]	**BHED**	**Bulk Head ED** Pointer to indicate the physical address of the first Endpoint Descriptor of the Bulk list.
[3:0]	**Reserved**	Reserved.

Host Controller Bulk Current ED Register (HcBulkCurrentED)

Register	Offset	R/W	Description	Reset Value
HcBulkCurrentED	USBH_BA+0x02C	R/W	Host Controller Bulk Current ED Register	0x0000_0000

31	30	29	28	27	26	25	24
BCED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BCED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BCED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BCED				**Reserved**			

Bits	Description	
[31:4]	**BCED**	**Bulk Current Head ED** Pointer to indicate the physical address of the current Endpoint Descriptor of the Bulk list.
[3:0]	**Reserved**	Reserved.

Host Controller Done Head Register (HcDoneHead)

Register	Offset	R/W	Description	Reset Value
HcDoneHead	USBH_BA+0x030	R/W	Host Controller Done Head Register	0x0000_0000

31	30	29	28	27	26	25	24
DH							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DH							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DH							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DH				**Reserved**			

Bits	Description	
[31:4]	**DH**	**Done Head** Pointer to indicate the physical address of the last completed Transfer Descriptor that was added to the Done queue.
[3:0]	**Reserved**	Reserved.

Host Controller Frame Interval Register (HcFmInterval)

Register	Offset	R/W	Description	Reset Value
HcFmInterval	USBH_BA+0x034	R/W	Host Controller Frame Interval Register	0x0000_2EDF

31	30	29	28	27	26	25	24
FIT	**Reserved**	**FSMPS**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FSMPS							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**FI**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FI							

Bits	Description	
[31]	**FIT**	**Frame Interval Toggle** This bit is toggled by Host Controller Driver when it loads a new value into FI (HcFmInterval[13:0]). 0 = Host Controller Driver didn't load new value into FI (HcFmInterval[13:0]). 1 = Host Controller Driver loads a new value into FI (HcFmInterval[13:0]).
[30]	**Reserved**	Reserved.
[29:16]	**FSMPS**	**FS Largest Data Packet** This field specifies a value that is loaded into the Largest Data Packet Counter at the beginning of each frame.
[15:14]	**Reserved**	Reserved.
[13:0]	**FI**	**Frame Interval** This field specifies the length of a frame as (bit times - 1). For 12,000 bit times in a frame, a value of 11,999 is stored here.

Host Controller Frame Remaining Register (HcFmRemaining)

Register	Offset	R/W	Description	Reset Value
HcFmRemaining	USBH_BA+0x038	R	Host Controller Frame Remaining Register	0x0000_0000

31	30	29	28	27	26	25	24
FRT	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		FR					
7	6	5	4	3	2	1	0
FR							

Bits	Description	
[31]	**FRT**	**Frame Remaining Toggle** This bit is loaded from the FIT (HcFmInterval[31]) whenever FR (HcFmRemaining[13:0]) reaches 0.
[30:14]	**Reserved**	Reserved.
[13:0]	**FR**	**Frame Remaining** When the Host Controller is in the USBOPERATIONAL state, this 14-bit field decrements each 12 MHz clock period. When the count reaches 0, (end of frame) the counter reloads with Frame Interval. In addition, the counter loads when the Host Controller transitions into USBOPERATIONAL.

Host Controller Frame Number Register (HcFmNumber)

Register	Offset	R/W	Description	Reset Value
HcFmNumber	USBH_BA+0x03C	R	Host Controller Frame Number Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FN**	**Frame Number** This 16-bit incrementing counter field is incremented coincident with the re-load of FR (HcFmRemaining[13:0]). The count rolls over from ‘FFFFh’ to ‘0h.’

Host Controller Periodic Start Register (HcPeriodicStart)

Register	Offset	R/W	Description	Reset Value
HcPeriodicStart	USBH_BA+0x040	R/W	Host Controller Periodic Start Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**PS**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PS							

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:0]	**PS**	**Periodic Start** This field contains a value used by the List Processor to determine where in a frame the Periodic List processing must begin.

Host Controller Low-speed Threshold Register (HcLSThreshold)

Register	Offset	R/W	Description	Reset Value
HcLSThreshold	USBH_BA+0x044	R/W	Host Controller Low-speed Threshold Register	0x0000_0628

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**LST**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LST							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**LST**	**Low-speed Threshold** This field contains a value which is compared to the FR (HcFmRemaining[13:0]) field prior to initiating a Low-speed transaction. The transaction is started only if FR (HcFmRemaining[13:0]) >= this field. The value is calculated by Host Controller Driver with the consideration of transmission and setup overhead.

Host Controller Root Hub Descriptor A Register (HcRhDescriptorA)

Register	Offset	R/W	Description	Reset Value
HcRhDescriptorA	USBH_BA+0x048	R/W	Host Controller Root Hub Descriptor A Register	0x0100_0902

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**NOCP**	**OCPM**	**Reserved**		**PSM**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NDP							

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**NOCP**	**No Overcurrent Protection** This bit describes how the overcurrent status for the Root Hub ports reported. 0 = Overcurrent status is reported. 1 = Overcurrent status is not reported.
[11]	**OCPM**	**Overcurrent Protection Mode** This bit describes how the overcurrent status for the Root Hub ports reported. This bit is only valid when NOCP (HcRhDescriptorA[12]) is cleared. 0 = Global overcurrent. 1 = Individual overcurrent.
[10:9]	**Reserved**	Reserved.
[8]	**PSM**	**Power Switching Mode** This bit is used to specify how the power switching of the Root Hub ports is controlled. 0 = Global switching. 1 = Individual switching.
[7:0]	**NDP**	**Number Downstream Ports** USB host control supports two downstream ports in this series of chip.

Host Controller Root Hub Descriptor B Register (HcRhDescriptorB)

Register	Offset	R/W	Description	Reset Value
HcRhDescriptorB	USBH_BA+0x04C	R/W	Host Controller Root Hub Descriptor B Register	0x0000_0000

31	30	29	28	27	26	25	24
PPCM							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PPCM							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:16]	**PPCM**	**Port Power Control Mask** Global power switching. This field is only valid if Power Switching Mode is set (individual port switching). When set, the port only responds to individual port power switching commands (Set/Clear Port Power). When cleared, the port only responds to global power switching commands (Set/Clear Global Power). 0 = Port power controlled by global power switching. 1 = Port power controlled by port power switching. **Note:** PPCM[15:3] and PPCM[0] are reserved.
[15:0]	**Reserved**	Reserved.

Host Controller Root Hub Status Register (HcRhStatus)

Register	Offset	R/W	Description	Reset Value
HcRhStatus	USBH_BA+0x050	R/W	Host Controller Root Hub Status Register	0x0000_0000

31	30	29	28	27	26	25	24
CRWE	Reserved						
23	22	21	20	19	18	17	16
Reserved						OCIC	LPSC
15	14	13	12	11	10	9	8
DRWE	Reserved						
7	6	5	4	3	2	1	0
Reserved						OCI	LPS

Bits	Description	
[31]	**CRWE**	**Clear Remote Wake-up Enable Bit** This bit is use to clear DRWE (HcRhStatus[15]). This bit always reads as 0. Write Operation: 0 = No effect. 1 = Clear DRWE (HcRhStatus[15]).
[31:18]	**Reserved**	Reserved.
[17]	**OCIC**	**Overcurrent Indicator Change** This bit is set by hardware when a change has occurred in OCI (HcRhStatus[1]). Write 1 to clear this bit to 0. 0 = OCI (HcRhStatus[1]) didn't change. 1 = OCI (HcRhStatus[1]) changed.
[16]	**LPSC**	**Set Global Power** In global power mode (PSM (HcRhDescriptorA[8]) = 0), this bit is written to 1 to enable power to all ports. This bit always reads as 0. Write Operation: 0 = No effect. 1 = Set global power.
[15]	**DRWE**	**Device Remote Wakeup Enable Bit** This bit controls if port's Connect Status Change as a remote wake-up event. Remote wake-up can only be triggered while port is in suspend state. Write Operation: 0 = No effect. 1 = Connect Status Change as a remote wake-up event Enabled. Read Operation: 0 = Connect Status Change as a remote wake-up event Disabled. 1 = Connect Status Change as a remote wake-up event Enabled.

[14:2]	**Reserved**	Reserved.
[1]	**OCI**	**overcurrent Indicator** This bit reflects the state of the overcurrent status pin. This field is only valid if NOCP (HcRhDescriptorA[12]) and OCPM (HcRhDescriptorA[11]) are cleared. 0 = No overcurrent condition. 1 = Overcurrent condition.
[0]	**LPS**	**Clear Global Power** In global power mode (PSM (HcRhDescriptorA[8]) = 0), this bit is written to 1 to clear all ports' power. This bit always reads as 0. Write Operation: 0 = No effect. 1 = Clear global power.

Host Controller Root Hub Port Status (HcRhPortStatus1)

Register	Offset	R/W	Description	Reset Value
HcRhPortStatus1	USBH_BA+0x058	R/W	Host Controller Root Hub Port Status [1]	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			PRSC	OCIC	PSSC	PESC	CSC
15	14	13	12	11	10	9	8
Reserved						LSDA	PPS
7	6	5	4	3	2	1	0
Reserved			PRS	POCI	PSS	PES	CCS

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**PRSC**	**Port Reset Status Change** This bit indicates that the port reset signal has completed. Write 1 to clear this bit to 0. 0 = Port reset is not complete. 1 = Port reset is complete.
[19]	**OCIC**	**Port Overcurrent Indicator Change** This bit is set when POCI (HcRhPortStatus1[3]) changes. Write 1 to clear this bit to 0. 0 = POCI (HcRhPortStatus1[3]) didn't change. 1 = POCI (HcRhPortStatus1[3]) changed.
[18]	**PSSC**	**Port Suspend Status Change** This bit indicates the completion of the selective resume sequence for the port. Write 1 to clear this bit to 0. 0 = Port resume is not complete. 1 = Port resume is complete.
[17]	**PESC**	**Port Enable Status Change** This bit indicates that the port has been disabled (PES (HcRhPortStatus1[1]) cleared) due to a hardware event. Write 1 to clear this bit to 0. 0 = PES (HcRhPortStatus1[1]) didn't change. 1 = PES (HcRhPortStatus1[1]) changed.

[16]	**CSC**	**Connect Status Change** This bit indicates connect or disconnect event has been detected (CCS (HcRhPortStatus1[0]) changed). Write 1 to clear this bit to 0. 0 = No connect/disconnect event (CCS (HcRhPortStatus1[0]) didn't change). 1 = Hardware detection of connect/disconnect event (CCS (HcRhPortStatus1[0]) changed).
[15:10]	**Reserved**	Reserved.
[9]	**LSDA**	**Low Speed Device Attached or Clear Port Power (Read/Write)** This bit defines the speed (and bus idle) of the attached device. It is only valid when CCS (HcRhPortStatus1[0]) is set. This bit is also used to clear port power. Write Operation: 0 = No effect. 1 = Clear PPS (HcRhPortStatus1[8]). Read Operation: 0 = Full Speed device. 1 = Low-speed device.
[8]	**PPS**	**Port Power Status or Set Port Power (Read/Write)** This bit reflects the power state of the port regardless of the power switching mode. Write Operation: 0 = No effect. 1 = Port Power Enabled. Read Operation: 0 = Port power is Disabled. 1 = Port power is Enabled.
[7:5]	**Reserved**	Reserved.
[4]	**PRS**	**Port Reset Status or Set Port Reset (Read/Write)** This bit reflects the reset state of the port. Write Operation: 0 = No effect. 1 = Set port reset. Read Operation 0 = Port reset signal is not active. 1 = Port reset signal is active.
[3]	**POCI**	**Port Overcurrent Indicator or Clear Port Suspend (Read/Write)** This bit reflects the state of the overcurrent status pin dedicated to this port. This field is only valid if NOCP (HcRhDescriptorA[12]) is cleared and OCPM (HcRhDescriptorA[11]) is set. This bit is also used to initiate the selective result sequence for the port. Write Operation: 0 = No effect. 1 = Clear port suspend. Read Operation: 0 = No overcurrent condition. 1 = Overcurrent condition.

[2]	**PSS**	**Port Suspend Status or Set Port Suspend (Read/Write)** This bit indicates the port is suspended Write Operation: 0 = No effect. 1 = Set port suspend. Read Operation: 0 = Port is not suspended. 1 = Port is selectively suspended.
[1]	**PES**	**Port Enable Status or Set Port Enable (Read/Write)** Write Operation: 0 = No effect. 1 = Set port enable. Read Operation: 0 = Port Disabled. 1 = Port Enabled.
[0]	**CCS**	**Current Connect Status or Clear Port Enable (Read/Write)** Write Operation: 0 = No effect. 1 = Clear port enable. Read Operation: 0 = No device connected. 1 = Device connected.

Host Controller PHY Control Register (HcPhyControl)

Register	Offset	R/W	Description	Reset Value
HcPhyControl	USBH_BA+0x200	R/W	Host Controller PHY Control Register	0x080F_0000

31	30	29	28	27	26	25	24
Reserved				**STBYEN**	**Reserved**		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**STBYEN**	**USB Transceiver Standby Enable Bit** This bit controls if USB transceiver could enter the standby mode to reduce power consumption. 0 = The USB transceiver would never enter the standby mode. 1 = The USB transceiver will enter standby mode while port is in power off state (port power is inactive).
[26:0]	**Reserved**	Reserved.

Host Controller Miscellaneous Control Register (HcMiscControl)

Register	Offset	R/W	Description	Reset Value
HcMiscControl	USBH_BA+0x204	R/W	Host Controller Miscellaneous Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**DPRT1**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**OCAL**	**Reserved**	**ABORT**	**Reserved**

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DPRT1**	**Disable Port 1** This bit controls if the connection between USB host controller and transceiver of port 1 is disabled. If the connection is disabled, the USB host controller will not recognize any event of USB bus. Set this bit high, the transceiver of port 1 will also be forced into the standby mode no matter what USB host controller operation is. 0 = The connection between USB host controller and transceiver of port 1 Enabled. 1 = The connection between USB host controller and transceiver of port 1 Disabled and the transceiver of port 1 will also be forced into the standby mode.
[15:4]	**Reserved**	Reserved.
[3]	**OCAL**	**Overcurrent Active Low** This bit controls the polarity of overcurrent flag from external power IC. 0 = Overcurrent flag is high active. 1 = Overcurrent flag is low active.
[2]	**Reserved**	Reserved.
[1]	**ABORT**	**AHB Bus Error Response** This bit indicates there is an Error response received in AHB bus. 0 = No Error response received. 1 = Error response received.
[0]	**Reserved**	Reserved.

EHCI Version Number Register (EHCVNR)

Register	Offset	R/W	Description	Reset Value
EHCVNR	HSUSBH_BA+0x000	R	EHCI Version Number Register	0x0095_0020

31	30	29	28	27	26	25	24
VERSION							
23	22	21	20	19	18	17	16
VERSION							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CRLEN							

Bits	Description	
[31:16]	**VERSION**	**Host Controller Interface Version Number** This is a two-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.
[15:8]	**Reserved**	Reserved.
[7:0]	**CRLEN**	**Capability Registers Length** This register is used as an offset to add to register base to find the beginning of the Operational Register Space.

EHCI Structural Parameters Register (EHCSPR)

Register	Offset	R/W	Description	Reset Value
EHCSPR	HSUSBH_BA+0x004	R	EHCI Structural Parameters Register	0x0000_0012

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
N_CC				N_PCC			
7	6	5	4	3	2	1	0
Reserved			PPC	N_PORTS			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:12]	**N_CC**	**Number of Companion Controller** This field indicates the number of companion controllers associated with this USB 2.0 host controller. A 0 in this field indicates there are no companion host controllers. Port-ownership hand-off is not supported. Only high-speed devices are supported on the host controller root ports. A value larger than 0 in this field indicates there are companion USB 1.1 host controller(s). Port-ownership hand-offs are supported. High, Full- and Low-speed devices are supported on the host controller root ports.
[11:8]	**N_PCC**	**Number of Ports Per Companion Controller** This field indicates the number of ports supported per companion host controller. It is used to indicate the port routing configuration to system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC.
[7:5]	**Reserved**	Reserved.
[4]	**PPC**	**Port Power Control** This field indicates whether the host controller implementation includes port power control. A 1 in this bit indicates the ports have port power switches. A 0 in this bit indicates the port do not have port power stitches. The value of this field affects the functionality of the Port Power field in each port status and control register.
[3:0]	**N_PORTS**	**Number of Physical Downstream Ports** This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register Space (see Table 2-8). Valid values are in the range of 1H to FH. A 0 in this field is undefined.

EHCI Capability Parameters Register (EHCCPR)

Register	Offset	R/W	Description	Reset Value
EHCCPR	HSUSBH_BA+0x008	R	EHCI Capability Parameters Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
EECP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IST				**Reserved**	**ASPC**	**PFLF**	**AC64**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**EECP**	**EHCI Extended Capabilities Pointer** 0 = No extended capabilities are implemented.
[7:4]	**IST**	**Isochronous Scheduling Threshold** This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is 0, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state.
[3]	**Reserved**	Reserved.
[2]	**ASPC**	**Asynchronous Schedule Park Capability** 0 = This EHCI host controller doesn't support park feature of high-speed queue heads in the Asynchronous Schedule.
[1]	**PFLF**	**Programmable Frame List Flag** 0 = System software must use a frame list length of 1024 elements with this EHCI host controller.
[0]	**AC64**	**64-bit Addressing Capability** 0 = Data structure using 32-bit address memory pointers.

USB Command Register (UCMDR)

Register	Offset	R/W	Description	Reset Value
UCMDR	HSUSBH_BA+0x020	R/W	USB Command Register	0x0008_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
ITC							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	IAAD	ASEN	PSEN	FLSZ		HCRST	RUN

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**ITC**	**Interrupt Threshold Control (Read/Write)** This field is used by system software to select the maximum rate at which the host controller will issue interrupts. The only valid values are defined below. If software writes an invalid value to this register, the results are undefined. Value Maximum Interrupt Interval 0x00 = Reserved. 0x01 = 1 micro-frame. 0x02 = 2 micro-frames. 0x04 = 4 micro-frames. 0x08 = 8 micro-frames (default, equates to 1 ms). 0x10 = 16 micro-frames (2 ms). 0x20 = 32 micro-frames (4 ms). 0x40 = 64 micro-frames (8 ms). Any other value in this register yields undefined results. Software modifications to this bit while HCHalted bit is equal to 0 results in undefined behavior.
[15:7]	**Reserved**	Reserved.
[6]	**IAAD**	**Interrupt on Asynchronous Advance Doorbell (Read/Write)** This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule state, it sets the Interrupt on Asynchronous Advance status bit in the USTSR register. If the Interrupt on Asynchronous Advance Enable bit in the USBINTR register is a 1 then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to a 0 after it has set the Interrupt on Asynchronous Advance status bit in the USTSR register to 1. Software should not write a 1 to this bit when the asynchronous schedule is disabled. Doing so will yield undefined results.

[5]	**ASEN**	**Asynchronous Schedule Enable (Read/Write)** This bit controls whether the host controller skips processing the Asynchronous Schedule. Values mean: 0 = Do not process the Asynchronous Schedule. 1 = Use the ASYNCLISTADDR register to access the Asynchronous Schedule.
[4]	**PSEN**	**Periodic Schedule Enable (Read/Write)** This bit controls whether the host controller skips processing the Periodic Schedule. Values mean: 0 = Do not process the Periodic Schedule. 1 = Use the PERIODICLISTBASE register to access the Periodic Schedule.
[3:2]	**FLSZ**	**Frame List Size (Read Only)** This field is R/W only if Programmable Frame List Flag in the HCCPARAMS registers is set to a 1. This field specifies the size of the frame list. The size the frame list controls which bits in the Frame Index Register should be used for the Frame List Current index. Values mean: 00 = 1024 elements (4096 bytes) Default value. 01 = 512 elements (2048 bytes). 10 = 256 elements (1024 bytes) – for resource-constrained environment. 11 = Reserved.
[1]	**HCRST**	**Host Controller Reset (Read/Write)** This control bit is used by software to reset the host controller. The effects of this on Root Hub registers are similar to a Chip Hardware Reset. When software writes a 1 to this bit, the Host Controller resets its internal pipelines, timers, counters, state machines, etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. All operational registers, including port registers and port state machines are set to their initial values. Port ownership reverts to the companion host controller(s), with the side effects. Software must reinitialize the host controller in order to return the host controller to an operational state. This bit is set to 0 by the Host Controller when the reset process is complete. Software cannot terminate the reset process early by writing 0 to this register. Software should not set this bit to a 1 when the HCHalted bit in the USTSR register is 0. Attempting to reset an actively running host controller will result in undefined behavior.
[0]	**RUN**	**Run/Stop (Read/Write)** When set to a 1, the Host Controller proceeds with execution of the schedule. The Host Controller continues execution as long as this bit is set to a 1. When this bit is set to 0, the Host Controller completes the current and any actively pipelined transactions on the USB and then halts. The Host Controller must halt within 16 micro-frames after software clears the Run bit. The HC Halted bit in the status register indicates when the Host Controller has finished its pending pipelined transactions and has entered the stopped state. Software must not write a 1 to this field unless the host controller is in the Halted state (i.e. HCHalted in the USTSR register is a 1). Doing so will yield undefined results. 0 = Stop. 1 = Run.

USB Status Register (USTSR)

Register	Offset	R/W	Description	Reset Value
USTSR	HSUSBH_BA+0x024	R/W	USB Status Register	0x0000_1000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ASS	PSS	RECLA	HCHalted	Reserved			
7	6	5	4	3	2	1	0
Reserved		IAA	HSERR	FLR	PCD	UERRINT	USBINT

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**ASS**	**Asynchronous Schedule Status (Read Only)** The bit reports the current real status of the Asynchronous Schedule. If this bit is 0 then the status of them Asynchronous Schedule is disabled. If this bit is a 1 then the status of the Asynchronous Schedule is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0).
[14]	**PSS**	**Periodic Schedule Status (Read Only)** The bit reports the current real status of the Periodic Schedule. If this bit is 0 then the status of the Periodic Schedule is disabled. If this bit is a 1 then the status of the Periodic Schedule is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0).
[13]	**RECLA**	**Reclamation (Read Only)** This is a read-only status bit, which is used to detect an empty asynchronous schedule.
[12]	**HCHalted**	**HCHalted (Read Only)** This bit is 0 whenever the Run/Stop bit is a 1. The Host Controller sets this bit to 1 after it has stopped executing as a result of the Run/Stop bit being set to 0, either by software or by the Host Controller hardware (e.g. internal error).
[11:6]	**Reserved**	Reserved.
[5]	**IAA**	**Interrupt on Asynchronous Advance** System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a 1 to the Interrupt on Asynchronous Advance Doorbell bit in the USBCMD register. This status bit indicates the assertion of that interrupt source. **Note:** This bit is cleared by writing 1 to it.

[4]	**HSERR**	**Host System Error** The Host Controller sets this bit to 1 when a serious error occurs during a host system access involving the Host Controller module. **Note:** This bit is cleared by writing 1 to it.
[3]	**FLR**	**Frame List Rollover** The Host Controller sets this bit to a 1 when the Frame List Index rolls over from its maximum value to 0. The exact value at which the rollover occurs depends on the frame list size. For example, if the frame list size (as programmed in the Frame List Size field of the USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX[13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a 1 every time FRINDEX[12] toggles. **Note:** This bit is cleared by writing 1 to it.
[2]	**PCD**	**Port Change Detect** The Host Controller sets this bit to a 1 when any port for which the Port Owner bit is set to 0 has a change bit transition from 0 to 1 or a Force Port Resume bit transition from a 0 to a 1 as a result of a J-K transition detected on a suspended port. This bit will also be set as a result of the Connect Status Change being set to a 1 after system software has relinquished ownership of a connected port by writing a 1 to a port's Port Owner bit. This bit is allowed to be maintained in the Auxiliary power well. Alternatively, it is also acceptable that on a D3 to D0 transition of the EHCI HC device, this bit is loaded with the OR of all of the PORTSC change bits (including: Force port resume, overcurrent change, enable/disable change and connect status change). **Note:** This bit is cleared by writing 1 to it.
[1]	**UERRINT**	**USB Error Interrupt** The Host Controller sets this bit to 1 when completion of a USB transaction results in an error condition (e.g., error counter underflow). If the TD on which the error interrupt occurred also had its IOC bit set, both this bit and USBINT bit are set. **Note:** This bit is cleared by writing 1 to it.
[0]	**USBINT**	**USB Interrupt** The Host Controller sets this bit to 1 on the completion of a USB transaction, which results in the retirement of a Transfer Descriptor that had its IOC bit set. The Host Controller also sets this bit to 1 when a short packet is detected (actual number of bytes received was less than the expected number of bytes). **Note:** This bit is cleared by writing 1 to it.

USB Interrupt Enable Register (UIENR)

Register	Offset	R/W	Description	Reset Value
UIENR	HSUSBH_BA+0x028	R/W	USB Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		IAAEN	HSERREN	FLREN	PCIEN	UERRIEN	USBIEN

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**IAAEN**	**Interrupt on Asynchronous Advance Enable or Disable Bit** When this bit is a 1, and the Interrupt on Asynchronous Advance bit in the USTSR register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the Interrupt on Asynchronous Advance bit. 0 = Interrupt on Asynchronous Advance Disabled. 1 = Interrupt on Asynchronous Advance Enabled.
[4]	**HSERREN**	**Host System Error Enable or Disable Bit** When this bit is a 1, and the Host System Error Status bit in the USTSR register is a 1, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Host System Error bit. 0 = Host System Error interrupt Disabled. 1 = Host System Error interrupt Enabled.
[3]	**FLREN**	**Frame List Rollover Enable or Disable Bit** When this bit is a 1, and the Frame List Rollover bit in the USTSR register is a 1, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Frame List Rollover bit. 0 = Frame List Rollover interrupt Disabled. 1 = Frame List Rollover interrupt Enabled.
[2]	**PCIEN**	**Port Change Interrupt Enable or Disable Bit** When this bit is a 1, and the Port Change Detect bit in the USTSR register is a 1, the host controller will issue an interrupt. The interrupt is acknowledged by software clearing the Port Change Detect bit. 0 = Port Change interrupt Disabled. 1 = Port Change interrupt Enabled.

[1]	**UERRIEN**	**USB Error Interrupt Enable or Disable Bit** When this bit is a 1, and the UERRINT bit in the USTSR register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the UERRINT bit. 0 = USB Error interrupt Disabled. 1 = USB Error interrupt Enabled.
[0]	**USBIEN**	**USB Interrupt Enable or Disable Bit** When this bit is a 1, and the USBINT bit in the USTSR register is a 1, the host controller will issue an interrupt at the next interrupt threshold. The interrupt is acknowledged by software clearing the USBINT bit. 0 = USB interrupt Disabled. 1 = USB interrupt Enabled.

USB Frame Index Register (UFINDR)

Register	Offset	R/W	Description	Reset Value
UFINDR	HSUSBH_BA+0x02C	R/W	USB Frame Index Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		FI					
7	6	5	4	3	2	1	0
FI							

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:0]	**FI**	**Frame Index** The value in this register increment at the end of each time frame (e.g. micro-frame). Bits [N:3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register. **FLSZ (UCMDR[3:2] Number Elements N** 0x0 1024 12 0x1 512 11 0x2 256 10 0x3 Reserved

USB Periodic Frame List Base Address Register (UPFLBAR)

Register	Offset	R/W	Description	Reset Value
UPFLBAR	HSUSBH_BA+0x034	R/W	USB Periodic Frame List Base Address Register	0x0000_0000

31	30	29	28	27	26	25	24
BADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BADDR				**Reserved**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:12]	**BADDR**	**Base Address** These bits correspond to memory address signals [31:12], respectively.
[11:0]	**Reserved**	Reserved.

USB Current Asynchronous List Address Register (UCALAR)

Register	Offset	R/W	Description	Reset Value
UCALAR	HSUSBH_BA+0x038	R/W	USB Current Asynchronous List Address Register	0x0000_0000

31	30	29	28	27	26	25	24
LPL							
23	22	21	20	19	18	17	16
LPL							
15	14	13	12	11	10	9	8
LPL							
7	6	5	4	3	2	1	0
LPL			Reserved				

Bits	Description	
[31:5]	LPL	**Link Pointer Low** These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH).
[4:0]	Reserved	Reserved.

USB Asynchronous Schedule Sleep Timer Register (UASSTR)

Register	Offset	R/W	Description	Reset Value
UASSTR	HSUSBH_BA+0x03C	R/W	USB Asynchronous Schedule Sleep Timer Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				ASSTMR			
7	6	5	4	3	2	1	0
ASSTMR							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**ASSTMR**	**Asynchronous Schedule Sleep Timer** This field defines the AsyncSchedSleepTime of EHCI spec. The asynchronous schedule sleep timer is used to control how often the host controller fetches asynchronous schedule list from system memory while the asynchronous schedule is empty. The default value of this timer is 12'h0 and can be set to 12'hBD6. Because this timer is implemented in UTMI clock (30 MHz) domain, the default sleeping time will be about 100us.

USB Configure Flag Register (UCFGR)

Register	Offset	R/W	Description	Reset Value
UCFGR	HSUSBH_BA+0x060	R/W	USB Configure Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							CF

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CF**	**Configure Flag** Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. Bit values and side-effects are listed below. 0 = Port routing control logic default-routes each port to an implementation dependent classic host controller. 1 = Port routing control logic default-routes all ports to this host controller.

USB Port Status and Control Register (UPSCR)

Register	Offset	R/W	Description	Reset Value
UPSCR0	HSUSBH_BA+0x064	R/W	USB Port 0 Status and Control Register	0x0000_2000
UPSCR1	HSUSBH_BA+0x068	R/W	USB Port 1 Status and Control Register	0x0000_2000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				PTC			
15	14	13	12	11	10	9	8
Reserved		PO	PP	LSTS		Reserved	PRST
7	6	5	4	3	2	1	0
SUSPEND	FPR	OCC	OCA	PEC	PE	CSC	CCS

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:16]	**PTC**	**Port Test Control (Read/Write)** When this field is 0, the port is NOT operating in a test mode. A non-zero value indicates that it is operating in test mode and the specific test mode is indicated by the specific value. The encoding of the test mode bits are (0x6 ~ 0xF are reserved): Bits Test Mode 0x0 = Test mode not enabled. 0x1 = Test J_STATE. 0x2 = Test K_STATE. 0x3 = Test SE0_NAK. 0x4 = Test Packet. 0x5 = Test FORCE_ENABLE.
[15:14]	**Reserved**	Reserved.
[13]	**PO**	**Port Owner (Read/Write)** This bit unconditionally goes to a 0b when the Configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is 0. System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a 1 to this bit when the attached device is not a high-speed device. A 1 in this bit means that a companion host controller owns and controls the port.
[12]	**PP**	**Port Power** Host controller has port power control switches. This bit represents the Current setting of the switch (0 = off, 1 = on). When power is not available on a port (i.e. PP equals a 0), the port is nonfunctional and will not report attaches, detaches, etc. When an overcurrent condition is detected on a powered port and PPC is a 1, the PP bit in each affected port may be transitioned by the host controller from a 1 to 0 (removing power from the port).

[11:10]	**LSTS**	**Line Status (Read Only)** These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. These bits are used for detection of low-speed USB devices prior to the port reset and enable sequence. This field is valid only when the port enable bit is 0 and the current connect status bit is set to a 1. The encoding of the bits are: Bits[11:10] USB State Interpretation 00 = SE0 Not Low-speed device, perform EHCI reset. 01 = K-state Low-speed device, release ownership of port. 10 = J-state Not Low-speed device, perform EHCI reset. 11 = Undefined Not Low-speed device, perform EHCI reset. This value of this field is undefined if Port Power is 0.
[9]	**Reserved**	Reserved.
[8]	**PRST**	**Port Reset (Read/Write)** When software writes a 1 to this bit (from a 0), the bus reset sequence as defined in the USB Specification Revision 2.0 is started. Software writes a 0 to this bit to terminate the bus reset sequence. Software must keep this bit at a 1 long enough to ensure the reset sequence, as specified in the USB Specification Revision 2.0, completes. **Note:** when software writes this bit to a 1, it must also write a 0 to the Port Enable bit. Note that when software writes a 0 to this bit there may be a delay before the bit status changes to a 0. The bit status will not read as a 0 until after the reset has completed. If the port is in high-speed mode after reset is complete, the host controller will automatically enable this port (e.g. set the Port Enable bit to a 1). A host controller must terminate the reset and stabilize the state of the port within 2 milliseconds of software transitioning this bit from a 1 to a 0. For example: if the port detects that the attached device is high-speed during reset, then the host controller must have the port in the enabled state within 2ms of software writing this bit to a 0. The HCHalted bit in the USTSR register should be a 0 before software attempts to use this bit. The host controller may hold Port Reset asserted to a 1 when the HCHalted bit is a 1. This field is 0 if Port Power is 0. 0 = Port is not in Reset. 1 = Port is in Reset.
[7]	**SUSPEND**	**Suspend (Read/Write)** Port Enabled Bit and Suspend bit of this register define the port states as follows: 00 = Port Disable. 01 = Port Disable. 10 = Port Enable. 11 = Port Suspend. When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction, if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. A write of 0 to this bit is ignored by the host controller. The host controller will unconditionally set this bit to a 0 when: Software sets the Force Port Resume bit to a 0 (from a 1). Software sets the Port Reset bit to a 1 (from a 0). If host software sets this bit to a 1 when the port is not enabled (i.e. Port enabled bit is a 0) the results are undefined. This field is 0 if Port Power is 0. 0 = Port not in suspend state. 1 = Port in suspend state.

[6]	**FPR**	**Force Port Resume (Read/Write)** This functionality defined for manipulating this bit depends on the value of the Suspend bit. For example, if the port is not suspended (Suspend and Enabled bits are a 1) and software transitions this bit to a 1, then the effects on the bus are undefined. Software sets this bit to a 1 to drive resume signaling. The Host Controller sets this bit to a 1 if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a 1 because a J-to-K transition is detected, the Port Change Detect bit in the USTSR register is also set to a 1. If software sets this bit to a 1, the host controller must not set the Port Change Detect bit. Note that when the EHCI controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a 1. Software must appropriately time the Resume and set this bit to a 0 when the appropriate amount of time has elapsed. Writing a 0 (from 1) causes the port to return to high-speed mode (forcing the bus below the port into a high-speed idle). This bit will remain a 1 until the port has switched to the high-speed idle. The host controller must complete this transition within 2 milliseconds of software setting this bit to a 0. This field is 0 if Port Power is 0. 0 = No resume (K-state) detected/driven on port. 1 = Resume detected/driven on port.
[5]	**OCC**	**Overcurrent Change** 1 = This bit gets set to a 1 when there is a change to Overcurrent Active. Software clears this bit by writing a 1 to this bit position. **Note:** This bit is cleared by writing 1 to it.
[4]	**OCA**	**Overcurrent Active (Read Only)** This bit will automatically transition from a 1 to a 0 when the overcurrent condition is removed. 0 = This port does not have an overcurrent condition. 1 = This port currently has an overcurrent condition.
[3]	**PEC**	**Port Enable/Disable Change** For the root hub, this bit gets set to a 1 only when a port is disabled due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification for the definition of a Port Error). Software clears this bit by writing a 1 to it. This field is 0 if Port Power is 0. 0 = No change. 1 = Port enabled/disabled status has changed. **Note:** This bit is cleared by writing 1 to it.
[2]	**PE**	**Port Enabled/Disabled (Read/Write)** Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a 1 to this field. The host controller will only set this bit to a 1 when the reset sequence determines that the attached device is a high-speed device. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled (0b) downstream propagation of data is blocked on this port, except for reset. This field is 0 if Port Power is 0. 0 = Port Disabled. 1 = Port Enabled.

[1]	**CSC**	**Connect Status Change (Read/Write)** Indicate a change has occurred in the port's Current Connect Status. The host controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be "setting" an already-set bit (i.e., the bit will remain set). Software sets this bit to 0 by writing a 1 to it. This field is 0 if Port Power is 0. 0 = No change. 1 = Change in Current Connect Status.
[0]	**CCS**	**Current Connect Status (Read Only)** This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change bit (Bit 1) to be set. This field is 0 if Port Power is 0. 0 = No device is present. 1 = Device is present on port.

USB PHY 0 Control Register (USBPCR0)

Register	Offset	R/W	Description	Reset Value
USBPCR0	HSUSBH_BA+0x0C4	R/W	USB PHY 0 Control Register	0x0000_0060

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**FDISCEN**	**FDISC**	**CLKVALID**	**Reserved**		**SUSPEND**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**FDISCEN**	**Force Disconnection Function Enable Bit** This bit is used to enable Force Disconnection Function (FDISC). 0 = Force Disconnection Function Disabled. 1 = Force Disconnection Function Enabled.
[12]	**FDISC**	**Force Disconnection Function** This bit is used to force the device to disconnect without really removing the device from the port. This bit is valid only if FDISCEN is asserted. 0 = The device is not forced to be disconnected. 1 = The device is forced to disconnected.
[11]	**CLKVALID**	**UTMI Clock Valid** This bit is a flag to indicate if the UTMI clock from USB 2.0 PHY is ready. Software program must prevent to write other control registers before this UTMI clock valid flag is active. 0 = UTMI clock is not valid. 1 = UTMI clock is valid.
[10:9]	**Reserved**	Reserved.
[8]	**SUSPEND**	**Suspend Assertion** This bit controls the suspend mode of USB PHY 0. While PHY was suspended, all circuits of PHY were powered down and outputs are tri-state. This bit is 1'b0 by default. This means the USB PHY 0 is suspended by default. It is necessary to set this bit 1'b1 to make USB PHY 0 leave suspend mode before doing configuration of USB host. 0 = USB PHY 0 was suspended. 1 = USB PHY 0 was not suspended.
[7:0]	**Reserved**	Reserved.

USB PHY 1 Control Register (USBPCR1)

Register	Offset	R/W	Description	Reset Value
USBPCR1	HSUSBH_BA+0x0C8	R/W	USB PHY 1 Control Register	0x0000_0020

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**FDISCEN**	**FDISC**	**Reserved**			**SUSPEND**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**FDISCEN**	**Force Disconnection Function Enable** This bit is used to enable Force Disconnection Function (FDISC). 0 = Force Disconnection Function Disabled. 1 = Force Disconnection Function Enabled.
[12]	**FDISC**	**Force Disconnection Function** This bit is used to force the device to disconnect without really removing the device from the port. This bit is valid only if FDISCEN is asserted. 0 = The device is not forced to disconnect. 1 = The device is forced to disconnect.
[11:9]	**Reserved**	Reserved.
[8]	**SUSPEND**	**Suspend Assertion** This bit controls the suspend mode of USB PHY 1. While PHY was suspended, all circuits of PHY were powered down and outputs are tri-state. This bit is 1'b0 by default. This means the USB PHY 0 is suspended by default. It is necessary to set this bit 1'b1 to make USB PHY 0 leave suspend mode before doing configuration of USB host. 0 = USB PHY 1 suspended. 1 = USB PHY 1 not suspended.
[7:0]	**Reserved**	Reserved.

6.36 USB 1.1 On-The-Go (OTG)

6.36.1 Overview

The OTG controller interfaces to USB PHY and USB controllers which consist of an USB 1.1 host controller and an USB 2.0 FS device controller. The OTG controller supports HNP and SRP protocols defined in the “On-The-Go and Embedded Host Supplement to the USB 2.0 Revision 2.0 Specification”.

An USB frame, including USB host, USB device, and OTG controller, can be configured as Host-only, Device-only, ID dependent or OTG device mode defined in USBROLE (SYS_USBPHY[1:0]). In Host-only mode, the USB frame acts as USB host. The USB frame can support both full-speed and low-speed transfer. In Device-only mode, the USB frame acts as USB device. USB device support full-speed transfer. In ID dependent mode, the USB frame can be USB Host or USB device depending on USB_ID pin state. In OTG device mode, the role of USB frame depends on the definition of OTG specification. The USB frame only supports full-speed transfer when OTG device acts as a peripheral

6.36.2 Features

- Built-in USB PHY
- Configurable to operate as:
  - Host-only
  - Device-only
  - ID dependent: The role of USB frame is only dependent on USB_ID pin state --as USB Host (USB_ID pin is at low) or USB Device (USB_ID pin is at high). Both HNP and SRP protocols are not supported.
  - OTG device: depends on USB_ID pin state to be A-device (USB_ID pin is low) or B-device (USB_ID pin is high). HNP and SRP protocols are supported.

6.36.3 Block Diagram

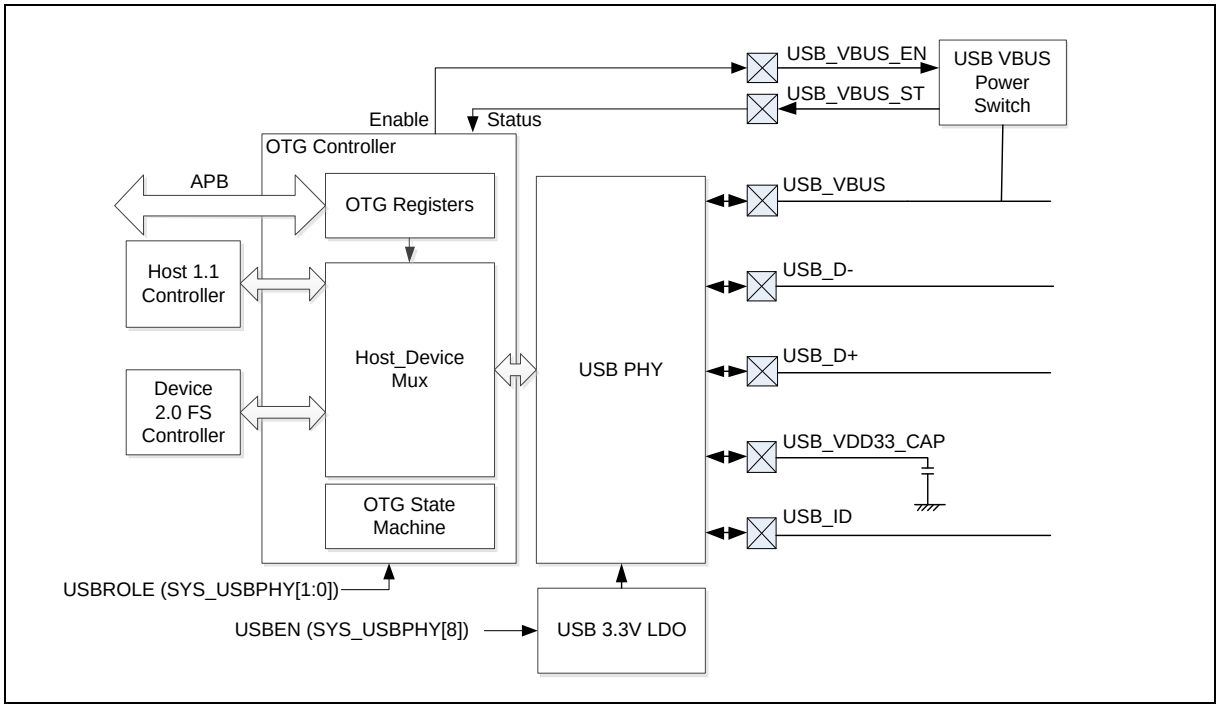


Figure 6.36-1 USB OTG Block Diagram

6.36.4 Basic Configuration

6.36.4.1 OTG Controller Basic Configuration

- Clock Source Configuration
  - Enable OTG peripheral clock on OTGCKEN (CLK_APBCLK0[26]).
- OTG Configuration
  - Refer description of SYS_REGLCTL register to disable the register protection function.
  - Setting role of USB frame is on USBROLE (SYS_USBPHY[1:0]).
  - Enable PHY on USBEN(SYS_USBPHY[8]).
  - Write 1 clear all interrupt status(OTG_INTSTS)

6.36.5 Functional Description

The role of USB frame depends on the setting of USBROLE (SYS_USBPHY[1:0]) and USB_ID pin state. The USBROLE configuration has precedence over USB_ID pin state. User can configure the OTG controller to USB Host mode, USB Device mode, ID dependent mode or OTG device mode. In USB Host mode, the host controller will interact with USB PHY directly. In USB Device mode, the device controller will interact with USB PHY directly. In these cases, the OTG controller is used simply as a multiplexer. In ID dependent mode, USB_ID pin state will decide USB frame to act as USB host or USB device. If the USB_ID pin is FALSE state (low level), the USB frame will act as USB host. If the USB_ID pin is TRUE state (high level), the USB frame will act as USB device. In OTG device mode, the OTG controller will handle OTG HNP and SRP protocols. If the USB_ID pin is FALSE state (low level), the OTG controller will act as an OTG A-device. If the USB_ID pin is TRUE state (high level), the OTG controller will act as an OTG B-device.

6.36.5.1 The Role of USB Frame

USB Device Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 0, the USB frame acts as USB device. The USB host function is not available.

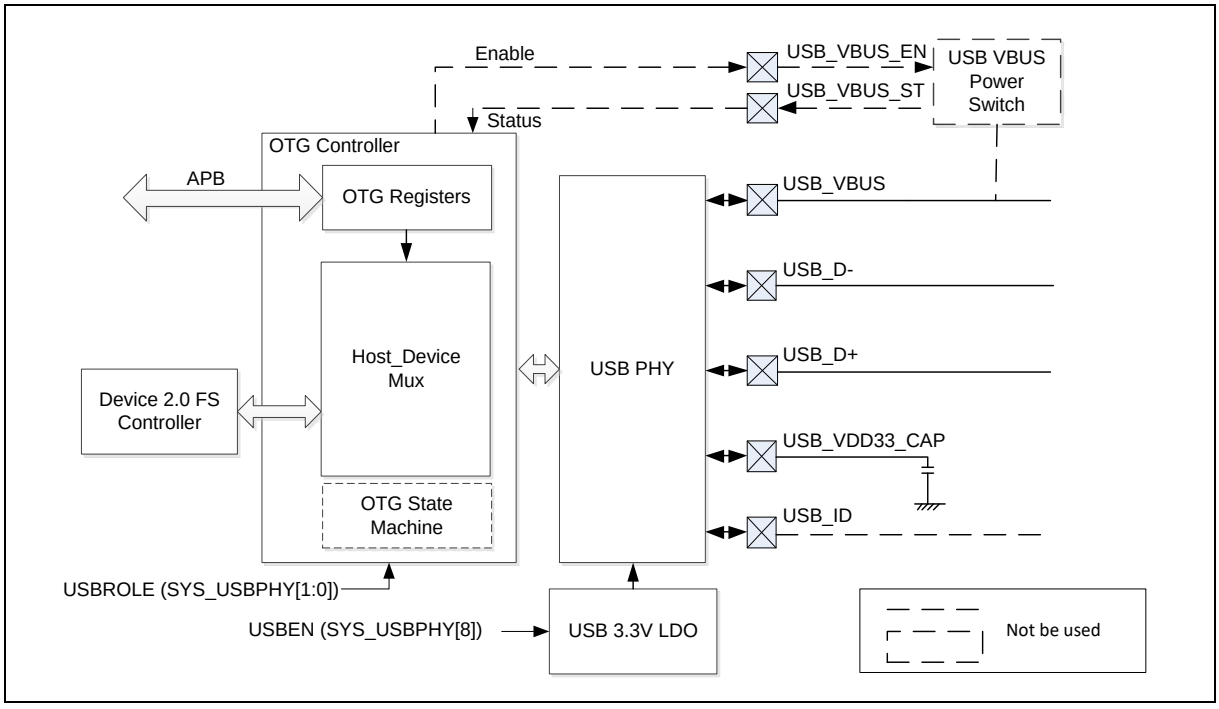


Figure 6.36-2 USB Device Mode

USB Host Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 1, the USB frame acts as USB host. The USB device function is not available.

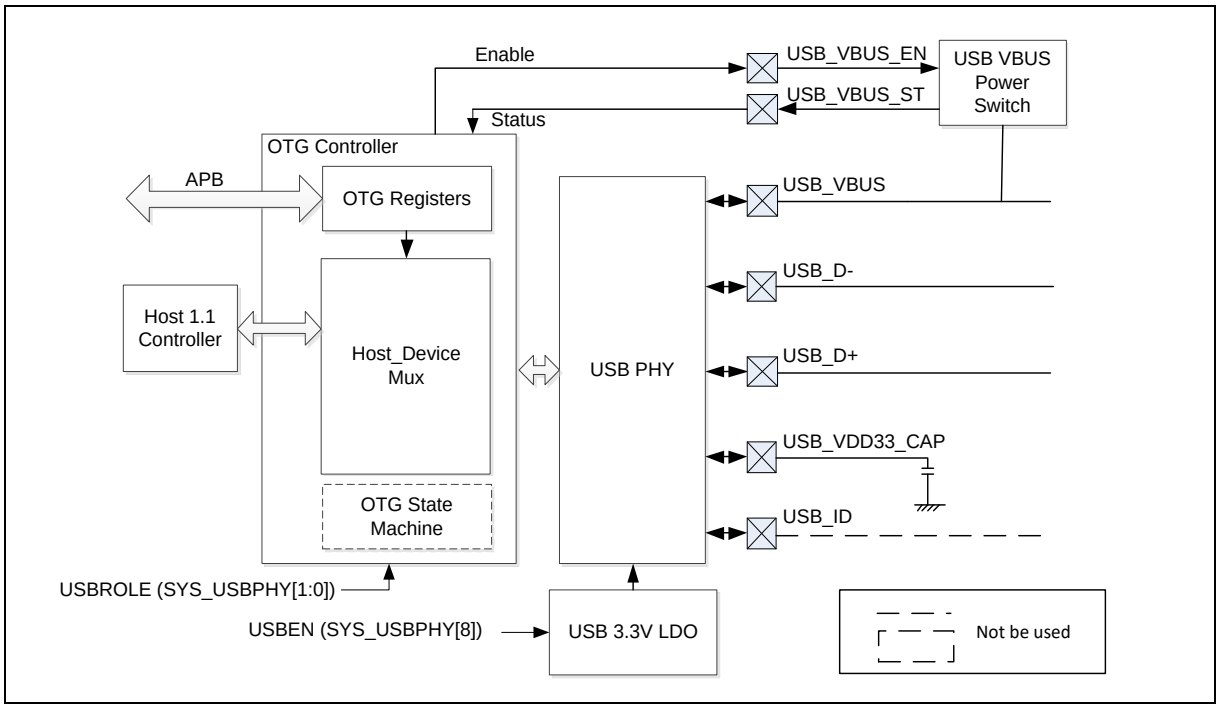


Figure 6.36-3 USB Host Mode

ID Dependent Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 2, the role of USB frame depends on USB_ID pin state. The ID detection function can be enabled by setting IDDETEN (OTG_PHYCTL[1]) to 1. The USB_ID pin state reflects on IDSTS (OTG_STATUS[1]). When the USB frame acts as USB host (USB_ID pin is at low level), the function is the same as USB Host mode. When the USB frame acts as USB device (USB_ID pin is at high level), the function is the same as USB Device mode.

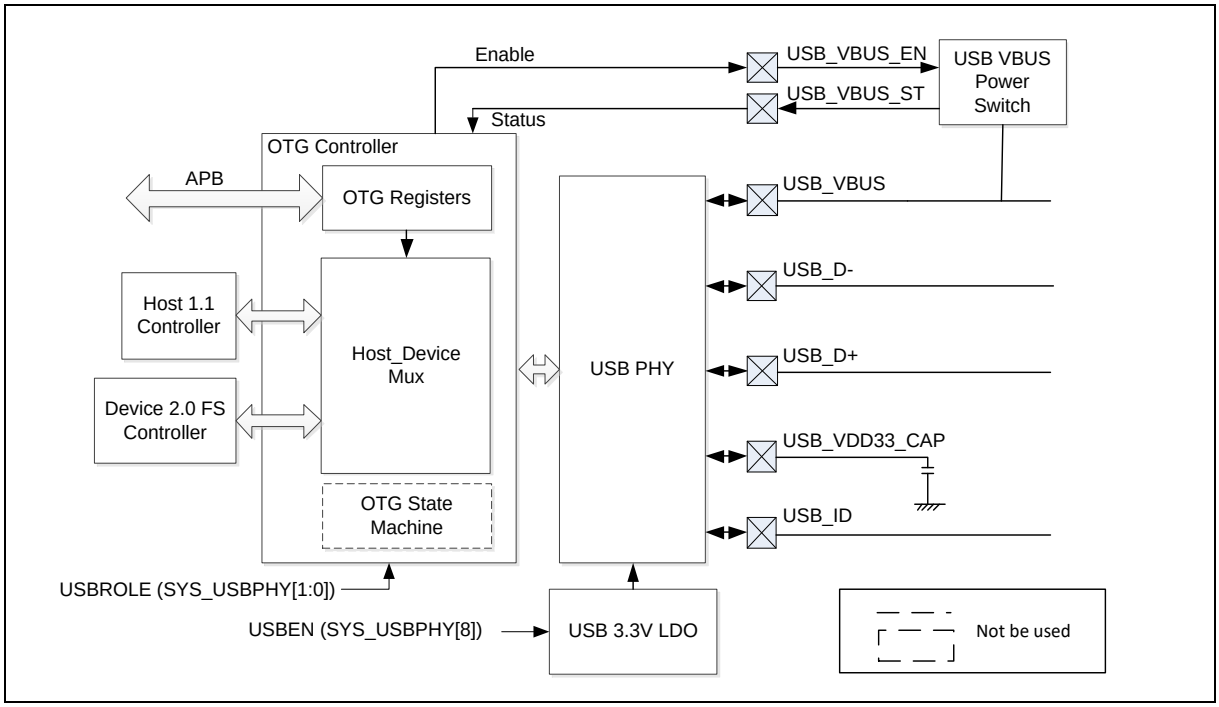


Figure 6.36-4 USB ID Dependent Mode

OTG Device Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 3, the role of USB frame depends on USB_ID pin state. The ID detection function can be enabled by setting IDDETEN (OTG_PHYCTL[1]) to 1. The USB_ID pin state reflects on IDSTS (OTG_STATUS[1]). When USB_ID pin state is at low level, the OTG controller acts as OTG A-device. When USB_ID pin state is at high level, the OTG controller acts as OTG B-device. Please refer to OTG specification to get detailed behavior of A-device and B-device.

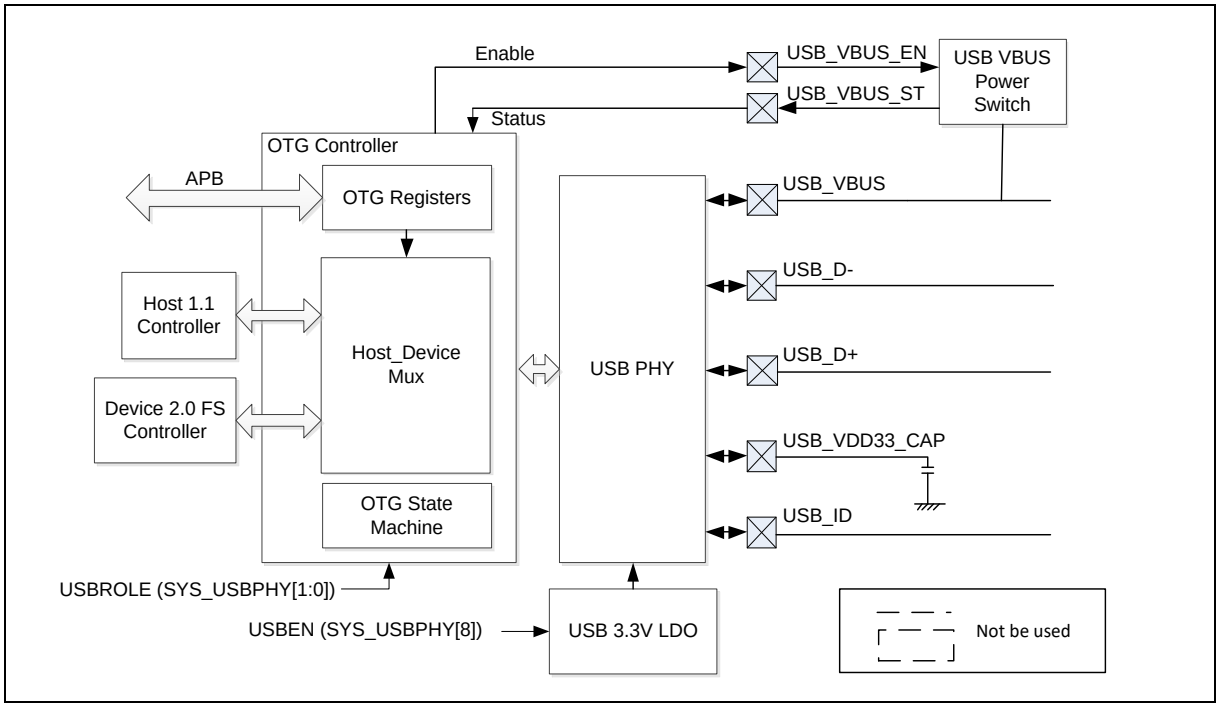


Figure 6.36-4 USB OTG Block Mode

6.36.5.2 Session Request Protocol (SRP)

When the USB frame is configured as OTG device mode, the OTG controller supports SRP to conserve power. Refer to OTG specification for details about SRP.

A-Device Session Request Protocol

1. A-device turns off USB bus power to conserve power. B-device recognizes such condition by checking V_{BUS} state.
2. B-device requests A-device to supply USB bus power by data line pulsing when B-device wants to connect to A-device.
3. A-device recognizes USB bus power request through checking SRPDETIF (OTG_INTSTS[13]).
4. A-device starts to drive V_{BUS} by setting BUSREQ (OTG_CTL[1]) to 1 once SRPDETIF (OTG_INTSTS[13]) is set to 1 by hardware. If V_{BUS} reaches valid level in specific time interval and B-device is connected, A-device will become USB host, HOSTIF (OTG_INTSTS[7]) will be set to 1. If V_{BUS} cannot reach valid level in specific time interval, it means overcurrent condition occurs. Then V_{BUS} error bit, VBEIF (OTG_INTSTS[1]), will be set to 1.

B-device Session Request Protocol

1. A-device turns off USB bus power to conserve power. B-device recognizes such condition by checking VBUSVLD (OTG_STATUS[5]).
2. B-device can request A-device to supply USB bus power by setting BUSREQ(OTG_CTL[1]) to 1.
3. B-device will generate data line pulsing as defined in OTG specification.
4. A-device will start to drive V_{BUS} after detecting data line pulsing and B-device can recognize such condition by checking VBUSVLD (OTG_STATUS[5]). If A-device drives V_{BUS} to valid level in specific time interval, B-device becomes USB peripheral, PDEVIF (OTG_INTSTS[6])

will be set to 1. If A-device does not drive V_{BUS} to valid level in specific time interval, SRP failure flag, SRPFIF (OTG_INTSTS[2]), will be set to 1 and B-device will go to idle state defined in OTG specification.

6.36.5.3 Host Negotiation Protocol (HNP)

When the USB frame is configured as OTG device mode, the host function can be transferred between two directly connected OTG device without changing the cable connection. Refer to OTG specification for details about HNP.

A-device Host Negotiation Protocol

1. A-Host defined in OTG specification sends SetFeature b_hnp_enable command to enable B-device HNP capability. B-device responses ACK to indicate B-device supports HNP. User needs to set HNPREQEN (OTG_CTL[2]) to 1 to enable HNP protocol.
2. A-Host goes to a_suspend state by setting BUSREQ (OTG_CTL[1]) to 0 and put USB bus into J-state (USB_D+ high and USB_D- low) when A-Host has finished all desired operations.
3. A-Host becomes A-Peripheral if A-Host detects B-peripheral disconnected by checking USB_D+ and USB_D- low in specific time interval. If A-Host cannot detect B-Peripheral disconnected in specific time interval, A-Host will back to idle state.

B-device Host Negotiation Protocol

1. After B-Peripheral receives SetFeature b_hnp_enable command successfully, user enables B-peripheral HNP function by setting HNPREQEN (OTG_CTL[2]) to 1.
2. User sets BUSREQ (OTG_CTL[1]) to 1 after detecting USB bus in J-state(USB_D+ high and USB_D- low). Then USB_D+ pull high resistor will be removed to cause USB disconnect state (USB_D+ low and USB_D- low).
3. If B-device detects A-device is connected (USB_D+ high) in specific time interval, B-device will become B-Host. If B-device cannot detect A-device is connected (USB_D+ high) in specific time interval, HNP failure flag, HNPFIF (OTG_INTSTS[3]), will be set to 1.

6.36.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
OTG Base Address: **OTG_BA = 0x4004_D000**				
OTG_CTL	OTG_BA+0x00	R/W	OTG Control Register	0x0000_0000
OTG_PHYCTL	OTG_BA+0x04	R/W	OTG PHY Control Register	0x0000_0000
OTG_INTEN	OTG_BA+0x08	R/W	OTG Interrupt Enable Register	0x0000_0000
OTG_INTSTS	OTG_BA+0x0C	R/W	OTG Interrupt Status Register	0x0000_0XX0
OTG_STATUS	OTG_BA+0x10	R	OTG Status Register	0x0000_000X

6.36.7 Register Description

OTG Control Register (OTG_CTL)

Register	Offset	R/W	Description	Reset Value
OTG_CTL	OTG_BA+0x00	R/W	OTG Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**WKEN**	**OTGEN**	**Reserved**	**HNPREQEN**	**BUSREQ**	**VBUSDROP**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**WKEN**	**OTG ID Pin Wake-up Enable Bit** 0 = OTG ID pin status change wake-up function Disabled. 1 = OTG ID pin status change wake-up function Enabled.
[4]	**OTGEN**	**OTG Function Enable Bit** User needs to set this bit to enable OTG function while the USB frame configured as OTG device. When the USB frame is not configured as OTG device, this bit must be low. 0 = OTG function Disabled. 1 = OTG function Enabled.
[3]	**Reserved**	Reserved.
[2]	**HNPREQEN**	**OTG HNP Request Enable Bit** When the USB frame acts as A-device, set this bit when A-device allows to process HNP protocol—A-device changes role from Host to Peripheral. This bit will be cleared when OTG state changes from a_suspend to a_peripheral or goes back to a_idle state. When the USB frame acts as B-device, set this bit after the OTG A-device successfully sends a SetFeature (b_hnp_enable) command to the OTG B-device to start role change — B-device changes role from Peripheral to Host. This bit will be cleared when OTG state changes from b_peripheral to b_wait_acon or goes back to b_idle state. 0 = HNP request Disabled. 1 = HNP request Enabled (A-device can change role from Host to Peripheral or B-device can change role from Peripheral to Host). **Note:** Refer to OTG specification to get a_suspend, a_peripheral, a_idle and b_idle state.
[1]	**BUSREQ**	**OTG Bus Request** If OTG A-device wants to do data transfers via USB bus, setting this bit will drive V_{BUS} high to detect USB device connection. If user won't use the bus any more, clearing this bit will drop V_{BUS} to save power. This bit will be cleared when A-device goes to A_wait_vfall state. This bit will be also cleared if VBUSDROP (OTG_CTL[0]) bit is set or IDSTS (OTG_STATUS[1]) changed. If user of an OTG-B Device wants to request V_{BUS}, setting this bit will run SRP protocol.

Bits	Description	
		This bit will be cleared if SRP failure (OTG A-device does not provide V_{BUS} after B-device issues SRP in specified interval, defined in OTG specification). This bit will be also cleared if VBUSDROP (OTG_CTL[0]) bit is set or IDSTS (OTG_STATUS[1]) changed. 0 = Not launch V_{BUS} in OTG A-device or not request SRP in OTG B-device. 1 = Launch V_{BUS} in OTG A-device or request SRP in OTG B-device.
[0]	**VBUSDROP**	**Drop V_{BUS} Control** If user application running on this OTG A-device wants to conserve power, set this bit to drop V_{BUS}. BUSREQ (OTG_CTL[1]) will be also cleared no matter A-device or B-device. 0 = Not drop the V_{BUS}. 1 = Drop the V_{BUS}.

OTG PHY Control Register (OTG_PHYCTL)

Register	Offset	R/W	Description	Reset Value
OTG_PHYCTL	OTG_BA+0x04	R/W	OTG PHY Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		VBSTSPOL	VBENPOL	Reserved		IDDETEN	OTGPHYEN

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**VBSTSPOL**	**Off-chip USB V_{BUS} Power Switch Status Polarity** The polarity of off-chip USB V_{BUS} power switch valid signal depends on the selected component. A USB_VBUS_ST pin is used to monitor the valid signal of the off-chip USB V_{BUS} power switch. Set this bit as following according to the polarity of off-chip USB V_{BUS} power switch. 0 = The polarity of off-chip USB V_{BUS} power switch valid status is high. 1 = The polarity of off-chip USB V_{BUS} power switch valid status is low.
[4]	**VBENPOL**	**Off-chip USB V_{BUS} Power Switch Enable Polarity** The OTG controller will enable off-chip USB V_{BUS} power switch to provide V_{BUS} power when need. A USB_VBUS_EN pin is used to control the off-chip USB V_{BUS} power switch. The polarity of enabling off-chip USB V_{BUS} power switch (high active or low active) depends on the selected component. Set this bit as following according to the polarity of off-chip USB V_{BUS} power switch. 0 = The off-chip USB V_{BUS} power switch enable is active high. 1 = The off-chip USB V_{BUS} power switch enable is active low.
[3:2]	**Reserved**	Reserved.
[1]	**IDDETEN**	**ID Detection Enable Bit** 0 = Detect ID pin status Disabled. 1 = Detect ID pin status Enabled.
[0]	**OTGPHYEN**	**OTG PHY Enable Bit** When the USB frame is configured as either OTG device or ID dependent, user needs to set this bit before using OTG function. If device is configured as neither OTG device nor ID dependent, this bit is “don’t care”. 0 = OTG PHY Disabled. 1 = OTG PHY Enabled.

OTG Interrupt Enable Register (OTG_INTEN)

Register	Offset	R/W	Description	Reset Value
OTG_INTEN	OTG_BA+0x08	R/W	OTG Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SRPDETIEN	Reserved	SECHGIEN	VBCHGIEN	AVLDCHGIEN	BVLDCHGIEN
7	6	5	4	3	2	1	0
HOSTIEN	PDEVIEN	IDCHGIEN	GOIDLEIEN	HNPFIEN	SRPFIEN	VBEIEN	ROLECHGIEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SRPDETIEN**	**SRP Detected Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[12]	**Reserved**	Reserved.
[11]	**SECHGIEN**	**SESSEND Status Changed Interrupt Enable Bit** If this bit is set to 1 and SESSEND (OTG_STATUS[2]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[10]	**VBCHGIEN**	**VBUSVLD Status Changed Interrupt Enable Bit** If this bit is set to 1 and VBUSVLD (OTG_STATUS[5]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[9]	**AVLDCHGIEN**	**A-device Session Valid Status Changed Interrupt Enable Bit** If this bit is set to 1 and AVLD (OTG_STATUS[4]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[8]	**BVLDCHGIEN**	**B-device Session Valid Status Changed Interrupt Enable Bit** If this bit is set to 1 and BVLD (OTG_STATUS[3]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[7]	**HOSTIEN**	**Act As Host Interrupt Enable Bit** If this bit is set to 1 and the device is changed as a host, an interrupt will be asserted.

Bits	Description	
		0 = This device as a host interrupt Disabled. 1 = This device as a host interrupt Enabled.
[6]	**PDEVIEN**	**Act As Peripheral Interrupt Enable Bit** If this bit is set to 1 and the device is changed as a peripheral, an interrupt will be asserted. 0 = This device as a peripheral interrupt Disabled. 1 = This device as a peripheral interrupt Enabled.
[5]	**IDCHGIEN**	**IDSTS Changed Interrupt Enable Bit** If this bit is set to 1 and IDSTS (OTG_STATUS[1]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[4]	**GOIDLEIEN**	**OTG Device Going to IDLE State Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Going to idle state means going to a_idle or b_idle state. Please refer to A-device state diagram and B-device state diagram in OTG specification.
[3]	**HNPFIEN**	**HNP Fail Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[2]	**SRPFIEN**	**SRP Fail Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[1]	**VBEIEN**	**V_{BUS} Error Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** V_{BUS} error means going to a_vbus_err state. Please refer to A-device state diagram in OTG specification.
[0]	**ROLECHGIEN**	**Role Changed Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note: Role is Host or Peripheral.**

OTG Interrupt Status Register (OTG_INTSTS)

Register	Offset	R/W	Description	Reset Value
OTG_INTSTS	OTG_BA+0x0C	R/W	OTG Interrupt Status Register	0x0000_0XX0

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SRPDETIF	Reserved	SECHGIF	VBCHGIF	AVLDCHGIF	BVLDCHGIF
7	6	5	4	3	2	1	0
HOSTIF	PDEVIF	IDCHGIF	GOIDLEIF	HNPFIF	SRPFIF	VBEIF	ROLECHGIF

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SRPDETIF**	**SRP Detected Interrupt Status** 0 = SRP not detected. 1 = SRP detected. **Note:** Write 1 to clear this status.
[12]	**Reserved**	**Note:**
[11]	**SECHGIF**	**SESSEND State Change Interrupt Status** 0 = SESSEND (OTG_STATUS[2]) not toggled. 1 = SESSEND (OTG_STATUS[2]) from high to low or from low to high. **Note:** Write 1 to clear this flag.
[10]	**VBCHGIF**	**VBUSVLD State Change Interrupt Status** 0 = VBUSVLD (OTG_STATUS[5]) not toggled. 1 = VBUSVLD (OTG_STATUS[5]) from high to low or from low to high. **Note:** Write 1 to clear this status.
[9]	**AVLDCHGIF**	**A-device Session Valid State Change Interrupt Status** 0 = AVLD (OTG_STATUS[4]) not toggled. 1 = AVLD (OTG_STATUS[4]) from high to low or low to high. **Note:** Write 1 to clear this status.
[8]	**BVLDCHGIF**	**B-device Session Valid State Change Interrupt Status** 0 = BVLD (OTG_STATUS[3]) not toggled. 1 = BVLD (OTG_STATUS[3]) from high to low or low to high. **Note:** Write 1 to clear this status.
[7]	**HOSTIF**	**Act As Host Interrupt Status** 0= This device does not act as a host. 1 = This device acts as a host. **Note:** Write 1 to clear this flag.

Bits	Description	
[6]	**PDEVIF**	**Act As Peripheral Interrupt Status** 0 = This device does not act as a peripheral. 1 = This device acts as a peripheral. **Note:** Write 1 to clear this flag.
[5]	**IDCHGIF**	**ID State Change Interrupt Status** 0 = IDSTS (OTG_STATUS[1]) not toggled. 1 = IDSTS (OTG_STATUS[1]) from high to low or from low to high. **Note:** Write 1 to clear this flag.
[4]	**GOIDLEIF**	**OTG Device Goes to IDLE Interrupt Status** Flag is set if the OTG device transfers from non-idle state to idle state. The OTG device will be neither a host nor a peripheral. 0 = OTG device does not go back to idle state (a_idle or b_idle). 1 = OTG device goes back to idle state(a_idle or b_idle). **Note 1:** Going to idle state means going to a_idle or b_idle state. Please refer to OTG specification. **Note 2:** Write 1 to clear this flag.
[3]	**HNPFIF**	**HNP Fail Interrupt Status** When A-device has granted B-device to be host and USB bus is in SE0 (both USB_D+ and USB_D- low) state, this bit will be set when A-device does not connect after specified interval expires. 0 = A-device connects to B-device before specified interval expires. 1 = A-device does not connect to B-device before specified interval expires. **Note:** Write 1 to clear this flag.
[2]	**SRPFIF**	**SRP Fail Interrupt Status** After initiating SRP, an OTG B-device will wait for the OTG A-device to drive V_{BUS} high at least TB_SRP_FAIL minimum, defined in OTG specification. This flag is set when the OTG B-device does not get V_{BUS} high after this interval. 0 = OTG B-device gets V_{BUS} high before this interval. 1 = OTG B-device does not get V_{BUS} high before this interval. **Note:** Write 1 to clear this flag.
[1]	**VBEIF**	**V_{BUS} Error Interrupt Status** This bit will be set when voltage on V_{BUS} cannot reach a minimum valid threshold 4.4V within a maximum time of 100ms after OTG A-device starting to drive V_{BUS} high. 0 = OTG A-device drives V_{BUS} over threshold voltage before this interval expires. 1 = OTG A-device cannot drive V_{BUS} over threshold voltage before this interval expires. **Note:** Write 1 to clear this flag and recover from the V_{BUS} error state.
[0]	**ROLECHGIF**	**OTG Role Change Interrupt Status** This flag is set when the role of an OTG device changed from a host to a peripheral, or changed from a peripheral to a host while USB_ID pin status does not change. 0 = OTG device role not changed. 1 = OTG device role changed. **Note:** Write 1 to clear this flag.

OTG Functional Status Register (OTG_STATUS)

Register	Offset	R/W	Description	Reset Value
OTG_STATUS	OTG_BA+0x10	R	OTG Status Register	0x0000_000X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
ASHOST	ASPERI	VBUSVLD	AVLD	BVLD	SESSEND	IDSTS	OVERCUR

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**ASHOST**	**As Host Status** When OTG acts as Host, this bit is set. 0 = OTG not as Host. 1 = OTG as Host.
[6]	**ASPERI**	**As Peripheral Status** When OTG acts as peripheral, this bit is set. 0 = OTG not as peripheral. 1 = OTG as peripheral.
[5]	**VBUSVLD**	**V_{BUS} Valid Status** When V_{BUS} is larger than 4.7V, this bit will be set to 1. 0 = V_{BUS} is not valid. 1 = V_{BUS} is valid.
[4]	**AVLD**	**A-Device Session Valid Status** 0 = A-device session is not valid. 1 = A-device session is valid.
[3]	**BVLD**	**B-device Session Valid Status** 0 = B-device session is not valid. 1 = B-device session is valid.
[2]	**SESSEND**	**Session End Status** When V_{BUS} voltage is lower than 0.4V, this bit will be set to 1. Session end means no meaningful power on V_{BUS}. 0 = Session is not end. 1 = Session is end.
[1]	**IDSTS**	**USB_ID Pin State of Mini-/Micro-Plug** 0 = Mini-A/Micro-A plug is attached.

Bits	Description	
		1 = Mini-B/Micro-B plug is attached.
[0]	**OVERCUR**	**Overcurrent Condition** The voltage on V_{BUS} cannot reach a minimum V_{BUS} valid threshold, 4.4V minimum, within a maximum time of 100ms after OTG A-device drives V_{BUS} high. 0 = OTG A-device drives V_{BUS} successfully. 1 = OTG A-device cannot drives V_{BUS} high in this interval.

6.37 High Speed USB 2.0 On-The-Go (HSOTG)

6.37.1 Overview

The HSOTG controller interfaces to USB PHY and USB controllers which consist of a USB 2.0 host controller and a USB 2.0 HS device controller. The OTG controller supports HNP and SRP protocols defined in the “On-The-Go and Embedded Host Supplement to the USB 2.0 Revision 1.3 Specification”.

USB frame, including USB host, USB device, and OTG controller, can be configured as Host-only, Device-only, ID dependent or OTG device mode defined in HSUSBROLE (SYS_USBPHY[17:16]). In Host-only mode, USB frame acts as USB host. USB frame can support high-speed, full-speed and low-speed transfer. In Device-only mode, USB frame acts as USB device. USB frame supports high-speed and full-speed transfer. In ID dependent mode, USB frame can be USB Host or USB device depends on USB_ID pin state. In OTG device mode, the role of USB frame depends on the definition of OTG specification. USB frame supports high-speed and full-speed transfer when OTG device acts as a peripheral.

6.37.2 Features

- Built in USB PHY
- Configurable to operate as:
  - Host-only
  - Device-only
  - ID dependent: The USB frame can act as USB Host (USB_ID pin is at low) or USB Device (USB_ID pin is at high) only depending on USB_ID pin state. HNP or SRP protocol is not supported.
  - OTG device: It can be A-device (USB_ID pin state is at low) or B-device (USB_ID pin state is at high) depending on USB_ID pin state. HNP and SRP protocols are supported.

6.37.3 Block Diagram

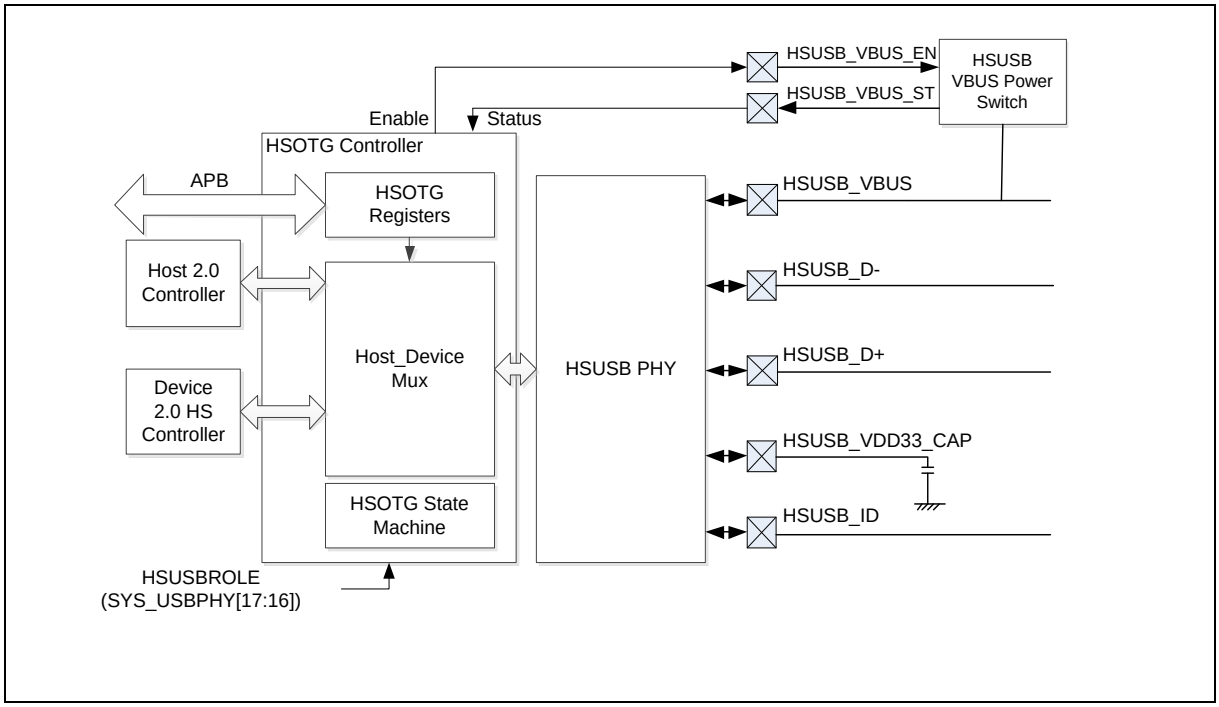


Figure 6.37-1 HS USB OTG Block Diagram

6.37.4 Basic Configuration

The OTG peripheral clock can be enabled by HSOTGCKEN (CLK_APBCLK0[30]). The role of USB frame is determined by HSUSBROLE (SYS_USBPHY[17:16]). These two configurations bits are write-protection. Before writing to these bits, user must disable the register protection function. Refer to the description of SYS_REGLCTL register for details. The function of HSUSB_VBUS_EN and HSUSB_VBUS_ST pins are configured in SYS_GPB_MFP2, SYS_GPB_MFP3, SYS_GPC_MFP3 or SYS_GPJ_MFP3 registers.

- Reset Configuration
  - Reset the HSOTG in HSOTGRST (SYS_IPRST1[30]).
- Clock Source Configuration
  - Enable the OTG peripheral clock in HSOTGCKEN (CLK_APBCLK0[30]).
- OTG Configuration
  - Refer to the description of SYS_REGLCTL register to disable the register protection function.
  - Set the role of USB frame is in HSUSBROLE (SYS_USBPHY[17:16]).
  - Enable PHY in HSUSBEN (SYS_USBPHY[24]).
  - Write 1 to clear all interrupt status(HSOTG_INTSTS)

6.37.5 Functional Description

The role of USB frame depends on the setting of HSUSBROLE (SYS_USBPHY[17:16]) and USB_ID pin state. The HSUSBROLE configuration has precedence over USB_ID pin state. User can configure the OTG controller to USB Host mode, USB Device mode, ID dependent mode or OTG device mode. In USB Host mode, the host controller will interact with USB PHY directly. In USB Device mode, the device controller will interact with USB PHY directly. In these cases, the OTG controller is used simply as a multiplexer. In ID dependent mode, USB_ID pin state will decide USB frame to act as USB host or

USB device. If the USB_ID pin is FALSE state (low level), USB frame will act as USB host. If the USB_ID pin is TRUE state (high level), USB frame will act as USB device. In OTG device mode, the OTG controller will handle OTG HNP and SRP protocols. If the USB_ID pin is FALSE state (low level), the OTG controller will act as an OTG A-device. If the USB_ID pin is TRUE state (high level), the OTG controller will act as an OTG B-device.

6.37.5.1 The Role of USB Frame

USB Device Mode

When HSUSBROLE (SYS_USBPHY[17:16]) is set to 0, USB frame acts as USB device. USB host function is not available.

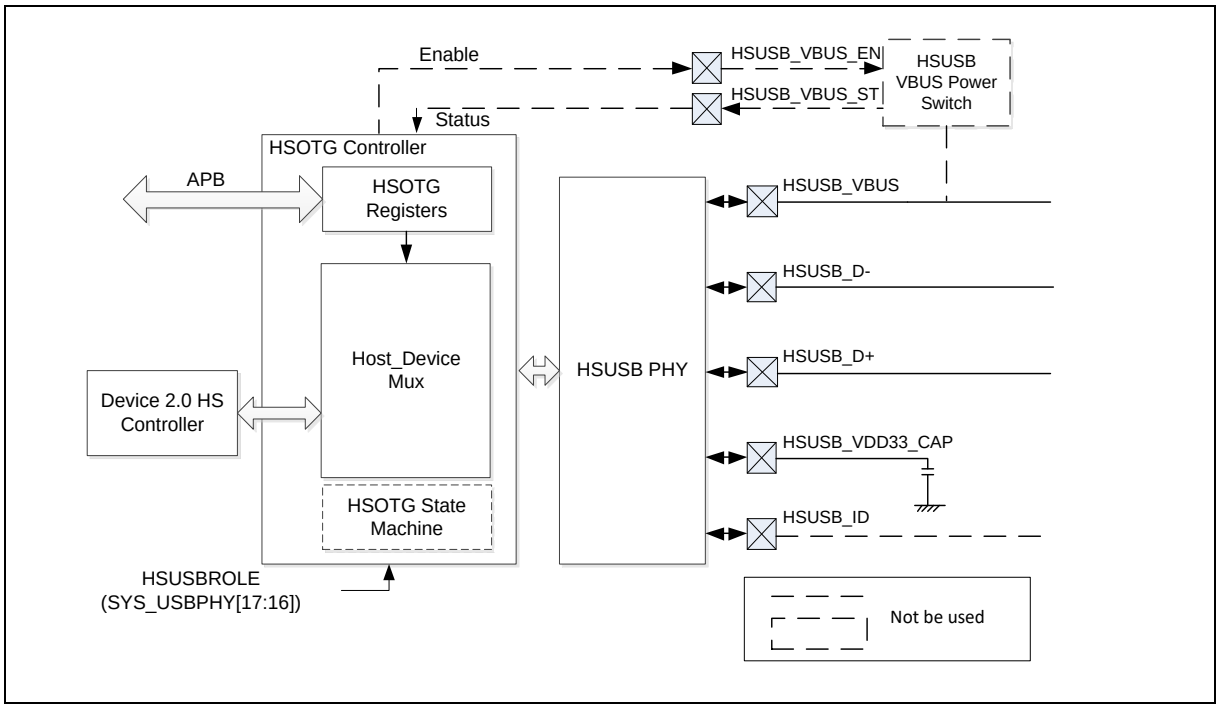


Figure 6.37-2 USB Device Mode

USB Host Mode

When HSUSBROLE (SYS_USBPHY[17:16]) is set to 1, USB frame acts as USB host. USB device function is not available.

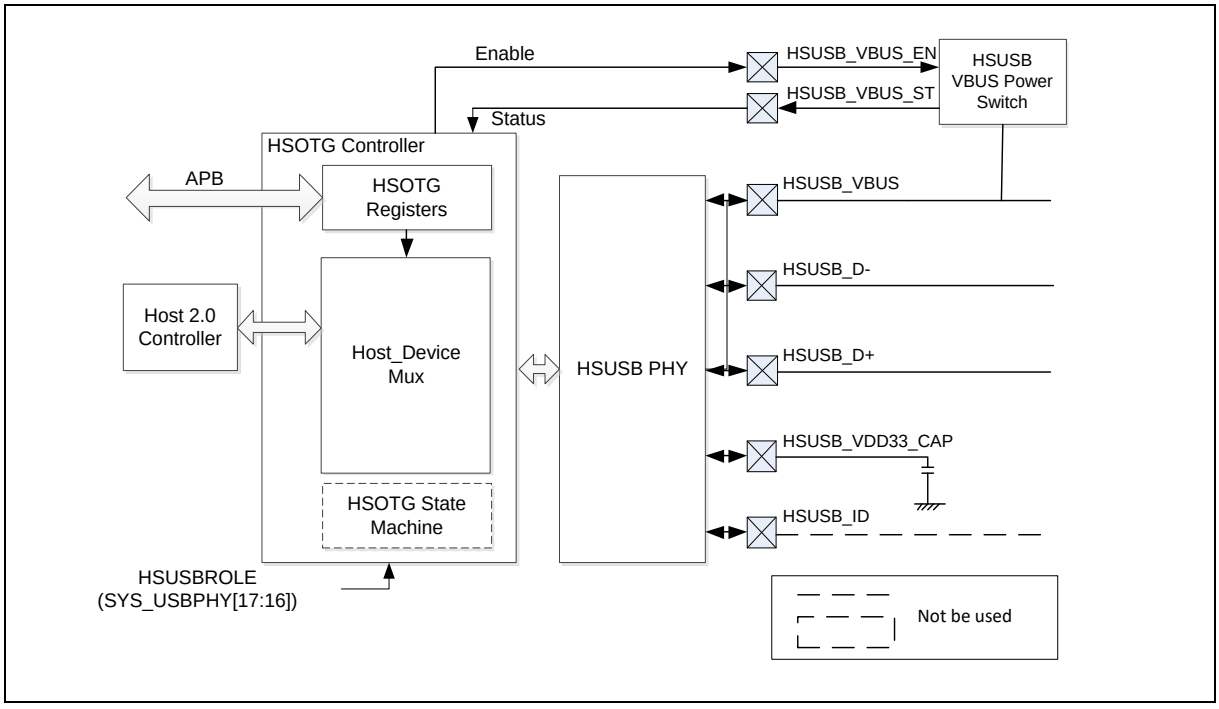


Figure 6.37-3 USB Host Mode

ID Dependent Mode

When HSUSBROLE (SYS_USBPHY[17:16]) is set to 2, the role of USB frame depends on USB_ID pin state. The ID detection function can be enabled by set IDDETEN (HSOTG_PHYCTL[1]) to 1. The USB_ID pin state reflects on IDSTS (HSOTG_STATUS[1]). When USB frame acts as USB host (USB_ID pin is at low level), the block diagram is the same as USB Host mode. When USB frame acts as USB device (USB_ID pin is at high level), the block diagram is the same as USB Device mode.

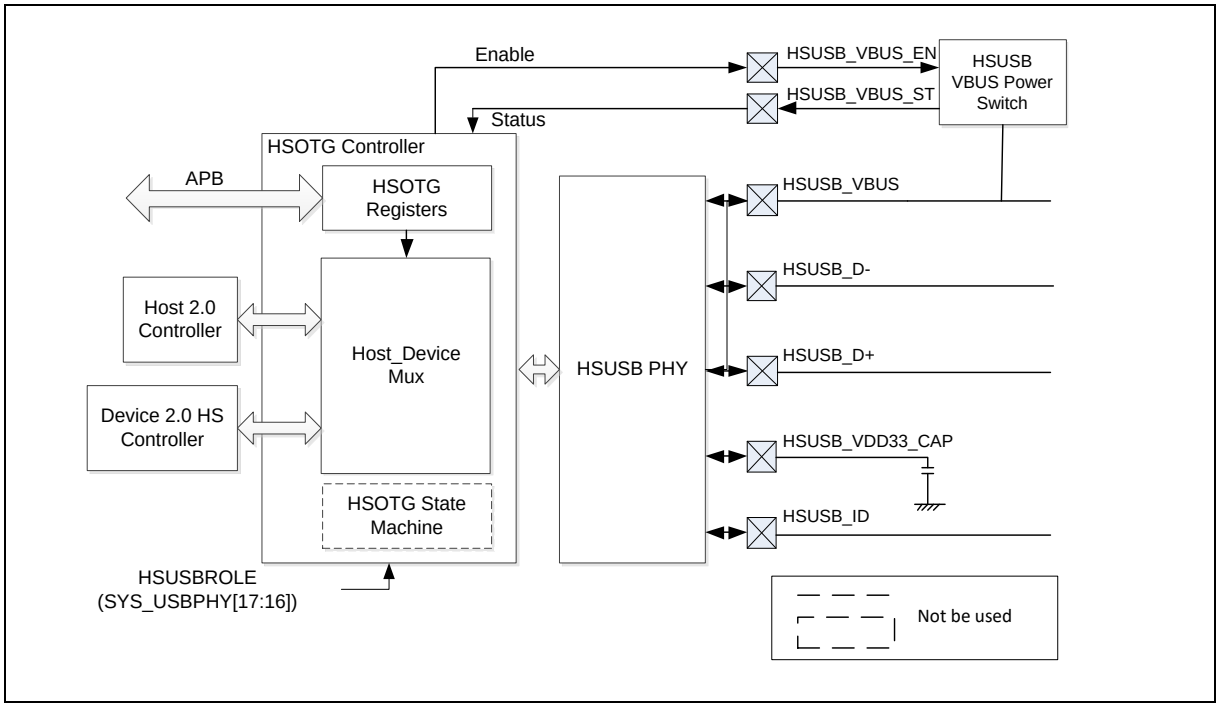


Figure 6.37-4 USB ID Dependent Mode

OTG device mode

When HSUSBROLE (SYS_USBPHY[17:16]) is set to 3, the role of USB frame depends on USB_ID pin state. The ID detection function can be enabled by setting IDDETEN (HSOTG_PHYCTL[1]) to 1. The USB_ID pin state reflects on IDSTS (HSOTG_STATUS[1]). When USB_ID pin state is at low level, the OTG controller acts as OTG A-device. When USB_ID pin state is at high level, the OTG controller acts as OTG B-device. Please refer to OTG specification to get detailed behavior of A-device and B-device.

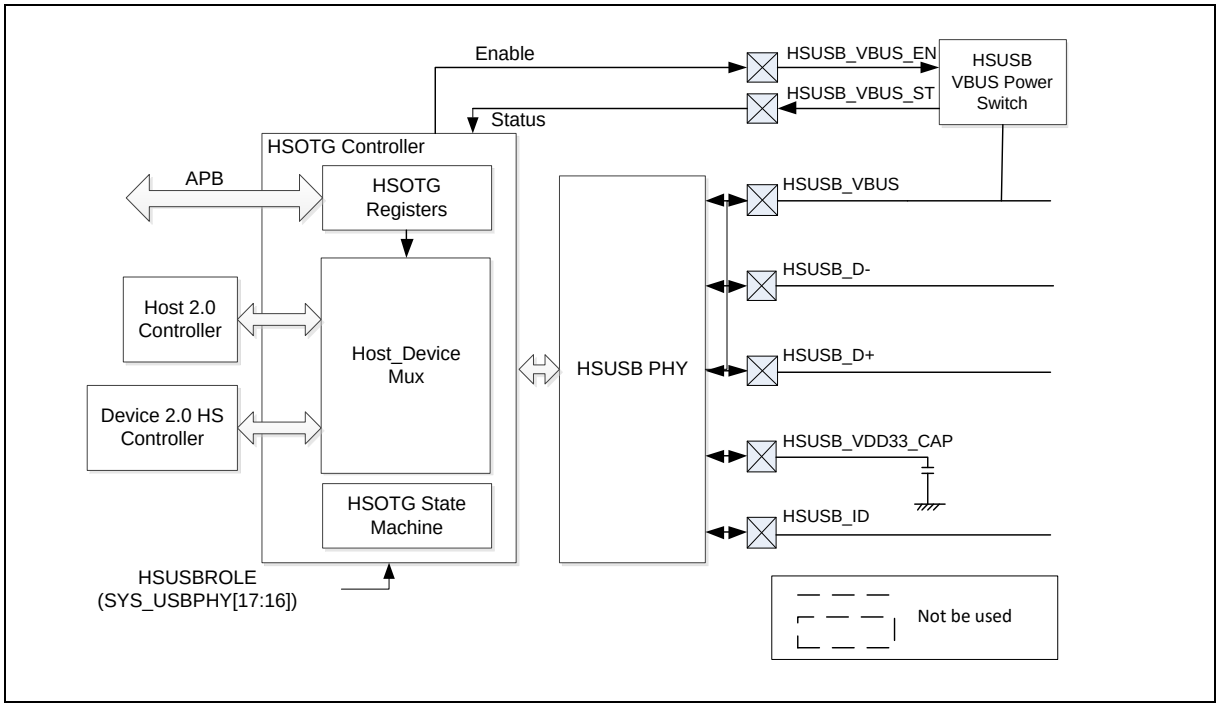


Figure 6.37-4 USB OTG Block Mode

6.37.5.2 Session Request Protocol (SRP)

When the USB frame is configured as OTG device mode, OTG controller supports SRP to conserve power. Refer to OTG specification for details about SRP.

A-Device Session Request Protocol

1. A-device turns off USB bus power to conserve power. B-device recognizes such condition by checking V_{BUS} status.
2. B-device requests A-device to supply USB bus power by Data line pulsing when B-device wants to connect to A-device.
3. A-device recognizes USB bus power request through checking SRPDETIF (HSOTG_INTSTS[13]).
4. A-device starts to drive V_{BUS} by setting BUSREQ (HSOTG_CTL[1]) to 1 once SRPDETIF (HSOTG_INTSTS[13]) is set to 1 by hardware. If V_{BUS} reaches valid level in specific time interval and B-device is connected, A-device will become USB host, HOSTIF (HSOTG_INTSTS[7]) will be set to 1. If V_{BUS} cannot reach valid level in specific time interval, it means overcurrent condition occurs. Then V_{BUS} error bit, VBEIF (HSOTG_INSTS[1]), will be set to 1.

B-device Session Request Protocol

1. A-device turns off USB bus power to conserve power. B-device recognizes such condition by checking VBUSVLD (HSOTG_STATUS[5]).
2. B-device can request A-device to supply USB bus power by setting BUSREQ (HSOTG_CTL[1]) to 1.
3. B-device will generate data line pulsing as defined in OTG specification.
4. A-device will start to drive V_{BUS} after detecting data line pulsing and B-device can recognize such condition by checking VBUSVLD (HSOTG_STATUS[5]). If A-device drives V_{BUS} to valid level in specific time interval, B-device becomes USB peripheral, PDEVIF (HSOTG_INTSTS[6]) will be set to 1. If A-device does not drive V_{BUS} to valid level in specific time interval, SRP failure flag, SRPFIF (HSOTG_INTSTS[2]), will be set to 1 and B-device will go to idle state defined in OTG specification.

6.37.5.3 Host Negotiation Protocol (HNP)

When the USB frame is configured as OTG device mode, the host function can be transferred between two directly connected OTG device without changing the cable connection. Refer to OTG specification for details about HNP.

A-device Host Negotiation Protocol

1. A-Host defined in OTG specification sends SetFeature b_hnp_enable command to enable B-device HNP capability. B-device responses ACK to indicate B-device supports HNP. User needs to set HNPREQEN (HSOTG_CTL[2]) to 1 to enable HNP protocol.
2. A-Host goes to a_suspend state by setting BUSREQ (HSOTG_CTL[1]) to 0 and put USB bus into J-state (HSUSB_D+ high and HSUSB_D- low) when A-Host has finished all desired operations.
3. A-Host becomes A-Peripheral if A-Host detects B-peripheral dis-connected by checking HSUSB_D+ and HSUSB_D- low in specific time interval. If A-Host cannot detect B-Peripheral dis-connected in specific time interval, A-Host will back to idle state.

B-device Host Negotiation Protocol

1. After B-Peripheral receives SetFeature b_hnp_enable command successfully, user enables B-peripheral HNP function by setting HNPREQEN (HSOTG_CTL[2]) to 1.
2. User sets BUSREQ (HSOTG_CTL[1]) to 1 after detecting USB bus in J-state (HSUSB_D+ high and HSUSB_D- low). Then HSUSB_D+ pull high resistor will be removed to cause USB disconnect state (HSUSB_D+ low and HSUSB_D- low).
3. If B-device detects A-device is connected (HSUSB_D+ high) in specific time interval, B-device will become B-Host. If B-device cannot detect A-device is connected (HSUSB_D+ high) in specific time interval, HNP failure flag, HNPFIF (HSOTG_INTSTS[3]), will be set to 1.

6.37.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
HSOTG Base Address: **HSOTG_BA = 0x4004_F000**				
HSOTG_CTL	HSOTG_BA+0x00	R/W	HSOTG Control Register	0x0000_0000
HSOTG_PHYCTL	HSOTG_BA+0x04	R/W	HSOTG PHY Control Register	0x0000_0200
HSOTG_INTEN	HSOTG_BA+0x08	R/W	HSOTG Interrupt Enable Register	0x0000_0000
HSOTG_INTSTS	HSOTG_BA+0x0C	R/W	HSOTG Interrupt Status Register	0x0000_0xx0
HSOTG_STATUS	HSOTG_BA+0x10	R	HSOTG Status Register	0x0000_000x

6.37.7 Register Description

HSOTG Control Register (HSOTG_CTL)

Register	Offset	R/W	Description	Reset Value
HSOTG_CTL	HSOTG_BA+0x00	R/W	HSOTG Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**WKEN**	**OTGEN**	**Reserved**	**HNPREQEN**	**BUSREQ**	**VBUSDROP**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**WKEN**	**OTG ID Pin Wake-up Enable Bit** 0 = OTG ID pin status change wake-up function Disabled. 1 = OTG ID pin status change wake-up function Enabled.
[4]	**OTGEN**	**OTG Function Enable Bit** User needs to set this bit to enable OTG function while the USB frame configured as OTG device. When the USB frame is not configured as OTG device, this bit is must be low. 0 = OTG function Disabled. 1 = OTG function Enabled.
[3]	**Reserved**	Reserved.
[2]	**HNPREQEN**	**OTG HNP Request Enable Bit** When the USB frame acts as A-device, set this bit when A-device allows to process HNP protocol—A-device changes role from Host to Peripheral. This bit will be cleared when OTG state changes from a_suspend to a_peripheral or goes back to a_idle state. When the USB frame acts as B-device, set this bit after the OTG A-device successfully sends a SetFeature (b_hnp_enable) command to the OTG B-device to start role change—B-device changes role from Peripheral to Host. This bit will be cleared when OTG state changes from b_peripheral to b_wait_acon or goes back to b_idle state. 0 = HNP request Disabled. 1 = HNP request Enabled (A-device can change role from Host to Peripheral or B-device can change role from Peripheral to Host). **Note:** Refer to OTG specification to get a_suspend, a_peripheral, a_idle and b_idle state.
[1]	**BUSREQ**	**OTG Bus Request** If OTG A-device wants to do data transfers via USB bus, setting this bit will drive V_{BUS} high to detect USB device connection. If user won't use the bus any more, clearing this bit will drop V_{BUS} to save power. This bit will be cleared when A-device goes to A_wait_vfall state. This bit will be also cleared if VBUSDROP (HSOTG_CTL[0]) bit is set or IDSTS (HSOTG_STATUS[1]) changed.

Bits	Description	
		If user of an OTG-B Device wants to request V_{BUS}, setting this bit will run SRP protocol. This bit will be cleared if SRP failure (OTG A-device does not provide V_{BUS} after B-device issues SRP in specified interval, defined in OTG specification). This bit will be also cleared if VBUSDROP (HSOTG_CTL[0]) bit is set or IDSTS (HSOTG_STATUS[1]) changed. 0 = Not launch V_{BUS} in OTG A-device or not request SRP in OTG B-device. 1 = Launch V_{BUS} in OTG A-device or request SRP in OTG B-device.
[0]	**VBUSDROP**	**Drop V_{BUS} Control** If user application running on this OTG A-device wants to conserve power, set this bit to drop V_{BUS}. BUSREQ (HSOTG_CTL[1]) will be also cleared no matter A-device or B-device. 0 = Not drop the V_{BUS}. 1 = Drop the V_{BUS}.

HSOTG PHY Control Register (HSOTG_PHYCTL)

Register	Offset	R/W	Description	Reset Value
HSOTG_PHYCTL	HSOTG_BA+0x04	R/W	HSOTG PHY Control Register	0x0000_0200

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					FSEL		
7	6	5	4	3	2	1	0
Reserved		VBSTSPOL	VBENPOL	Reserved		IDDETEN	OTGPHYEN

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:8]	**FSEL**	**Reference Clock Frequency Select** Selects OTG PHY reference clock frequency which is from HXT. 000 = Reference clock is 9.6 MHz . 001 = Reference clock is 10 MHz . 010 = Reference clock is 12 MHz . 011 = Reference clock is 19.2 MHz . 100 = Reference clock is 20 MHz . 101 = Reference clock is 24 MHz . 111 = Reference clock is 50 MHz .
[7:6]	**Reserved**	Reserved.
[5]	**VBSTSPOL**	**Off-chip USB V_{BUS} Power Switch Status Polarity** The polarity of off-chip USB V_{BUS} power switch valid signal depends on the selected component. A USB_VBUS_ST pin is used to monitor the valid signal of the off-chip USB V_{BUS} power switch. Set this bit as following according to the polarity of off-chip USB V_{BUS} power switch. 0 = The polarity of off-chip USB V_{BUS} power switch valid status is high. 1 = The polarity of off-chip USB V_{BUS} power switch valid status is low.
[4]	**VBENPOL**	**Off-chip USB V_{BUS} Power Switch Enable Polarity** The OTG controller will enable off-chip USB V_{BUS} power switch to provide V_{BUS} power when need. A USB_VBUS_EN pin is used to control the off-chip USB V_{BUS} power switch. The polarity of enabling off-chip USB V_{BUS} power switch (high active or low active) depends on the selected component. Set this bit as following according to the polarity of off-chip USB V_{BUS} power switch. 0 = The off-chip USB V_{BUS} power switch enable is active high. 1 = The off-chip USB V_{BUS} power switch enable is active low.
[3:2]	**Reserved**	Reserved.
[1]	**IDDETEN**	**ID Detection Enable Bit**

Bits	Description	
		0 = Detect ID pin status Disabled. 1 = Detect ID pin status Enabled.
[0]	**OTGPHYEN**	**OTG PHY Enable Bit** When the USB frame is configured as either OTG device or ID dependent, user needs to set this bit before using OTG function. If device is configured as neither OTG device nor ID dependent, this bit is “don’t care”. 0 = OTG PHY Disabled. 1 = OTG PHY Enabled.

HSOTG Interrupt Enable Register (HSOTG_INTEN)

Register	Offset	R/W	Description	Reset Value
HSOTG_INTEN	HSOTG_BA+0x08	R/W	HSOTG Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SRPDETIEN	Reserved	SECHGIEN	VBCHGIEN	AVLDCHGIEN	BVLDCHGIEN
7	6	5	4	3	2	1	0
HOSTIEN	PDEVIEN	IDCHGIEN	GOIDLEIEN	HNPFIEN	SRPFIEN	VBEIEN	ROLECHGIEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SRPDETIEN**	**SRP Detected Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[12]	**Reserved**	Reserved.
[11]	**SECHGIEN**	**SESSEND Status Changed Interrupt Enable Bit** If this bit is set to 1 and SESSEND (HSOTG_STATUS[2]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[10]	**VBCHGIEN**	**VBUSVLD Status Changed Interrupt Enable Bit** If this bit is set to 1 and VBUSVLD (HSOTG_STATUS[5]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[9]	**AVLDCHGIEN**	**A-device Session Valid Status Changed Interrupt Enable Bit** If this bit is set to 1 and AVLD (HSOTG_STATUS[4]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[8]	**BVLDCHGIEN**	**B-device Session Valid Status Changed Interrupt Enable Bit** If this bit is set to 1 and BVLD (HSOTG_STATUS[3]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[7]	**HOSTIEN**	**Act As Host Interrupt Enable Bit** If this bit is set to 1 and the device is changed as a host, an interrupt will be asserted.

Bits	Description	
		0 = This device as a host interrupt Disabled. 1 = This device as a host interrupt Enabled.
[6]	**PDEVIEN**	**Act As Peripheral Interrupt Enable Bit** If this bit is set to 1 and the device is changed as a peripheral, an interrupt will be asserted. 0 = This device as a peripheral interrupt Disabled. 1 = This device as a peripheral interrupt Enabled.
[5]	**IDCHGIEN**	**IDSTS Changed Interrupt Enable Bit** If this bit is set to 1 and IDSTS (HSOTG_STATUS[1]) status is changed from high to low or from low to high, an interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[4]	**GOIDLEIEN**	**OTG Device Going to IDLE State Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Going to idle state means going to a_idle or b_idle state. Please refer to A-device state diagram and B-device state diagram in OTG specification.
[3]	**HNPFIEN**	**HNP Fail Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[2]	**SRPFIEN**	**SRP Fail Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[1]	**VBEIEN**	**V_{BUS} Error Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** V_{BUS} error means going to a_vbus_err state. Please refer to A-device state diagram in OTG specification.
[0]	**ROLECHGIEN**	**Role Changed Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** The role can be Host or Peripheral.

HSOTG Interrupt Status Register (HSOTG_INTSTS)

Register	Offset	R/W	Description	Reset Value
HSOTG_INTSTS	HSOTG_BA+0x0C	R/W	HSOTG Interrupt Status Register	0x0000_0xx0

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SRPDETIF	Reserved	SECHGIF	VBCHGIF	AVLDCHGIF	BVLDCHGIF
7	6	5	4	3	2	1	0
HOSTIF	PDEVIF	IDCHGIF	GOIDLEIF	HNPFIF	SRPFIF	VBEIF	ROLECHGIF

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SRPDETIF**	**SRP Detected Interrupt Status** 0 = SRP not detected. 1 = SRP detected. **Note:** Write 1 to clear this status.
[12]	**Reserved**	
[11]	**SECHGIF**	**SESSEND State Change Interrupt Status** 0 = SESSEND (HSOTG_STATUS[2]) not toggled. 1 = SESSEND (HSOTG_STATUS[2]) from high to low or from low to high. **Note:** Write 1 to clear this flag.
[10]	**VBCHGIF**	**VBUSVLD State Change Interrupt Status** 0 = VBUSVLD (HSOTG_STATUS[5]) not toggled. 1 = VBUSVLD (HSOTG_STATUS[5]) from high to low or from low to high. **Note:** Write 1 to clear this status.
[9]	**AVLDCHGIF**	**A-device Session Valid State Change Interrupt Status** 0 = AVLD (HSOTG_STATUS[4]) not toggled. 1 = AVLD (HSOTG_STATUS[4]) from high to low or low to high. **Note:** Write 1 to clear this status.
[8]	**BVLDCHGIF**	**B-device Session Valid State Change Interrupt Status** 0 = BVLD (HSOTG_STATUS[3]) not toggled. 1 = BVLD (HSOTG_STATUS[3]) from high to low or low to high. **Note:** Write 1 to clear this status.
[7]	**HOSTIF**	**Act As Host Interrupt Status** 0 = This device does not act as a host. 1 = This device acts as a host. **Note:** Write 1 to clear this flag.

Bits	Description	
[6]	**PDEVIF**	**Act As Peripheral Interrupt Status** 0= This device does not act as a peripheral. 1 = This device acts as a peripheral. **Note:** Write 1 to clear this flag.
[5]	**IDCHGIF**	**ID State Change Interrupt Status** 0 = IDSTS (HSOTG_STATUS[1]) not toggled. 1 = IDSTS (HSOTG_STATUS[1]) from high to low or from low to high. **Note:** Write 1 to clear this flag.
[4]	**GOIDLEIF**	**OTG Device Goes to IDLE Interrupt Status** Flag is set if the OTG device transfers from non-idle state to idle state. The OTG device will be neither a host nor a peripheral. 0 = OTG device does not go back to idle state (a_idle or b_idle). 1 = OTG device goes back to idle state(a_idle or b_idle). **Note 1:** Going to idle state means going to a_idle or b_idle state. Please refer to OTG specification. **Note 2:** Write 1 to clear this flag.
[3]	**HNPFIF**	**HNP Fail Interrupt Status** When A-device has granted B-device to be host and USB bus is in SE0 (both HSUSB_D+ and HSUSB_D- low) state, this bit will be set when A-device does not connect after specified interval expires. 0 = A-device connects to B-device before specified interval expires. 1 = A-device does not connect to B-device before specified interval expires. **Note:** Write 1 to clear this flag.
[2]	**SRPFIF**	**SRP Fail Interrupt Status** After initiating SRP, an OTG B-device will wait for the OTG A-device to drive V_{BUS} high at least TB_SRP_FAIL minimum, defined in OTG specification. This flag is set when the OTG B-device does not get V_{BUS} high after this interval. 0 = OTG B-device gets V_{BUS} high before this interval. 1 = OTG B-device does not get V_{BUS} high before this interval. **Note:** Write 1 to clear this flag.
[1]	**VBEIF**	**V_{BUS} Error Interrupt Status** This bit will be set when voltage on V_{BUS} cannot reach a minimum valid threshold 4.4V within a maximum time of 100ms after OTG A-device starting to drive V_{BUS} high. 0 = OTG A-device drives V_{BUS} over threshold voltage before this interval expires. 1 = OTG A-device cannot drive V_{BUS} over threshold voltage before this interval expires. **Note:** Write 1 to clear this flag and recover from the V_{BUS} error state.
[0]	**ROLECHGIF**	**OTG Role Change Interrupt Status** This flag is set when the role of an OTG device changed from a host to a peripheral, or changed from a peripheral to a host while USB_ID pin status does not change. 0 = OTG device role not changed. 1 = OTG device role changed. **Note:** Write 1 to clear this flag.

HSOTG Functional Status Register (HSOTG_STATUS)

Register	Offset	R/W	Description	Reset Value
HSOTG_STATUS	HSOTG_BA+0x10	R	HSOTG Status Register	0x0000_000x

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
ASHOST	ASPERI	VBUSVLD	AVLD	BVLD	SESSEND	IDSTS	OVERCUR

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**ASHOST**	**As Host Status** When OTG acts as Host, this bit is set. 0 = OTG not as Host. 1 = OTG as Host.
[6]	**ASPERI**	**As Peripheral Status** When OTG acts as peripheral, this bit is set. 0 = OTG not as peripheral. 1 = OTG as peripheral.
[5]	**VBUSVLD**	**V_{BUS} Valid Status** When V_{BUS} is larger than 4.7V, this bit will be set to 1. 0 = V_{BUS} is not valid. 1 = V_{BUS} is valid.
[4]	**AVLD**	**A-Device Session Valid Status** 0 = A-device session is not valid. 1 = A-device session is valid.
[3]	**BVLD**	**B-device Session Valid Status** 0 = B-device session is not valid. 1 = B-device session is valid.
[2]	**SESSEND**	**Session End Status** When V_{BUS} voltage is lower than 0.4V, this bit will be set to 1. Session end means no meaningful power on V_{BUS}. 0 = Session is not end. 1 = Session is end.
[1]	**IDSTS**	**USB_ID Pin State of Mini-/Micro-Plug** 0 = Mini-A/Micro-A plug is attached.

Bits	Description	
		1 = Mini-B/Micro-B plug is attached.
[0]	**OVERCUR**	**Overcurrent Condition** The voltage on V_{BUS} cannot reach a minimum V_{BUS} valid threshold, 4.4V minimum, within a maximum time of 100ms after OTG A-device drives V_{BUS} high. 0 = OTG A-device drives V_{BUS} successfully. 1 = OTG A-device cannot drive V_{BUS} high in this interval.

6.38 Controller Area Network with Flexible Data-Rate (CAN FD)

6.38.1 Overview

The CAN FD controller performs communication according to ISO 11898-1:2015 and need be connected to additional transceiver hardware for the physical layer.

The CAN FD controller consists of one CAN Core, Memory access control and arbiter, Tx Handler, Rx Handler, a shared Message RAM memory and a 32-bit AHB interface for control and configuration registers.

The message storage is intended to be a single-ported Message RAM outside of the CAN Core module. It is connected to the CAN Core via the memory control interface. The Message RAM implements filters, receive FIFOs, transmit event FIFOs and transmit FIFOs.

All functions concerning the handling of messages are implemented by the Rx Handler and the Tx Handler. The Rx Handler manages message acceptance filtering, the transfer of received messages from the CAN Core to the Message RAM as well as providing received message status information. The Tx Handler is responsible for the transfer of transmit messages from the Message RAM to the CAN Core as well as providing transmitted status information.

The controller's clock domain concept allows the separation among CAN Core clock and the AHB clock.

6.38.2 Features

- Conform with CAN protocol version 2.0 part A, B and ISO 11898-1: 2015
- CAN FD with up to 64 data bytes supported
- CAN Error Logging
- AUTOSAR support
- SAE J1939 support
- Improved acceptance filtering
- Two configurable Receive FIFOs
- Separate signaling on reception of High Priority Messages
- Configurable Transmit FIFO, Transmit Queue, Transmit Event FIFO
- Direct Message RAM access for CPU
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains (CAN Core clock and AHB clock)

6.38.3 Block Diagram

The controller interfaces with the AMBA AHB bus. Figure 6.38-1 shows the block diagram of the controller.

CAN Core

CAN Protocol Controller and Rx/Tx shift register. Handle all 2.0 part A, B and ISO 11898-1:2015 protocol functions. Supports 11-bit and 29-bit identifiers

Sync

Synchronizes signals from the AHB clock domain to the CAN Core clock domain and vice versa

Control and Configuration Registers

All registers and related configuration used to control and to configure the CAN Core.

Interrupt & power down control

The CAN FD controller provides two interrupt lines CANFDx_INT0 and CANFDx_INT1. By programming EINT0 and EINT1 bits in CANFD_ILE register, the interrupt lines can be enabled or disabled separately.

The CAN FD controller can be set into power down controlled by both CANnSTREQ (CLK_STOPREQ[3:0]), and CANnSTACK (CLK_STOPACK[3:0]) where n= 0,..,3.

Memory access control and Arbiter

The arbiter is responsible for the Message RAM access authority between the CPU and the interface of Tx/Rx Handler to an external 32-bit Message RAM.

Message RAM (SRAM)

Stores message objects and Identifier masks. The maximum Message RAM size is 1536 words.

Tx/Rx Message Handler

State Machine that controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM as well as the generation of interrupts as programmed in the Control and Configuration registers.

AHB Interface

Connect the CAN FD controller to the AHB bus.

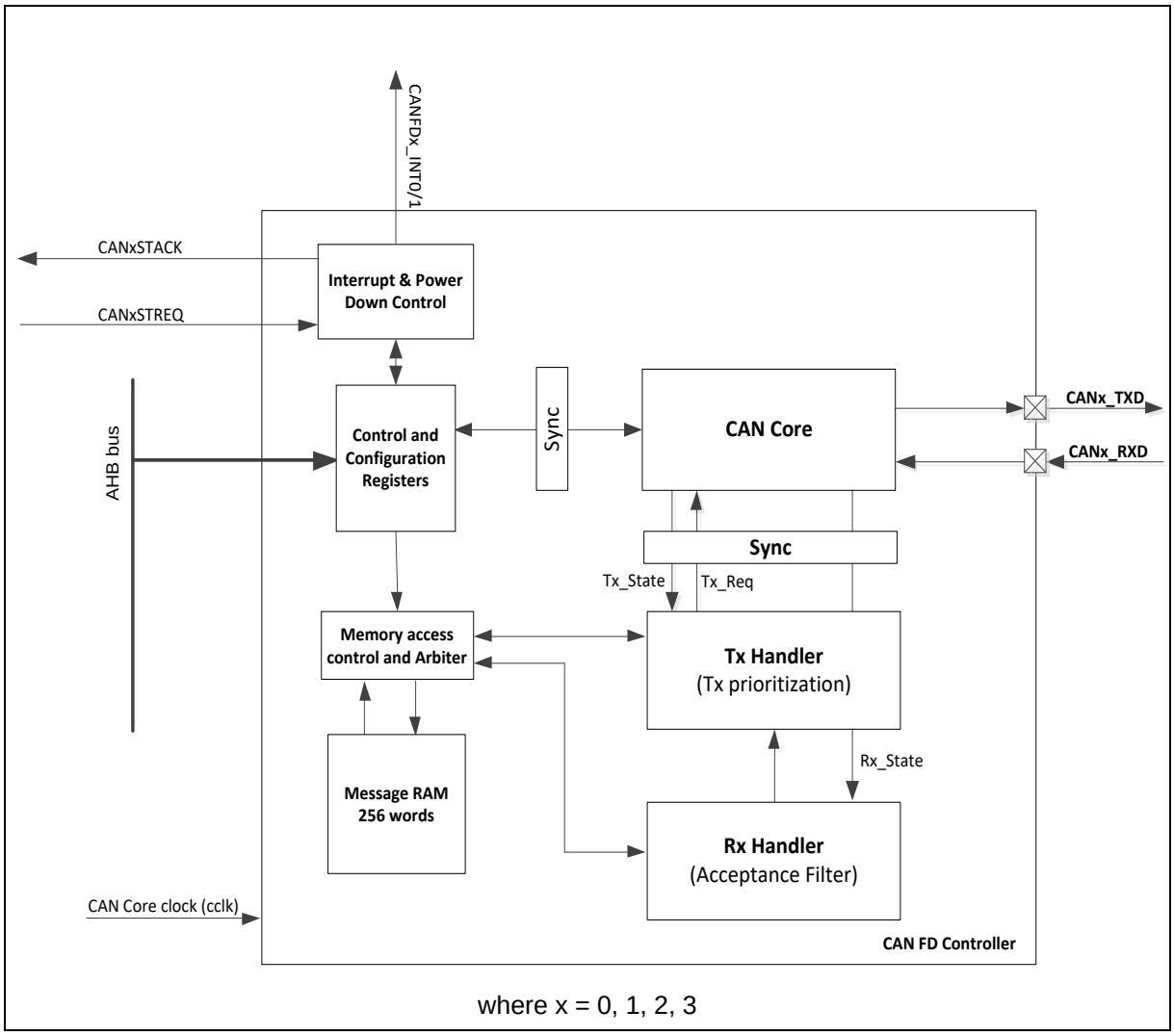


Figure 6.38-1 CAN FD Controller Block Diagram

6.38.4 Basic Configuration

6.38.4.1 CAN Basic Configuration

- Clock source Configuration
  - Enable CANFD0 controller clock (CANFD0CKEN (CLK_AHBCLK1[20])).
  - Enable CANFD1 controller clock (CANFD1CKEN (CLK_AHBCLK1[21])).
  - Enable CANFD2 controller clock (CANFD2CKEN (CLK_AHBCLK1[22])).
  - Enable CANFD3 controller clock (CANFD3CKEN (CLK_AHBCLK1[23])).
- Reset Configuration
  - Reset CANFD0 controller (CANFD0RST (SYS_IPRST0[20])).
  - Reset CANFD1 controller (CANFD1RST (SYS_IPRST0[21])).
  - Reset CANFD2 controller (CANFD2RST (SYS_IPRST0[22])).
  - Reset CANFD3 controller (CANFD3RST (SYS_IPRST0[23])).

6.38.5 Functional Description

6.38.5.1 Operation Mode

Software Initialization

Software initialization is started by setting bit INIT (CANFD_CCCR[0]) bit, either by software or by a hardware reset, when an uncorrected bit error was detected in the Message RAM, or by going Bus_Off. While INIT (CANFD_CCCR[0]) bit is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output CANx_TXD pin is recessive (HIGH). The counters of the Error Management Logic EML are unchanged. Setting INIT (CANFD_CCCR[0]) bit does not change any configuration register. Resetting INIT (CANFD_CCCR[0]) bit finishes the software initialization. Afterwards the Bit Stream Processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (=Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the CAN FD configuration registers is only enabled when both bits INIT (CANFD_CCCR[0]) and CCE (CANFD_CCCR[1]) are set (protected write). CCE (CANFD_CCCR[1]) can only be set/reset while INIT (CANFD_CCCR[0]) = 1. CCE (CANFD_CCCR[1]) is automatically reset when INIT (CANFD_CCCR[0]) is reset.

The following registers are reset when CCE (CANFD_CCCR[1]) is set:

- HPMS - High Priority Message Status
- RXF0S - Rx FIFO 0 Status
- RXF1S - Rx FIFO 1 Status
- TXFQS - Tx FIFO/Queue Status
- TXBRP - Tx Buffer Request Pending
- TXBTO - Tx Buffer Transmission Occurred
- TXBCF - Tx Buffer Cancellation Finished
- TXEFS - Tx Event FIFO Status

The Timeout Counter value TOC (CANFD_TOCV[15:0]) is preset to the value configured by TOP (CANFD_TOCC[31:16]) when CCE (CANFD_CCCR[1]) is set.

In addition the state machines of the Tx Handler and Rx Handler are held in idle state while CCE (CANFD_CCCR[1]) bit is set.

The following registers are only writeable while CCE (CANFD_CCCR[1]) bit is reset.

- TXBAR - Tx Buffer Add Request
- TXBCR - Tx Buffer Cancellation Request

TEST (CANFD_CCCR[7]) bit and MON (CANFD_CCCR[5]) bit can only be set by the software while both of INIT (CANFD_CCCR[0]) bit and CCE (CANFD_CCCR[1]) bit are set. Both bits may be reset at any time. DAR (CANFD_CCCR[6]) can only be set/reset while both INIT (CANFD_CCCR[0]) and CCE (CANFD_CCCR[1]) are set.

Normal Operation

Once the CAN FD controller is initialized and INIT (CANFD_CCCR[0]) is reset to zero, the CAN FD controller synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including Message ID and DLC are stored into a dedicated Rx Buffer or into Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx Buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not implemented.

CAN FD Operation

There are two variants in the CAN FD frame transmission:

(1). The CAN FD frame without bit rate switching.

(2). The CAN FD frame where control field, data field, and CRC field are transmitted with a higher bit rate than the beginning and the end of the frame.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers will now be decoded as FDF bit.

If FDF is recessive, it signifies a CAN FD frame.

If FDF is dominant, it signifies a Classic CAN frame.

In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bit rate inside of this CAN FD frame is switched. A CAN FD bit rate switch is signified by res = dominant and BRS = recessive. The coding of res = recessive is reserved for future expansion of the protocol.

In case the CAN FD controller receives a frame with FDF = recessive and res = recessive, it will signal a Protocol Exception Event by setting bit PXE (CANFD_PSR[14]) bit. When Protocol Exception Handling is enabled (PXHD (CANFD_CCCR[12]) is reset), this causes the operation state to change from Receiver (ACT (CANFD_PSR[4:3]) = 2'b10) to Integrating (ACT (CANFD_PSR[4:3]) = 00) at the next sample point.

In case Protocol Exception Handling is disabled (PXHD (CANFD_CCCR[12]) is set), the CAN FD controller will treat a recessive res bit as a form error and will respond with an error frame.

CAN FD operation is enabled by programming FDOE (CANFD_CCCR[8]). In case FDOE (CANFD_CCCR[8]) = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of Classic CAN frames is always possible. Whether a CAN FD frame or a Classic CAN frame is transmitted can be configured via bit FDF in the respective Tx Buffer element. With FDOE (CANFD_CCCR[8]) = 0, received frames are interpreted as Classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if bit FDF of a Tx Buffer element is set. FDOE (CANFD_CCCR[8]) and BRSE (CANFD_CCCR[9]) can only be changed while INIT (CANFD_CCCR[0]) and CCE (CANFD_CCCR[1]) are both set.

With FDOE (CANFD_CCCR[8]) = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in Classic CAN format. If FDOE (CANFD_CCCR[8]) is set and BRSE (CANFD_CCCR[9]) is clear, only bit FDF of a Tx Buffer element is evaluated. If both FDOE (CANFD_CCCR[8]) and BRSE (CANFD_CCCR[9]) are set, transmission of CAN FD frames with bit rate switching is enabled. All Tx Buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bit rate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup all nodes are transmitting Classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN Partial Networking have to be transmitted in Classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has completed. Then all nodes switch back to Classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15, in standard CAN all code a data field of 8 bytes, which are coded according to Table 6.38-1 below.

DLC	9	10	11	12	13	14	15
Number of Data Bytes	12	16	20	24	32	48	64

Table 6.38-1 DLC coding in CAN FD format

In CAN FD frames, the bit timing will be switched inside the frame, after the BRS (Bit Rate Switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the Nominal Bit Timing & Prescaler Register NBTP. In the following CAN FD data phase, the data phase bit timing is used as defined by the Data Bit Timing & Prescaler Register DBTP. The bit timing is switched back from the data phase timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the CAN Core clock frequency (cclk). Example: with a CAN FD controller clock frequency of 20 MHz and the shortest configurable bit time of 4 time quanta (tq), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD and CAN FD with bit rate switching, the value of the bit ESI (Error Status Indicator) is determined by the transmitter’s error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant.

Transmitter Delay Compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin CANx_TXD the CAN FD Controller receives the transmitted data from its local CAN transceiver via pin CANx_RXD. The received data is delayed by the transceiver loop delay. In case this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. In order to enable a data phase bit time that is even shorter than the transceiver loop delay, the delay compensation is introduced. Without transceiver delay compensation, the bit rate in the data phase of a CAN FD frame is limited by the transceiver loop delay.

The CAN FD Controller implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the CAN FD data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point (SSP). If a bit error is detected, the transmitter will react on this bit error at the next following regular sample point. During arbitration phase the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in ISO 11898-1:2015. It is enabled by setting bit TDC (CANFD_DBTP[23]).

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the CAN FD controllers transmit output CANx_TXD pin through the transceiver to the receive input CANx_RXD pin plus the transmitter delay compensation offset as configured by TDCO (CANFD_TDCR[[14:8]). The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (e.g. half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq (minimum time quantum).

TDCV (CANFD_PSR[22:16]) shows the actual transmitter delay compensation value. TDCV (CANFD_PSR[22:16]) is cleared when INIT (CANFD_CCCR[0]) is set and is updated at each transmission of an FD frame while TDC (CANFD_DBTP[23]) is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the CAN FD controller:

- The sum of the measured delay from CANx_TXD to CANx_RXD and the configured transmitter delay compensation offset TDCO (CANFD_TDCR[[14:8]) has to be less than 6 bit times in the data phase.

- The sum of the measured delay from CANx_TXD to CANx_RXD and the configured transmitter delay compensation offset TDCO (CANFD_TDCR[[14:8]) has to be less than or equal to 127 mtq (minimum time quantum). In case this sum exceeds 127 mtq, the maximum value of 127 mtq is used f or transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, that stops checking of receive bits at the SSPs.

If transmitter delay compensation is enabled by programming TDC (CANFD_DBTP[23]) is set, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input CANx_RXD pin of the transmitter. The resolution of this measurement is one mtq (minimum time quantum).

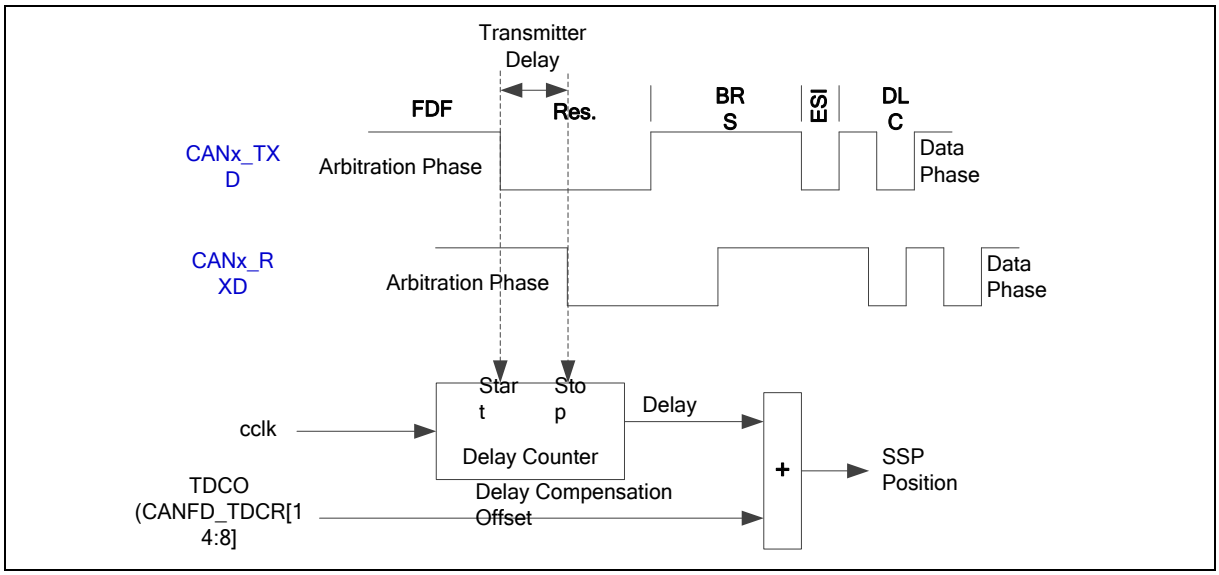


Figure 6.38-2 Transceiver Delay Measurement

To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit, resulting in a too early SSP position, the use of a transmitter delay compensation filter window can be enabled by programming TDCF (CANFD_TDCR[6:0]). This defines a minimum value for the SSP position. Dominant edges on CANx_RXD, that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF (CANFD_TDCR[6:0]) AND CANx_RXD is low.

Restricted Operation Mode

In Restricted Operation Mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (REC (CANFD_ECR[14:8]), TEC (CANFD_ECR[7:0])) are frozen while Error Logging (CEL (CANFD_ECR[23:16])) is active. The software can set the CAN FD controller into Restricted Operation mode by setting bit ASM (CANFD_CCCR[2]). The bit can only be set by the software when both CCE (CANFD_CCCR[1]) and INIT (CANFD_CCCR[0]) are set to 1. The bit can be reset by the software at any time.

Restricted Operation Mode is automatically entered when the Tx Handler was not able to read data from the Message RAM in time. To leave Restricted Operation Mode, the software has to reset ASM (CANFD_CCCR[2]).

The Restricted Operation Mode can be used in applications that adapt themselves to different CAN bit rates. In this case the application tests different bit rates and leaves the Restricted Operation Mode after it has received a valid frame.

Note: The Restricted Operation Mode must not be combined with the Loop Back Mode (internal or external).

Bus Monitor Mode

The CAN FD controller is set in Bus Monitoring Mode by programming MON (CANFD_CCCR[5]) to one. In Bus Monitoring Mode (refer to ISO 11898-1:2015, 10.14 Bus monitoring), the CAN FD controller is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the controller is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the controller monitors this dominant bit, although the CAN bus may remain in recessive state. In Bus Monitoring Mode register CANFD_TXBRP is held in reset state.

The Bus Monitoring Mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 6.38-3 shows the connection of signals CANx_TXD pin and CANx_RXD pin to the CAN FD controller in Bus Monitoring Mode.

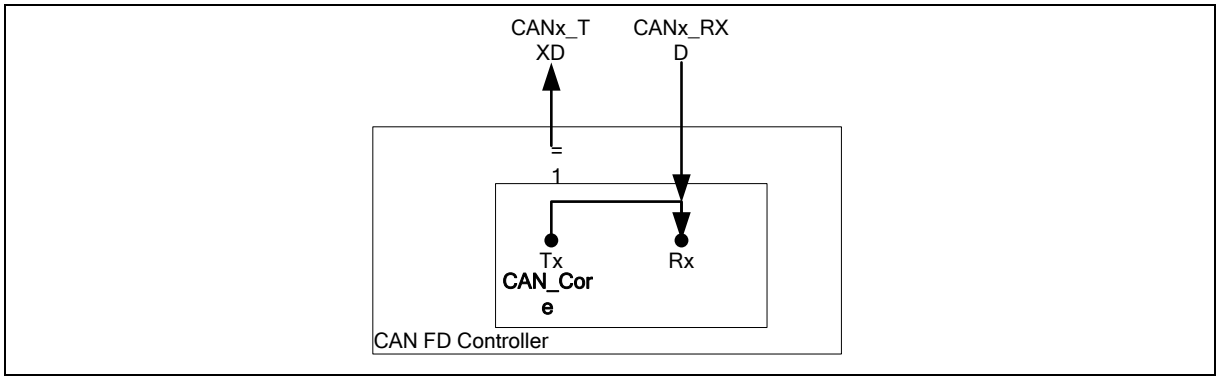


Figure 6.38-3 Pin Controller in Bus Monitor Mode

Disable Automatic Retransmission (DAR) mode

According to the CAN Specification (refer to ISO 11898-1:2015, 8.3.4 Recovery Management), the CAN FD controller provides means for automatic retransmission of frames that have lost arbitration or that have been disturbed by errors during transmission. By default, automatic retransmission is enabled. To support time-triggered communication as described in ISO 11898-1:2015, chapter 9.2, the automatic retransmission may be disabled via DAR (CANFD_CCCR[6]).

Frame Transmission in DAR Mode

In DAR mode all transmissions are automatically cancelled after they started on the CAN bus. A Tx Buffer’s Tx Request Pending bit TRPx (CANFD_TXBRP[31:0]) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx Buffer Transmission Occurred bit TOn (CANFD_TXBTO[31:0]) set
  - Corresponding Tx Buffer Cancellation Finished bit CFn (CANFD_TXBCF[31:0]) not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx Buffer Transmission Occurred bit TOn (CANFD_TXBTO[31:0]) set
  - Corresponding Tx Buffer Cancellation Finished bit CFn (CANFD_TXBCF[31:0]) set
- Arbitration lost or frame transmission disturbed:

- Corresponding Tx Buffer Transmission Occurred bit TOn (CANFD_TXBTO[31:0]) not set
- Corresponding Tx Buffer Cancellation Finished bit CFn (CANFD_TXBCF[31:0]) set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx Event FIFO element is written with Event Type ET = 10 (transmission in spite of cancellation).

Power Mode

The CAN FD controller can be set into Power-down mode controlled by input signal power_down_req or via CSR (CANFD_CCCR[4]). As long as the clock stop request signal power_down_req is active, bit CSR (CANFD_CCCR[4]) is read as one.

When all pending transmission requests have completed, the CAN FD controller waits until bus idle state is detected. Then the CAN FD controller sets then INIT (CANFD_CCCR[0]) to one to prevent any further CAN transfers. Now the CAN FD controller acknowledges that it is ready for power down by setting output signal power_down_ack to one and CSA(CANFD_CCCR[3]) to one. Before the clocks are switched off, further register accesses can be made except for INIT (CANFD_CCCR[0]) which is held at one.

Note: In case of a heavily disturbed CAN bus, it may happen that idle state is never reached and INIT (CANFD_CCCR[0]) is therefore not set by the controller. This situation can be detected by polling ACT (CANFD_PSR[4:3]). In case the controller does not enter idle state, the software can write INIT (CANFD_CCCR[0]) to 1 which stops CAN communication of the CAN FD controller immediately, regardless whether there is a transmission/reception ongoing or not.

To leave Power-down mode, the application has to turn on the module clocks before resetting control register flag CSR (CANFD_CCCR[4]). The controller will acknowledge this by resetting CSA (CANFD_CCCR[3]). Afterwards, the application can restart CAN communication by resetting bit INIT (CANFD_CCCR[0]).

Test Mode

To enable write access to register TEST, bit TEST (CANFD_CCCR[7]) has to be set to 1. This allows the configuration of the test modes and test functions.

Four output functions are available for the CAN transmit pin CANx_TXD by programming TX (CANFD_TEST[6:5]). Additionally to its default function – the serial data output – it can drive the CAN Sample Point signal to monitor the CAN FD controller bit timing and it can drive constant dominant or recessive values. The actual value at pin CANx_RXD can be read from RX (CANFD_TEST[7]). Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN Core clock and AHB clock domain, there may be a delay of several AHB clock periods between writing to TX (CANFD_TEST6:5]) until the new configuration is visible at output pin CANx_TXD. This applies also when reading input pin CANx_RXD via RX (CANFD_TEST[7]).

Note: Test modes should be used for production tests or self-test only. The software control for pin CANx_TXD interferes with all CAN protocol functions. It is not recommended to use test modes for application.

- External Loop Back Mode

  The CAN FD controller can be set in External Loop Back Mode by programming LBCK (CANFD_TEST[4]) to one. In Loop Back Mode, the controller treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into an Rx Buffer or an Rx FIFO. Figure 6.38-4 shows the connection of signals CANx_TXD pin and CANx_RXD pin to the CAN FD controller in External Loop Back Mode.

  This mode is provided for hardware self-test. To be independent from external stimulation, the controller ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in Loop Back Mode. In this mode the controller performs an internal

feedback from its Tx output to its Rx input. The actual value of the CANx_RXD input pin is disregarded by the controller. The transmitted messages can be monitored at the CANx_TXD pin.

- Internal Loop Back Mode

  Internal Loop Back Mode is entered by programming bits LBCK (CANFD_TEST[4]) and MON (CANFD_CCCR[5]) to one. This mode can be used for a “Hot Selftest”, meaning the controller can be tested without affecting a running CAN system connected to the pins CANx_TXD and CANx_RXD. In this mode pin CANx_RXD is disconnected from the controller and pin CANx_TXD is held recessive. Figure 6.38-4 shows the connection of CANx_TXD and CANx_RXD to the controller in case of Internal Loop Back Mode.

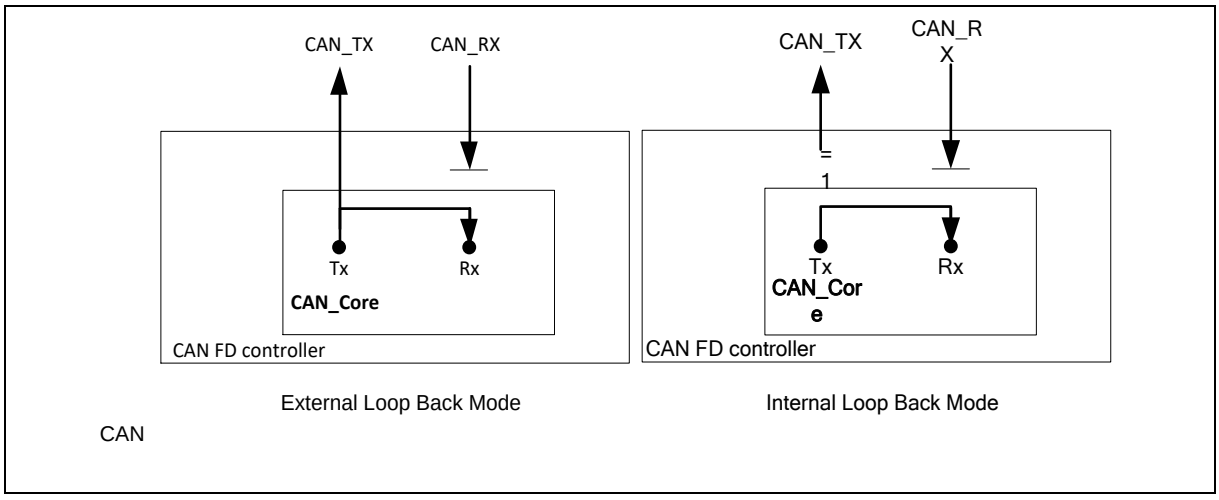


Figure 6.38-4 Pin Control in Loop Back Modes

6.38.5.2 Timestamp Generation

For timestamp generation the CAN FD controller supplies a 16-bit wrap-around counter. A pre-scaler TCP (CANFD_TSCC[19:16]) can be configured to clock the counter in multiples of CAN bit times (1…16). The counter is readable via TSC (CANFD_TSCV[15:0]). A write access to register CANFD_TSCV resets the counter to 0. When the timestamp counter wraps around interrupt flag TSW (CANFD_IR[16]) is set.

On start of frame reception / transmission (SOF) the counter value is captured and stored into the **Timestamp Generation** section of an Rx Buffer / Rx FIFO (RXTS(RX1A[15:0])) or Tx Event FIFO (TXTS(E1A[15:0])) element.

By programming bit TSS (CANFD_TSCC[[1:0]), a 16-bit timestamp can be used.

6.38.5.3 Timeout Counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO the CANFD controller supplies a 16-bit timeout counter. It operates as down-counter and uses the same pre-scaler controlled by TCP (CANFD_TSCC[19:16]) as the timestamp counter. The timeout counter is configured via register CANFD_TOCC. The actual counter value can be read from TOC (CANFD_TOCV[15:0]). The timeout counter can only be started while INIT (CANFD_CCCR[0]) = 0. It is stopped when INIT (CANFD_CCCR[0]) = 1, e.g. when the CAN FD Controller enters Bus_Off state.

The operation mode is selected by TOS (CANFD_TOCC[2:1]). When operating in Continuous Mode, the counter starts when INIT (CANFD_CCCR[0]) is reset. A write to TOCV presets the counter to the value configured by TOP (CANFD_TOCC[31:16]) and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP (CANFD_TOCC[31:16]). Down-counting is started when the first FIFO element

is stored. Writing to TOCV has no effect.

When the counter reaches zero, interrupt flag TOO (CANFD_IR[18]) is set. In Continuous Mode, the counter is immediately restarted at TOP (CANFD_TOCC[31:16]).

Note: The clock signal for the timeout counter is derived from the CAN Core's sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization / re-synchronization mechanism of the CAN Core. If the bit rate switch feature in CAN FD controller is used, the timeout counter is clocked differently in arbitration and data field.

6.38.5.4 Rx Handling

The Rx Handler controls the acceptance filtering, the transfer of received messages to the Rx Buffers or to one of the two Rx FIFOs, as well as the Rx FIFO's put and get Indices.

Acceptance Filtering

The CAN FD controller offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to an Rx Buffer or to Rx FIFO 0,1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from 0 to 128 elements for the 11-bit filter and from 0 to 64 for the 29-bit filter)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled / disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global Filter Configuration CANFD_GFC
- Standard ID Filter Configuration CANFD_SIDFC
- Extended ID Filter Configuration CANFD_XIDFC
- Extended ID AND Mask CANFD_XIDAM

Depending on the configuration of the filter element (SFEC/EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx Buffer
- Store received frame in Rx Buffer and generate pulse at filter event pin
- Reject received frame
- Set High Priority Message interrupt flag HPM (CANFD_IR[8])
- Set High Priority Message interrupt flag HPM (CANFD_IR[8]) and store received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx Buffer or Rx FIFO has been found, the Message Handler starts writing the received message data in portions of 32 bit to the matching Rx Buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (e.g. CRC error), this message is discarded with the

following impact on the affected Rx Buffer or Rx FIFO:

Rx Buffer

New Data flag of matching Rx Buffer is not set, but Rx Buffer (partly) overwritten with received data. For error type refer to LEC (CANFD_PSR[2:0]) respectively DLEC (CANFD_PSR[10:8]).

Rx FIFO

Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type refer to LEC (CANFD_PSR[2:0]) respectively DLEC (CANFD_PSR[10:8]). In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **Rx FIFO Overwrite Mode** section have to be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx Buffer, the unmodified received identifier is stored independent of the filter(s) used. The result of the acceptance filter process is strongly depending on the sequence of configured filter elements.

- Range Filter

  The filter matches for all received frames with Message IDs in the range defined by SF1ID/SF2ID resp. EF1ID/EF2ID.

  There are two possibilities when range filtering is used together with extended frames:

  EFT = 2'b00: The Message ID of received frames is ANDed with the Extended ID AND Mask (CANFD_XIDAM) before the range filter is applied.

  EFT = 2'b11: The Extended ID AND Mask (CANFD_XIDAM) is not used for range filtering.

- Filter for specific IDs

  A filter element can be configured to filter for one or two specific Message IDs. To filter for one specific Message ID, the filter element has to be configured with SF1ID = SF2ID resp. EF1ID = EF2ID.

- Classic Bit Mask Filter

  Classic bit mask filtering is intended to filter groups of Message IDs by masking single bits of a received Message ID. With classic bit mask filtering SF1ID/EF1ID is used as Message ID filter, while SF2ID/EF2ID is used as filter mask.

  A zero bit at the filter mask will mask out the corresponding bit position of the configured ID filter, e.g. the value of the received Message ID at that bit position is not relevant for acceptance filtering.

  Only those bits of the received Message ID where the corresponding mask bits are one are relevant for acceptance filtering.

  In case all mask bits are one, a match occurs only when the received Message ID and the Message

  ID filter are identical. If all mask bits are zero, all Message IDs match.

- Standard Message ID Filtering

  Figure 6.38-6 below shows the flow for standard Message ID (11-bit Identifier) filtering. The Standard Message ID Filter element is described in **Message RAM** section.

  Controlled by the global filter configuration GFC and the Standard ID filter configuration (CANFD_SIDFC) Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

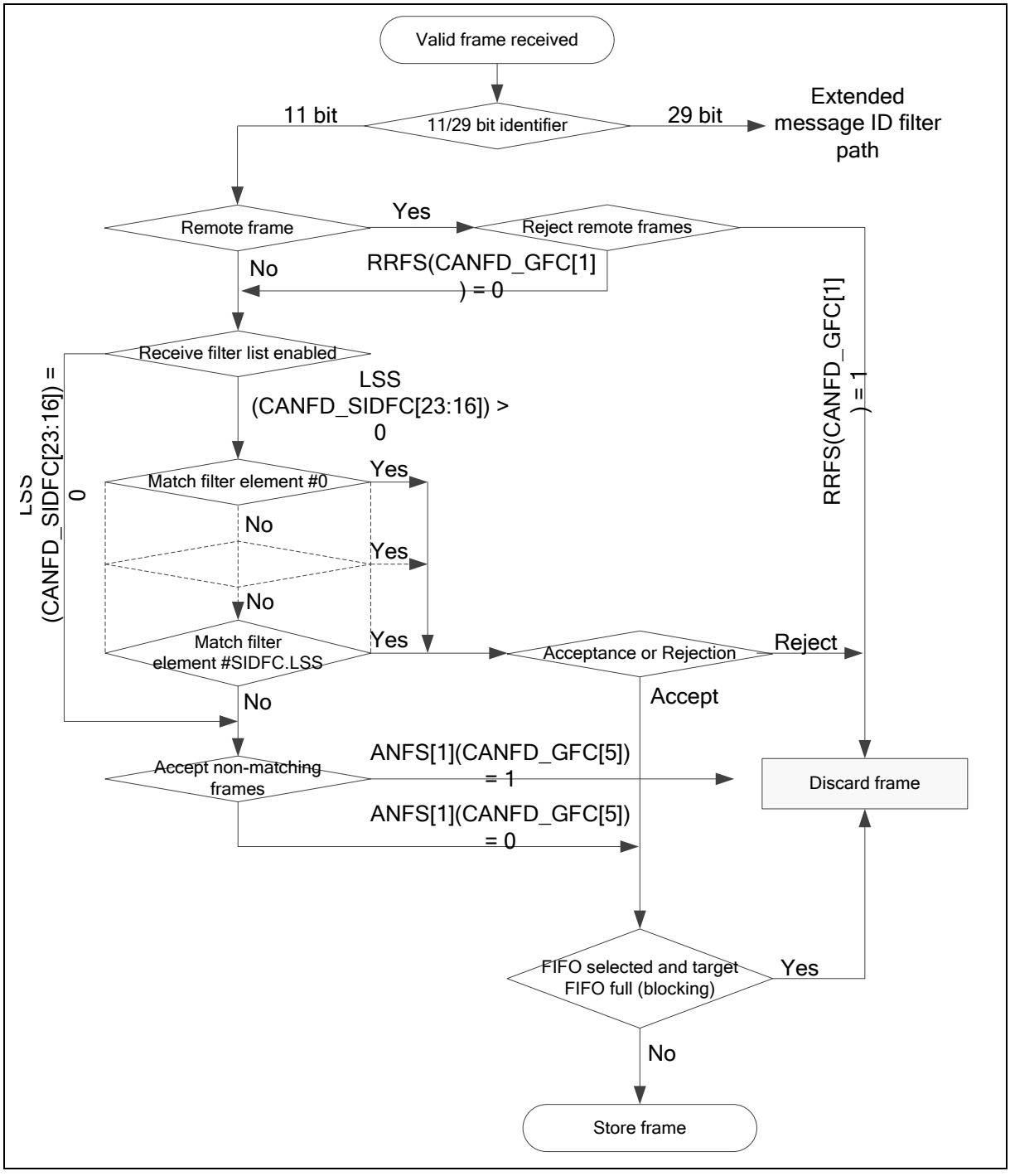


Figure 6.38-5 Standard Message ID Filter Path

- Extended Message ID Filtering

  Figure 6.38-6 shows the flow for extended Message ID (29-bit Identifier) filtering. The Extended Message ID Filter element is described in **Message RAM** Section.

  Controlled by the global filter configuration (CANFD_GFC) and the Extended ID filter configuration (CANFD_XIDFC) Message ID, Remote Transmission Request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The Extended ID AND Mask (CANFD_XIDAM) is AND-ed with the received identifier before the filter list is executed.

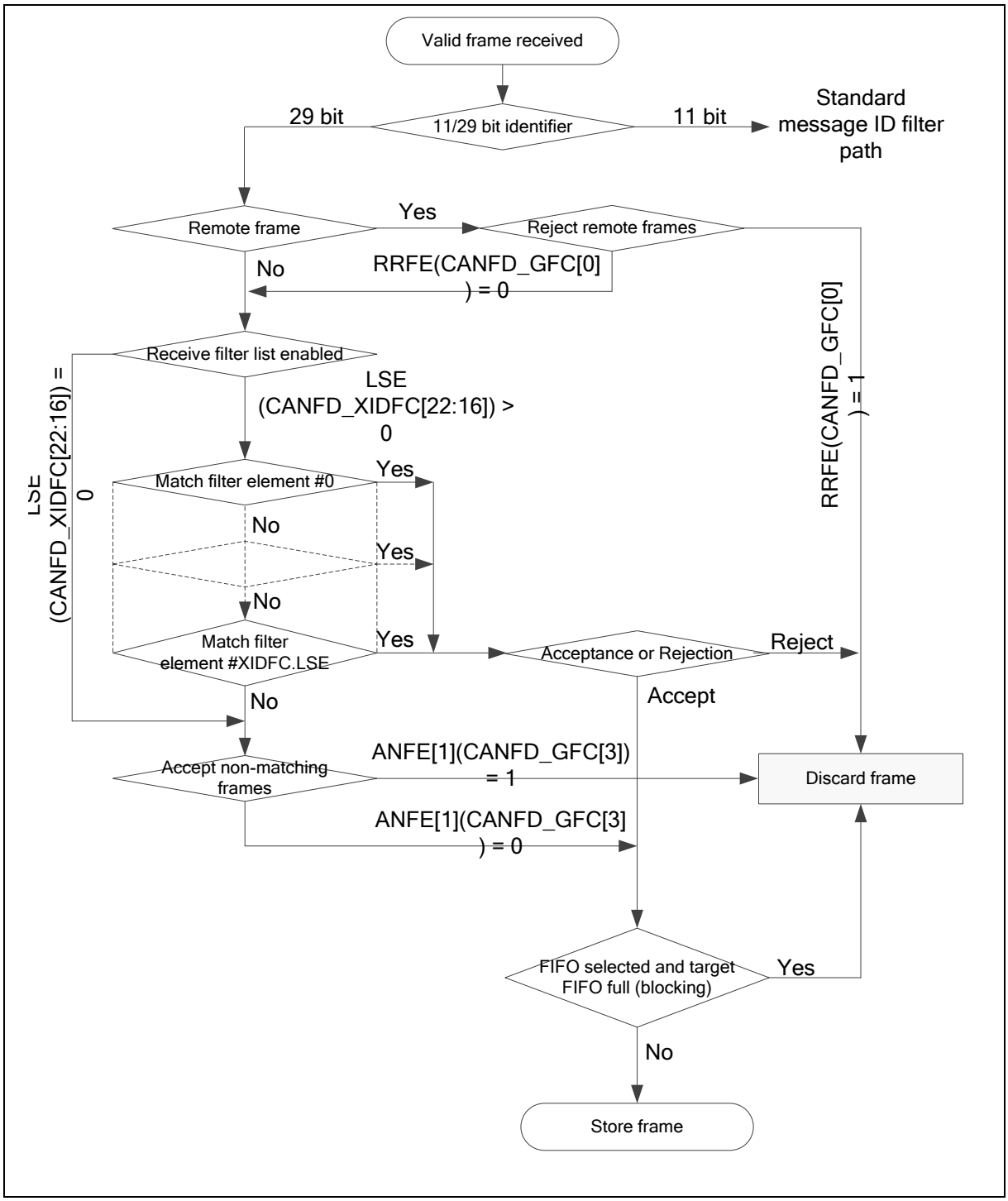


Figure 6.38-6 Extended Message ID Filter Path

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers CANFD_RXF0C and CANFD_RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by

the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1 refer to **Acceptance Filtering** section. The Rx FIFO element is described in Rx Buffer and **FIFO Element** section.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by FnWM (CANFD_RxFnC[30:24]), interrupt flag RFnW (CANFD_IR) is set. When the Rx FIFO put Index reaches the Rx FIFO get Index an Rx FIFO Full condition is signalled by FnF (CANFD_RXFnS). In addition, interrupt flag RFnF (CANFD_IR) is set.

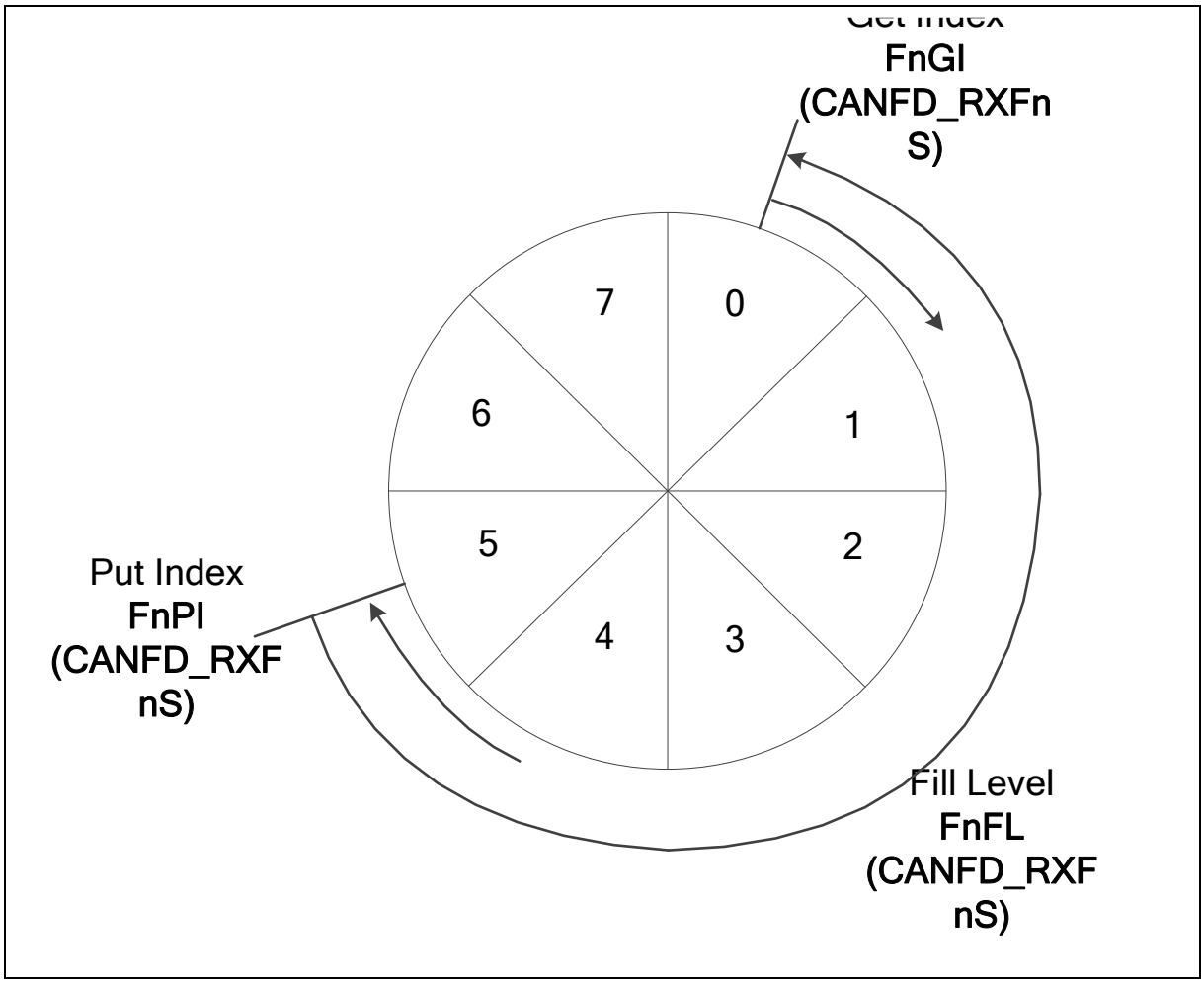


Figure 6.38-7 Rx FIFO Status

When reading from an Rx FIFO, Rx FIFO Get Index FnGI (CANFD_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address FnSA (CANFD_RXFnC).

RBDS(CANFD_RXESC[10:8]) FnDS(CANFD_RXESC[2:0])	Data Field [Bytes]	FIFO Element Size [RAM Words]
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8

101	32	10
110	48	14
111	64	18

Table 6.38-2 Rx Buffer / FIFO Element Size

- Rx FIFO Blocking Mode

  The Rx FIFO blocking mode is configured by FnOM (CANFD_RXFnC[31]) = 0. This is the default operation mode for the Rx FIFOs.

  When an Rx FIFO full condition is reached (FnPI (CANFD_RXFnS[21:16]) = FnGI (CANFD_RXFnS[13:8])), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO Get Index has been incremented. An Rx FIFO full condition is signalled by FnF (CANFD_RXFnS[24]) = 1. In addition interrupt flag RFnF (CANFD_IR) is set.

  In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signalled by RFnL (CANFD_RXFnS[25]) = 1. In addition interrupt flag RFnL (CANFD_IR) is set.

- Rx FIFO Overwrite Mode

  The Rx FIFO overwrite mode is configured by FnOM (CANFD_RXFnC[31]) = 1.

  When an Rx FIFO full condition (FnPI (CANFD_RXFnS[21:16]) = FnGI (CANFD_RXFnS[13:8])) is signalled by FnF (CANFD_RXFnS[24]) = 1, the next message accepted for the FIFO will overwrite the oldest FIFO message. Put and get index are both incremented by one.

  When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signalled, reading of the Rx FIFO elements should start at least at get index + 1. The reason for that is, that it might happen, that a received message is written to the Message RAM (put index) while the CPU is reading from the Message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. Figure 6.38-8 shows an offset of two with respect to the get index when reading the Rx FIFO. In this case the two messages stored in element 1 and 2 are lost.

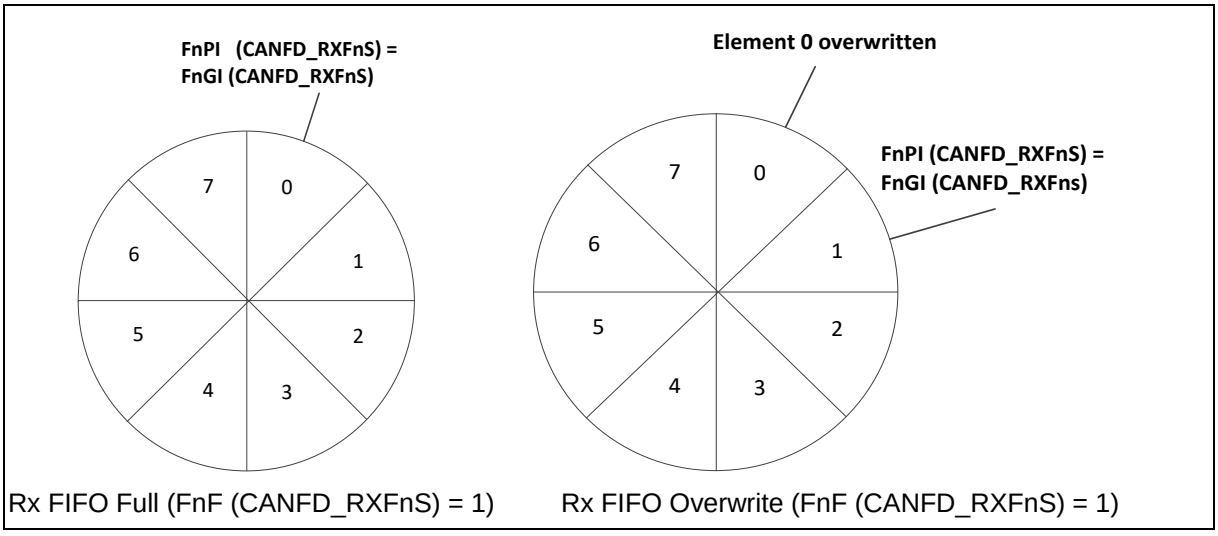


Figure 6.38-8 Rx FIFO Overflow Handling

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO Acknowledge Index FnA (CANFD_RXFnA[5:0]). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF (CANFD_RXFnS[24]) = '0').

Dedicated Rx Buffers

The CAN FD controller supports up to 64 dedicated Rx Buffers. The start address of the dedicated Rx Buffer section is configured via RBSA (CANFD_RXBC[15:2]).

For each Rx Buffer a Standard or Extended Message ID Filter Element with SFEC / EFEC = 111 and SFID2 / EFID2[10:9] = 00 has to be configured (refer to **Standard Message ID Filter Element** and **Extended Message ID Filter Element**).

After a received message has been accepted by a filter element, the message is stored into the Rx Buffer in the Message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition the flag DRX(CANFD_IR[19]) (Message stored in Dedicated Rx Buffer) in the interrupt register is set.

Filter Element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

Table 6.38-3 Example Filter Configuration for Rx Buffers

After the last word of a matching received message has been written to the Message RAM, the respective New Data flag in register NDAT1,2 is set. As long as the New Data flag is set, the respective Rx Buffer is locked against updates from received matching frames. The New Data flags have to be reset by the software by writing a '1' to the respective bit position.

While an Rx buffer new data flag is set, a message ID filter element referencing this specific Rx Buffer will not match, causing the acceptance filtering to continue. The following message ID filter elements may cause the received message to be stored into another Rx Buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

- Rx Buffer Handling
  - Reset interrupt flag DRX(CANFD_IR[19])
  - Read New Data registers
  - Read messages from Message RAM
  - Reset New Data flags of processed messages

6.38.5.5 Tx Handling

The Tx Handler handles transmission requests for the dedicated Tx Buffers, the Tx FIFO, and the Tx Queue. It controls the transfer of transmit messages to the CAN Core, the Put and Get Indices, and the Tx Event FIFO. Up to 32 Tx Buffers can be set up for message transmission. The CAN mode for transmission (Classic CAN or CAN FD) can be configured separately for each Tx Buffer element. The Tx Buffer element is described in **Tx Buffer Element** section. Table 6.38-4 describes the possible configurations for frame transmission.

CANFD_CCCR		Tx Buffer Element		Frame Transmission
BRSE	FDOE	FDF	BRS	
ignored	0	ignored	ignored	Classic CAN

0	1	0	ignored	Classic CAN
0	1	1	ignored	FD without bit rate switching
1	1	0	ignored	Classic CAN
1	1	1	0	FD without bit rate switching
1	1	1	1	FD with bit rate switching

Table 6.38-4 Possible Configuration for Frame Transmission

Note: AUTOSAR requires at least three Tx Queue Buffers and support of transmit cancellation

The Tx Handler starts a Tx scan of the Message RAM to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register CANFD_TXBRP is updated, or when a transmission has been started. CANFD_TXBRP is updated when a transmission finishes, or after the software has requested a transmission by writing to CANFD_TXBAR, or when the Host has cancelled a pending transmission by writing to CANFD_TXBCR.

Transmit Pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority should be inverse. This may lead to a case where one electronic control unit (ECU) sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If e.g. CAN ECU-1 has the transmit pause feature enabled and is requested by its application software to transmit four messages, it will, after the first successful message transmission, wait for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The transmit pause feature is controlled by bit TXP (CANFD_CCCR[14]). If the bit is set, the controller will, each time it has successfully transmitted a message, pause for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is transmit pause disabled (TXP (CANFD_CCCR[14]) = 0).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Dedicated Tx Buffers

Dedicated Tx Buffers are intended for message transmission under complete control of the CPU. Each Dedicated Tx Buffer is configured with a specific Message ID. In case that multiple Tx Buffers are configured with the same Message ID, the Tx Buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an Add Request via ARn (CANFD_TXBAR[31:0]). The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their Message ID.

A dedicated Tx buffer allocates four 32-bit words in the Message RAM (refer to Table 6.38-5). Therefore the start address of a dedicated Tx Buffer in the Message RAM is calculated by adding transmit buffer index (0…31) to the Tx buffer start address in TBSA (CANFD_TXBC[15:2]).

TBDS (CANFD_TXESC[2:0])	Data Field [Bytes]	Element Size [RAM Words]

000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Table 6.38-5 Tx Buffer / FIFO / Queue Element Size

Tx FIFO

Tx FIFO operation is configured by programming TFQM (CANFD_TXBC[30]) to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index TFGI (CANFD_TXFQS[12:8]). After each transmission the Get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same Message ID from different Tx Buffers in the order these messages have been written to the Tx FIFO. The controller calculates the Tx FIFO Free Level TFFL (CANFD_TXFQS[5:0]) as difference between get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx Buffer referenced by the put index TFQPI(CANFD_TXFQS[20:16]). An add request increments the Put Index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO Full (TFQF (CANFD_TXFQS[21]) = 1) is signaled. In this case no further messages should be written to the Tx FIFO until the next message has been transmitted and the Get Index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a 1 to the TXBAR bit related to the Tx Buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx Buffers starting with the Put Index. The transmissions are then requested via CANFD_TXBAR. The put index is then cyclically incremented by n. The number of requested Tx buffers should not exceed the number of free Tx Buffers as indicated by the Tx FIFO Free Level.

When a transmission request for the Tx Buffer referenced by the get Index is cancelled, the Get index is incremented to the next Tx Buffer with pending transmission request and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx Buffer, the get index and the FIFO free level remain unchanged.

A Tx FIFO element allocates element size 32-bit words in the Message RAM (refer to Table 6.38-5). Therefore the start address of the next available (free) Tx FIFO Buffer is calculated by adding the put index TFQPI (CANFD_TXFQS[20:16]) * element size to the Tx buffer start address TBSA (CANFD_TXBC[15:2]).

Tx Queue

Tx Queue operation is configured by programming TFQM (CANFD_TXBC[30]) to 1. Messages stored in the Tx Queue are transmitted starting with the message with the lowest Message ID (highest priority). In case that multiple Queue Buffers are configured with the same Message ID, the Queue Buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx Buffer referenced by the Put Index TFQPI (CANFD_TXFQS[20:16]). An Add Request cyclically increments the Put Index to the next free Tx Buffer. In case that the Tx Queue is full (TFQF (CANFD_TXFQS[21]) = 1), the Put Index is not valid and no further message should be written to the Tx Queue until at least one of the requested messages has been sent out or a pending transmission request has been cancelled.

The application may use register CANFD_TXBRP instead of the put index and may place messages to any Tx Buffer without pending transmission request.

A Tx Queue Buffer allocates element size 32-bit words in the Message RAM (refer to Table 6.38-5). Therefore the start address of the next available (free) Tx Queue Buffer is calculated by adding Tx FIFO/Queue put index TFQPI (CANFD_TXFQS[20:16]) * element size to the Tx buffer start address TBSA (CANFD_TXBC[15:2]).

Mixed Dedicated Tx Buffers / Tx FIFO

In this case the Tx Buffers section in the Message RAM is subdivided into a set of dedicated Tx Buffers and a Tx FIFO. The number of dedicated Tx Buffers is configured by NDTB (CANFD_TXBC[21:16]). The number of Tx Buffers assigned to the Tx FIFO is configured by TFQS (CANFD_TXBC[29:24]). In case TFQS (CANFD_TXBC[29:24]) is programmed to zero, only Dedicated Tx Buffers are used.

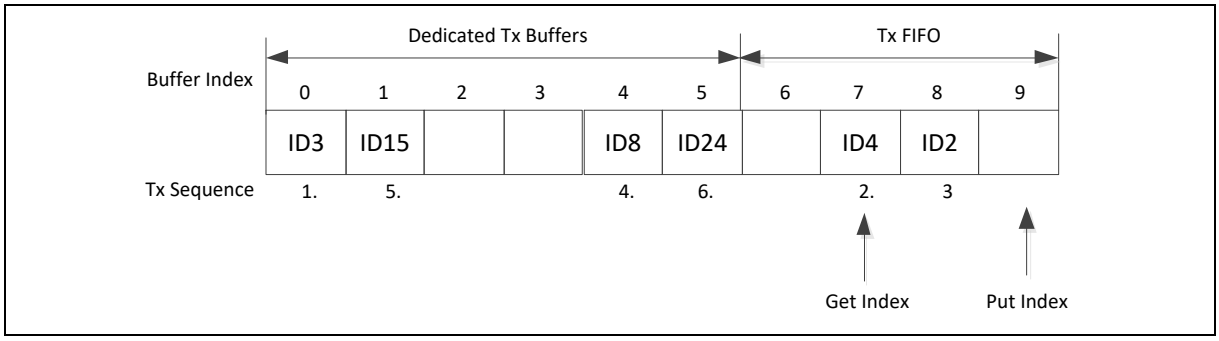


Figure 6.38-9 Example of Mixed Configuration Dedicated Tx Buffers / Tx FIFO

Tx prioritization:

- Scan Dedicated Tx Buffers and oldest pending Tx FIFO Buffer (referenced by TFGI (CANFD_TXFQS[12:8]))
- Buffer with lowest Message ID gets highest priority and is transmitted next.

Mixed Dedicated Tx Buffers / Tx Queue

In this case the Tx Buffers section in the Message RAM is subdivided into a set of dedicated Tx Buffers and a Tx Queue. The number of dedicated Tx Buffers is configured by NDTB (CANFD_TXBC[21:16]). The number of Tx Queue Buffers is configured by TFQS (CANFD_TXBC[29:24]). In case TFQS (CANFD_TXBC[29:24]) is programmed to zero, only dedicated Tx Buffers are used.

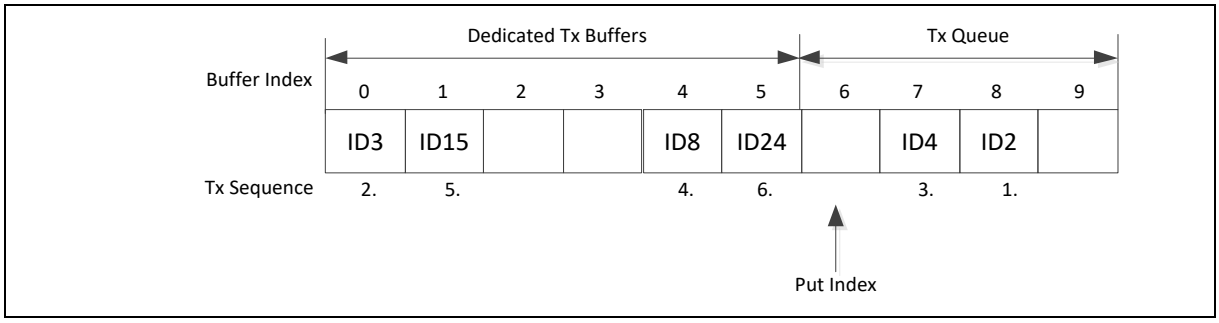


Figure 6.38-10 Example of mixed Configuration Dedicated Tx Buffers / Tx Queue

Tx prioritization:

- Scan all Tx Buffers with activated transmission request
- Tx Buffer with lowest Message ID gets highest priority and is transmitted next

Transmit Cancellation

The CAN FD controller supports transmit cancellation. This feature is especially intended for gateway applications and AUTOSAR based applications. To cancel a requested transmission from a dedicated Tx Buffer or a Tx Queue Buffer the software has to write a 1 to the corresponding bit position (=number of Tx Buffer) of register CANFD_TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register CANFD_TXBCF to 1. In case a transmit cancellation is requested while a transmission from a Tx Buffer is already ongoing, the corresponding CANFD_TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding CANFD_TXBTO and CANFD_TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding CANFD_TXBCF bit is set.

Note: In case a pending transmission is cancelled immediately before this transmission could have been started, there follows a short time window where no transmission is started even if another message is also pending in this node. This may enable another node to transmit a message which may have a lower priority than the second message in this node.

Tx Event Handling

To support Tx event handling the controller has implemented a Tx Event FIFO. After the controller has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx Event FIFO element. To link a Tx event to a Tx Event FIFO element, the message marker from the transmitted Tx Buffer is copied into the Tx Event FIFO element.

The Tx Event FIFO can be configured to a maximum of 32 elements. The Tx Event FIFO element is described in **Tx Event FIFO Element** section.

The purpose of the Tx Event FIFO is to decouple handling transmit status information from transmit message handling i.e. a Tx Buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx Event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx Buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx Buffer before overwriting that Tx Buffer.

When a Tx Event FIFO full condition is signaled by TEFF (CANFD_IR[14]), no further elements are written to the Tx Event FIFO until at least one element has been read out and the Tx Event FIFO get Index has been incremented. In case a Tx event occurs while the Tx Event FIFO is full, this event is discarded and interrupt flag TEFL (CANFD_IR[15]) is set.

To avoid a Tx Event FIFO overflow, the Tx Event FIFO watermark can be used. When the Tx Event FIFO fill level reaches the Tx Event FIFO watermark configured by EFWM (CANFD_TXEFC[29:24]), interrupt flag TEFW (CANFD_IR[13]) is set.

When reading from the Tx Event FIFO, two times the Tx Event FIFO get Index EFGI (CANFD_TXEFS[12:8]) has to be added to the Tx Event FIFO start address EFSA (CANFD_TXEFC[15:2]).

6.38.5.6 FIFO Acknowledge Handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx Event FIFO are controlled by writing to the corresponding FIFO Acknowledge Index (refer to **Rx FIFO 0 Acknowledge**, **Rx FIFO 1 Acknowledge**, and **Tx Event FIFO Acknowledge**). Writing to the FIFO Acknowledge Index will set the FIFO Get Index to the FIFO Acknowledge Index plus one and thereby updates the FIFO Fill Level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO Acknowledge Index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO Acknowledge Index only once at the end of that read sequence (value: Index of the last element read), to update the FIFO’ s get index.

Due to the fact that the CPU has free access to the controller Message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a High Priority Message from one of the two Rx FIFOs. In this case the FIFO's Acknowledge Index should not be written because this would set the get index to a wrong position and also alters the FIFO's Fill Level. In this case some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO Acknowledge Index. The controller does not check for erroneous values.

6.38.5.7 Message RAM

For storage of Rx/Tx messages and for storage of the filter configuration a single-ported Message RAM has to be connected to the CAN Core module.

Message RAM Configuration

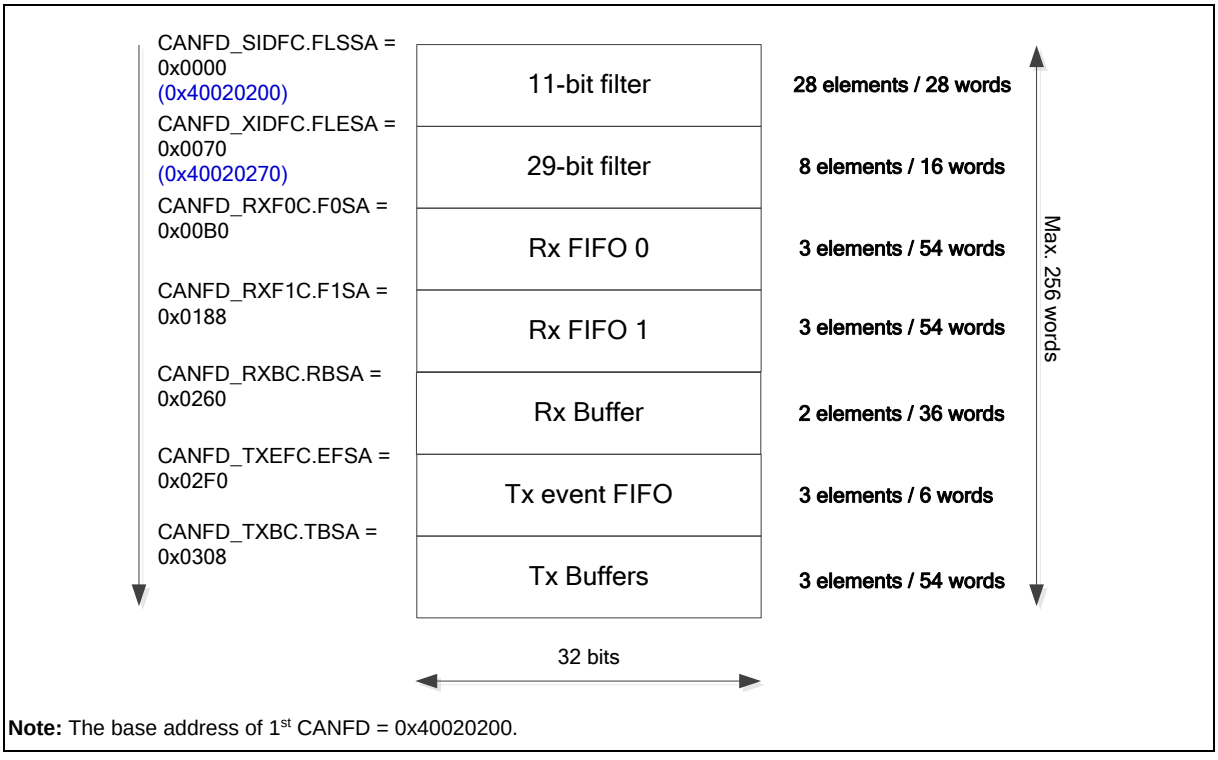
The address of the Message RAM and the address of the register are accessed through the same AHB interface. The maximum address of register used by the CAN FD controller is within 0x200. Therefore, the message SRAM used by the CAN controller must be used after 0x200. When reading or writing the SRAM content, the CAN controller will automatically subtract the offset of 0x200.

For example:

When the CPU wants to access the location of SRAM address is 0x100, its AHB address should be set as CANFD Base address + 0x200+0x100.

The Message RAM has a width of 32 bits. The CAN FD controller can be configured to allocate up to 1536 words in the Message RAM. It is not necessary to configure each of the sections listed in Figure 6.38-11, nor is there any restriction with respect to the sequence of the sections.

When operated in CAN FD mode the required Message RAM size strongly depends on the element size configured for Rx FIFO0, Rx FIFO1, Rx Buffers, and Tx Buffers via F0DS (CANFD_RXESC[2:0], F1DS (CANFD_RXESC[6:4]), RBDS (CANFD_RXESC[10:8]), and TBDS (CANFD_TXESC[2:0]).



Note: The base address of 1st CANFD = 0x40020200.

Figure 6.38-11 Message RAM configuration

When the controller addresses the Message RAM it addresses 32-bit words, not single bytes. The configurable start addresses are 32-bit word addresses i.e. only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Rx Buffer and FIFO Element

Two Rx FIFOs and up to 64 Rx Buffers can be configured in the Message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx Buffer / FIFO element is shown in below. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register CANFD_RXESC.

Bit	31			24	23			16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]								
R1	ANMF	FIDX[6:0]			-	-	FDF	BRS	DLC[3:0]	RXTS[15:0]		
R2	DB3[7:0]				DB2[7:0]				DB1[7:0]		DB0[7:0]	
R3	DB7[7:0]				DB6[7:0]				DB5[7:0]		DB4[7:0]	

Rn	DBm[7:0]				DBm-1[7:0]				DBm-2[7:0]		DBm-3[7:0]	

Table 6.38-6 Rx Buffer and FIFO Element

Bit Field	Description
R0 Bit 31 ESI	Error State Indicator 0 = Transmitting node is error active 1 = Transmitting node is error passive
R0 Bit 30 XTD	Extended Identifier Signals to the MCU whether the received frame has a standard or extended identifier. 0 = 11-bit standard identifier 1 = 29-bit extended identifier
R0 Bit 29 RTR	Remote Transmission Request Signals to the Host whether the received frame is a data frame or a remote frame. 0 = Received frame is a data frame 1 = Received frame is a remote frame **Note:** There are no remote frames in CAN FD format. In CAN FD frames (FDF = 1'), the dominant RRS (Remote Request Substitution) bit replaces bit RTR (Remote Transmission Request)
R0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 Bit 31 ANMF	Accepted Non-matching Frame Acceptance of non-matching frames may be enabled via ANFS (CANFD_GFC[5:4]) and ANFE (CANFD_GFC[3:2]). 0 = Received frame matching filter index FIDX 1 = Received frame did not match any Rx filter element

R1 Bits 30:24 FIDX[6:0]	Filter Index 0-127=Index of matching Rx acceptance filter element (invalid if ANMF = '1'). **Note:** Range is 0 to LSS (CANFD_SIDFC[23:16]) – 1 for standard and 0 to LSE (CANFD_XIDFC[22:16]) – 1 for extended.
R1 Bit 21 FDF	FD Format 0 = Classical CAN frame format 1 = CAN FD frame format (new DLC-coding and CRC)
R1 Bit 20 BRS	Bit Rate Switch 0 = Frame received without bit rate switching 1 = Frame received with bit rate switching
R1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 = CAN + CAN FD: received frame has 0-8 data bytes 9-15 = CAN: received frame has 8 data bytes 9-15 = CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 Bits 15:0 RXTS[15:0]	Rx Timestamp Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the Timestamp Counter Prescaler TCP (CANFD_TSCC[19:16]).
R2 Bits 31:24 DB3[7:0]	Data Byte 3
R2 Bits 23:16 DB2[7:0]	Data Byte 2
R2 Bits 15:8 DB1[7:0]	Data Byte 1
R2 Bits 7:0 DB0[7:0]	Data Byte 0
R3 Bits 31:24 DB7[7:0]	Data Byte 7
R3 Bits 23:16 DB6[7:0]	Data Byte 6
R3 Bits 15:8 DB5[7:0]	Data Byte 5
R3 Bits 7:0 DB4[7:0]	Data Byte 4
…	…
Rn Bits 31:24 DBm[7:0]	Data Byte mm
Rn Bits 23:16 DBm-1[7:0]	Data Byte m-1
Rn Bits 15:8 DBm-2[7:0]	Data Byte m-2
Rn Bits 7:0 DBm-3[7:0]	Data Byte m-3 **Note:** Depending on the configuration of the element size (RXESC), between two and sixteen 32-bit words (Rn = 2 ..17) are used for storage of a CAN message's data field

Table 6.38-7 Rx Buffer and FIFO Element Description

Tx Buffer Element

The Tx Buffers section can be configured to hold dedicated Tx Buffers as well as a Tx FIFO / Tx Queue. In case that the Tx Buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx Queue, the dedicated Tx Buffers start at the beginning of the Tx Buffers section followed by the buffers assigned to the Tx FIFO or Tx Queue. The Tx Handler distinguishes between dedicated Tx Buffers and Tx FIFO / Tx Queue by evaluating the Tx Buffer configuration TFQS (CANFD_TXBC[29:24]) and NDTB (CANFD_TXBC[21:16]). The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register CANFD_TXESC.

Bit	31	30	29	28:0	23	22	21	20	19:16	15:8	7:0
T0	ESI	XTD	RTR	ID[28:0]							
T1	MM[7:0]				EFC	-	FDF	BRS	DLC[3:0]	MM[15:8]	-
T2	DB3[7:0]				DB2[7:0]					DB1[7:0]	DB0[7:0]
T3	DB7[7:0]				DB6[7:0]					DB5[7:0]	DB4[7:0]

Tn	DBm[7:0]				DBm-1[7:0]					DBm-2[7:0]	DBm-3[7:0]

Table 6.38-8 Tx Buffer and FIFO Element

Bit Field	Description
T0 Bit 31 ESI	Error State Indicator 0 = ESI bit in CAN FD format depends only on error passive flag 1 = ESI bit in CAN FD format transmitted recessive **Note:** The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node will always transmit the ESI bit recessive.
T0 Bit 30 XTD	Extended Identifier 0 = 11-bit standard identifier 1 = 29-bit extended identifier
T0 Bit 29 RTR	Remote Transmission Request 0 = Transmit data frame 1 = Transmit remote frame **Note:** When RTR = 1, the controller transmits a remote frame according to ISO 11898-1:2015, even if FDOE (CANFD_CCCR[8]) enables the transmission in CAN FD format.
T0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18].
T1 Bits 31:24 MM[7:0]	Message Marker Written by CPU during Tx Buffer configuration. Copied into Tx Event FIFO element for identification of Tx message status.
T1 Bit 23 EFC	Event FIFO Control 0 = Don't store Tx events 1 = Store Tx events
T1 Bit 21	FD Format

FDF	0 = Frame transmitted in Classic CAN format 1 = Frame transmitted in CAN FD format
T1 Bit 20 BRS	Bit Rate Switch 0 = CAN FD frames transmitted without bit rate switching 1 = CAN FD frames transmitted with bit rate switching **Note:** Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled FDOE (CANFD_CCCR[8]) = 1. Bit BRS is only evaluated when in addition BRSE (CANFD_CCCR[9]) = 1.
T1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 = CAN + CAN FD: transmit frame has 0-8 data bytes 9-15 = CAN: transmit frame has 8 data bytes 9-15 = CAN FD: transmit frame has 12/16/20/24/32/48/64 data bytes
T2 Bits 31:24 DB3[7:0]	Data Byte 3
T2 Bits 23:16 DB2[7:0]	Data Byte 2
T2 Bits 15:8 DB1[7:0]	Data Byte 1
T2 Bits 7:0 DB0[7:0]	Data Byte 0
T3 Bits 31:24 DB7[7:0]	Data Byte 7
T3 Bits 23:16 DB6[7:0]:	Data Byte 6
T3 Bits 15:8 DB5[7:0]	Data Byte 5
T3 Bits 7:0 DB4[7:0]	Data Byte 4
... ...	... ...
Tn Bits 31:24 DBm[7:0]	Data Byte m
Tn Bits 23:16 DBm-1[7:0]	Data Byte m-1
Tn Bits 15:8 DBm-2[7:0]	Data Byte m-2
Tn Bits 7:0 DBm-3[7:0]	Data Byte m-3 **Note:** Depending on the configuration of the element size (TXESC), between two and sixteen 32-bit words (Tn = 2 ..17) are used for storage of a CAN message's data field.

Table 6.38-9 Tx Buffer Element Description

Tx Event FIFO Element

Each element stores information about transmitted messages. By reading the Tx Event FIFO the MCU gets this information in the order the messages were transmitted. Status information about the Tx Event

FIFO can be obtained from register CANFD_TXEFS.

Bit	31			24	23			16	15	8	7	0
E0	ESI	XTD	RTR	ID[28:0]								
E1	MM[7:0]				ET [1:0]	FDF	BRS	DLC[3:0]	TXTS[15:0]			

Table 6.38-10 Tx Event FIFO Element

Bit Field	Description
E0 Bit 31 ESI	Error State Indicator 0 = Transmitting node is error active 1 = Transmitting node is error passive
E0 Bit 30 XTD	Extended Identifier 0 = 11-bit standard identifier 1 = 29-bit extended identifier
E0 Bit 29 RTR	Remote Transmission Request 0 = Data frame transmitted 1 = Remote frame transmitted
E0 Bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
E1 Bits 31:24 MM[7:0]	Message Marker Copied from Tx Buffer into Tx Event FIFO element for identification of Tx message status.
E1 Bit 23:22 ET[1:0]	Event Type 00 = Reserved 01 = Tx event 10 = Transmission in spite of cancellation (always set for transmissions in DAR mode) 11 = Reserved
E1 Bit 21 FDF	FD Format 0 = Classical CAN frame format 1 = CAN FD frame format (new DLC-coding and CRC)
E1 Bit 20 BRS	Bit Rate Switch 0 = Frame transmitted without bit rate switching 1 = Frame transmitted with bit rate switching
E1 Bits 19:16 DLC[3:0]	Data Length Code 0-8 = CAN + CAN FD: frame with 0-8 data bytes transmitted 9-15 = CAN: frame with 8 data bytes transmitted 9-15 = CAN FD: frame with 12/16/20/24/32/48/64 data bytes transmitted
E1 Bits 15:0 TXTS[15:0]	Tx Timestamp Timestamp Counter value captured on start of frame transmission. Resolution depending on configuration of the Timestamp Counter Prescaler TCP (CANFD_TSCC[19:16]).

Table 6.38-11 Tx Event FIFO Element Description

Standard Message ID Filter Element

Up to 128 filter elements can be configured for 11-bit IDs. When accessing a Standard Message ID Filter element, its address is the Filter List Standard Start Address FLSSA (CANFD_SIDFC[15:2]) plus the index of the filter element (0…127).

Bit	31		24	23	16	15	8	7	0
S0	SFT [1:0]	SFEC [2:0]	SFID1[10:0]			-	SFID2[10:0]		

Table 6.38-12 Standard Message ID Filter Element

Bit Field		Description
Bits 31:30 SFT[1:0]		Standard Filter Type 00 = Range filter from SFID1 to SFID2 (SFID2 ≥ SFID1) 01 = Dual ID filter for SFID1 or SFID2 10 = Classic filter: SFID1 = filter, SFID2 = mask 11 = Filter element disabled **Note:** With SFT = 11 the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = 000)
Bit 29:27 SFEC[2:0]		Standard Filter Element Configuration All enabled filter elements are used for acceptance filtering of 11-bit ID frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = 100, 101, or 110 a match sets interrupt flag HPM (CANFD_IR[8]) and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 = Disable filter element 001 = Store in Rx FIFO 0 if filter matches 010 = Store in Rx FIFO 1 if filter matches 011 = Reject ID if filter matches, not intended to be used with Sync messages 100 = Set priority if filter matches, not intended to be used with Sync messages, no storage 101 = Set priority and store in FIFO 0 if filter matches 110 = Set priority and store in FIFO 1 if filter matches 111 = Store into Rx Buffer, configuration of SFT[1:0] ignored
Bits 26:16 SFID1[10:0]		Standard Filter ID 1 First ID of standard ID filter element. When filtering for Rx Buffers, Sync messages, this field defines the ID of the message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 15:0	SFID2[10:0]	Standard filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: SFEC = 001 ... 110 Second ID of standard ID filter element SFEC = 111 Filter for Rx buffers
	SFID2[10:9]	Decides whether the received message is stored into an Rx buffer . 00: Store message into an Rx buffer
	SFID2[8:6]	The bits are used to control the filter event pins at the Extension Interface. A '1' at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one system clock period in case the filter matches. SFID2[8] is used by the calibration unit.
	SFID2[5:0]	Defines the offset to the Rx buffer start address RBSA (CANFD_RXBC[15:2]) for storage of a matching message.

Table 6.38-13 Standard Message ID Filter Element Description

Extended Message ID Filter Element

Up to 64 filter elements can be configured for 29-bit IDs. When accessing an Extended Message ID Filter element, its address is the Filter List Extended Start Address FLESA (CANFD_XIDFC[15:2]) plus two times the index of the filter element (0…63).

Bit	31		24	23	16	15	8	7	0
F0	EFECV [2:0]		EFID1[28:0]						
F0	EFT [1:0]	-	EFID2[28:0]						

Table 6.38-14 Extended Message ID Filter Element

Field		Description
F0 Bit 31:29 EFEC[2:0]		Extended Filter Element Configuration All enabled filter elements are used for acceptance filtering of 29-bit ID frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag HPM (CANFD_IR[8]) and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match. 000 = Disable filter element 001 = Store in Rx FIFO 0 if filter matches 010 = Store in Rx FIFO 1 if filter matches 011 = Reject ID if filter matches 100 = Set priority if filter matches 101 = Set priority and store in FIFO 0 if filter matches 110 = Set priority and store in FIFO 1 if filter matches 111 = Store into Rx Buffer, configuration of EFT[1:0] ignored
F0 Bits 28:0 EFID1[28:0]		Extended Filter ID 1 First ID of extended ID filter element. When filtering for Rx Buffers, Sync messages, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only CANFD_XIDAM masking mechanism (refer to Extended Message ID Filtering) is used.
F1 Bits 31:30 EFT[1:0]		Extended Filter Type 00 = Range filter from EFID1 to EFID2 (EFID2 >- EFID1) 01 = Dual ID filter for EFID1 or EFID2 10 = Classic filter: EFID1 = filter, EFID2 = mask 11 = Range filter from EFID1 to EFID2 (EFID2 >= EFID1), CANFD_XIDAM mask not applied.
F1 Bits 28:0	EFID2[10:0]	Extended filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: – SFEC = 001 ... 110 Second ID of extended ID filter element – SFEC = 111 Filter for Rx buffers
	EFID2[10:9]	Decides whether the received message is stored into an Rx Buffer. 00 = Store message into an Rx Buffer
	EFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A one at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one system clock period in case the filter matches.
	EFID2[5:0]	Defines the offset to the Rx Buffer Start Address RBSA (CANFD_RXBC[15:2]) for storage of a matching message.

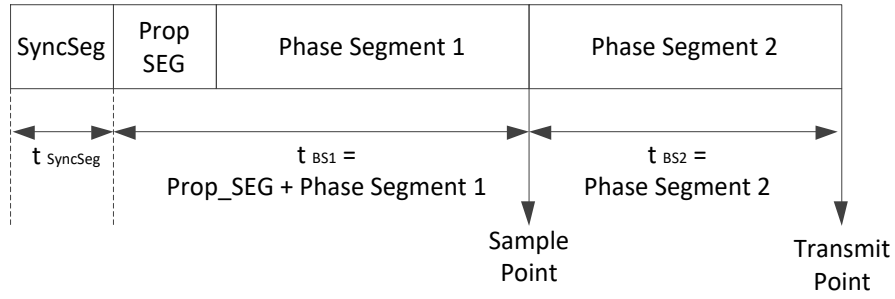
Table 6.38-15 Extended Message ID Filter Element Description

6.38.5.8 Bit Timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in Figure 6.38-12, its operation may be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum (1 x tq).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable between 1 and 16 time quanta, but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but may also be automatically shortened to compensate for negative phase drifts.



Note: BS = Bit segment

Figure 6.38-12 Bit Timing

The baud rate is the inverse of bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components. Figure 6.38-12 indicates that bit time = $t_{SyncSeg} + t_{BS1} + t_{BS2}$, where:

for the nominal bit time

– tq = (NBRP (CANFD_NBTP[24:16])+ 1) * $t_{canfd_tq_clk}$

– $t_{SyncSeg}$ = 1 tq

– t_{BS1} = tq * (NTSEG1(CANFD_NBTP[15:8]) + 1)

– t_{BS2} = tq * (NTSEG2(CANFD_NBTP[6:0]) + 1)

for the data bit time

– tq = (DBRP(CANFD_DBTP[20:16]) + 1) * $t_{canfd_tq_clk}$

– $t_{SyncSeg}$ = 1 tq

– t_{BS1} = tq * (DTSEG1(CANFD_DBTP[12:8]) + 1)

– t_{BS2} = tq * (DTSEG2(CANFD_DBTP[7:4]) + 1)

The (Re)Synchronization Jump Width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SE.G. BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SE.G. BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the Bit Timing register is only possible while the device is in Standby mode. Registers CANFD_DBTP and CANFD_NBTP (dedicated, respectively, to data and nominal bit timing) are only accessible when CCE (CANFD_CCCR[1]) and INIT (CANFD_CCCR[0]) are set.

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

6.38.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
CANFD Base Address: **CANFDn_BA = 0x4002_0000+ (0x4000 *n)** **n=0, 1, 2, 3**				
CANFD_DBTP	CANFDn_BA+0x00C	R	Data Bit Timing & Prescaler Register (P*)	0x0000_0A33
CANFD_TEST	CANFDn_BA+0x010	R	Test Register (P*)	0x0000_0000
CANFD_RWD	CANFDn_BA+0x014	R	RAM Watchdog (P*)	0x0000_0000
CANFD_CCCR	CANFDn_BA+0x018	R/W	CC Control Register (Pp*)	0x0000_0001
CANFD_NBTP	CANFDn_BA+0x01C	R	Nominal Bit Timing & Prescaler Register (P*)	0x0600_0A03
CANFD_TSCC	CANFDn_BA+0x020	R	Timestamp Counter Configuration (P*)	0x0000_0000
CANFD_TSCV	CANFDn_BA+0x024	R	Timestamp Counter Value (C*)	0x0000_0000
CANFD_TOCC	CANFDn_BA+0x028	R	Timeout Counter Configuration (P*)	0xFFFF_0000
CANFD_TOCV	CANFDn_BA+0x02C	R	Timeout Counter Value (C*)	0x0000_FFFF
CANFD_ECR	CANFDn_BA+0x040	R	Error Counter Register (X*)	0x0000_0000
CANFD_PSR	CANFDn_BA+0x044	R	Protocol Status Register (XS*)	0x0000_0707
CANFD_TDCR	CANFDn_BA+0x048	R	Transmitter Delay Compensation Register (P*)	0x0000_0000
CANFD_IR	CANFDn_BA+0x050	R/W	Interrupt Register	0x0000_0000
CANFD_IE	CANFDn_BA+0x054	R/W	Interrupt Enable	0x0000_0000
CANFD_ILS	CANFDn_BA+0x058	R/W	Interrupt Line Select	0x0000_0000
CANFD_ILE	CANFDn_BA+0x05C	R/W	Interrupt Line Enable	0x0000_0000
CANFD_GFC	CANFDn_BA+0x080	R	Global Filter Configuration (P*)	0x0000_0000
CANFD_SIDFC	CANFDn_BA+0x084	R	Standard ID Filter Configuration (P*)	0x0000_0000
CANFD_XIDFC	CANFDn_BA+0x088	R	Extended ID Filter Configuration (P*)	0x0000_0000
CANFD_XIDAM	CANFDn_BA+0x090	R	Extended ID AND Mask (P*)	0x1FFF_FFFF
CANFD_HPMS	CANFDn_BA+0x094	R	High Priority Message Status	0x0000_0000
CANFD_NDAT1	CANFDn_BA+0x098	R/W	New Data 1	0x0000_0000
CANFD_NDAT2	CANFDn_BA+0x09C	R/W	New Data 2	0x0000_0000
CANFD_RXF0C	CANFDn_BA+0x0A0	R	Rx FIFO 0 Configuration (P*)	0x0000_0000
CANFD_RXF0S	CANFDn_BA+0x0A4	R	Rx FIFO 0 Status	0x0000_0000
CANFD_RXF0A	CANFDn_BA+0x0A8	R/W	Rx FIFO 0 Acknowledge	0x0000_0000

CANFD_RXBC	CANFDn_BA+0x0AC	R	Rx Buffer Configuration (P*)	0x0000_0000
CANFD_RXF1C	CANFDn_BA+0x0B0	R	Rx FIFO 1 Configuration (P*)	0x0000_0000
CANFD_RXF1S	CANFDn_BA+0x0B4	R	Rx FIFO 1 Status	0x0000_0000
CANFD_RXF1A	CANFDn_BA+0x0B8	R/W	Rx FIFO 1 Acknowledge	0x0000_0000
CANFD_RXESC	CANFDn_BA+0x0BC	R	Rx Buffer / FIFO Element Size Configuration (P*)	0x0000_0000
CANFD_TXBC	CANFDn_BA+0x0C0	R	Tx Buffer Configuration (P*)	0x0000_0000
CANFD_TXFQS	CANFDn_BA+0x0C4	R	Tx FIFO/Queue Status	0x0000_0000
CANFD_TXESC	CANFDn_BA+0x0C8	R	Tx Buffer Element Size Configuration (P*)	0x0000_0000
CANFD_TXBRP	CANFDn_BA+0x0CC	R	Tx Buffer Request Pending	0x0000_0000
CANFD_TXBAR	CANFDn_BA+0x0D0	R/W	Tx Buffer Add Request	0x0000_0000
CANFD_TXBCR	CANFDn_BA+0x0D4	R/W	Tx Buffer Cancellation Request	0x0000_0000
CANFD_TXBTO	CANFDn_BA+0x0D8	R	Tx Buffer Transmission Occurred	0x0000_0000
CANFD_TXBCF	CANFDn_BA+0x0DC	R	Tx Buffer Cancellation Finished	0x0000_0000
CANFD_TXBTIE	CANFDn_BA+0x0E0	R/W	Tx Buffer Transmission Interrupt Enable	0x0000_0000
CANFD_TXBCIE	CANFDn_BA+0x0E4	R/W	Tx Buffer Cancellation Finished Interrupt Enable	0x0000_0000
CANFD_TXEFC	CANFDn_BA+0x0F0	R	Tx Event FIFO Configuration (P*)	0x0000_0000
CANFD_TXEFS	CANFDn_BA+0x0F4	R	Tx Event FIFO Status	0x0000_0000
CANFD_TXEFA	CANFDn_BA+0x0F8	R/W	Tx Event FIFO Acknowledge	0x0000_0000

Note:

R = Read, S = Set on read, X = Reset on read, W = Write, P = Protected write, p = Protected set, C = Clear/preset on write, r = release, d = date

6.38.7 Register Description

Data Bit Timing & Prescaler Register (CANFD_DBTP)

Register	Offset	R/W	Description	Reset Value
CANFD_DBTP	CANFDn_BA+0x00C	R	Data Bit Timing & Prescaler Register (P*)	0x0000_0A33

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
TDC	Reserved		DBRP				
15	14	13	12	11	10	9	8
Reserved			DTSEG1				
7	6	5	4	3	2	1	0
DTSEG2				DSJW			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**TDC**	**Transmitter Delay Compensation** *0*=Transmitter Delay Compensation disabled. *1*=Transmitter Delay Compensation enabled.
[22:21]	**Reserved**	Reserved.
[20:16]	**DBRP**	**Data Bit Rate Prescaler** The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 31. When **TDC** = '1', the range is limited to 0,1. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[15:13]	**Reserved**	Reserved.
[12:8]	**DTSEG1**	**Data time segment before sample point** Valid values are 0 to 31. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. t_{BS1} = (DTSEG1 + 1) x tq.
[7:4]	**DTSEG2**	**Data time segment after sample point** Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. t_{BS2} = (DTSEG2 + 1) x tq.
[3:0]	**DSJW**	**Data Re-Synchronization Jump Width** Valid values are 0 to 15. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. t_{SJW} = (DSJW + 1) x tq.

TEST Register (CANFD_TEST)

Register	Offset	R/W	Description	Reset Value
CANFD_TEST	CANFDn_BA+0x010	R	Test Register (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
RX	TX		LBCK	Reserved			

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**RX**	**Receive Pin** Monitors the actual value of pin CANx_RXD 0 = The CAN bus is dominant (CANx_RXD = 0). 1 = The CAN bus is recessive (CANx_RXD = 1).
[6:5]	**TX**	**Control of Transmit Pin** 00 = Reset value, CANx_TXD controlled by the CAN Core, updated at the end of the CAN bit time. 01 = Sample Point can be monitored at pin CANx_TXD. 10 = Dominant (‘0’) level at pin CANx_TXD. 11 = Recessive (‘1’) level at pin CANx_TXD.
[4]	**LBCK**	**Loop Back Mode** 0 = Reset value, Loop Back Mode is disabled. 1 = Loop Back Mode is enabled (refer to 6.38.5.1 TEST Mode).
[3:0]	**Reserved**	Reserved.

RAM Watchdog Register (CANFD_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured by the WDC (CANFD_RWD[7:0]) bits. The counter is reloaded with WDC bits when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to zero, the counter stops and interrupt flag WDI (CANFD_IR[26]) bit is set.

Register	Offset	R/W	Description	Reset Value
CANFD_RWD	CANFDn_BA+0x014	R	RAM Watchdog (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
WDV							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**WDV**	**Watchdog Value** Actual Message RAM Watchdog Counter Value.
[7:0]	**WDC**	**Watchdog Configuration** Start value of the Message RAM Watchdog Counter. With the reset value of 00 the counter is disabled.

CC Control Register (CANFD_CCCR)

Register	Offset	R/W	Description	Reset Value
CANFD_CCCR	CANFDn_BA+0x018	R/W	CC Control Register (Pp*)	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NISO	TXP	EFBI	PXHD	Reserved		BRSE	FDOE
7	6	5	4	3	2	1	0
TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**NISO**	**Non ISO Operation** If this bit is set, the CAN FD controller uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0. 0 = CAN FD frame format according to ISO 11898-1:2015. 1 = CAN FD frame format according to Bosch CAN FD Specification V1.0.
[14]	**TXP**	**Transmit Pause** If this bit is set, the CAN FD controller pauses for two CAN bit times before starting the next transmission after itself has successfully transmitted a frame (refer to 6.38.5.5). 0 = Transmit pause disabled. 1 = Transmit pause enabled.
[13]	**EFBI**	**Edge Filtering during Bus Integration** 0 = Edge filtering disabled. 1 = Two consecutive dominant tq required to detect an edge f or hard synchronization.
[12]	**PXHD**	**Protocol Exception Handling Disable** 0 = Protocol exception handling enabled. 1 = Protocol exception handling disabled. **Note:** When protocol exception handling is disabled, the controller will transmit an error frame when it detects a protocol exception condition.
[11:10]	**Reserved**	Reserved.
[9]	**BRSE**	**Bit Rate Switch Enable** 0 = Bit rate switching for transmissions disabled. 1 = Bit rate switching for transmissions enabled. **Note:** When CAN FD operation is disabled FDOE = 0, BRSE is not evaluated.
[8]	**FDOE**	**FD Operation Enable** 0 = FD operation disabled. 1 = FD operation enabled.

[7]	**TEST**	**Test Mode Enable** 0 = Normal operation, register TEST holds reset values. 1 = Test Mode, write access to register TEST enabled.
[6]	**DAR**	**Disable Automatic Retransmission** 0 = Automatic retransmission of messages not transmitted successfully enabled. 1 = Automatic retransmission disabled.
[5]	**MON**	**Bus Monitoring Mode** Bit MON can only be set by the Host when both CCE and INIT are set to 1. The bit can be reset by the Host at any time. 0 = Bus Monitoring Mode is disabled. 1 = Bus Monitoring Mode is enabled.
[4]	**CSR**	**Clock Stop Request** 0 = No clock stop is requested. 1 = Clock stop requested. When clock stop is requested, first INIT and then CSA will be set after all pending transfer requests have been completed and the CAN bus reached idle.
[3]	**CSA**	**Clock Stop Acknowledge** 0 = No clock stop acknowledged. 1 = The Controller may be set in power down by stopping AHB clock and CAN Core clock.
[2]	**ASM**	**Restricted Operation Mode** Bit ASM can only be set by the Host when both CCE and INIT are set to 1. The bit can be reset by the software at any time. This bit will be set automatically set to 1 when the Tx handler was not able to read data from the message RAM in time. For a description of the Restricted Operation Mode refer to **Restricted Operation Mode**. 0 = Normal CAN operation. 1 = Restricted Operation Mode active.
[1]	**CCE**	**Configuration Change Enable** 0 = The CPU has no write access to the protected configuration registers. 1 = The CPU has write access to the protected configuration registers (while INIT (CANFD_CCCR[0]) = 1).
[0]	**INIT**	**Initialization** 0 = Normal Operation. 1 = Initialization is started. **Note:** Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

Nominal Bit Timing & Prescaler Register (CANFD_NBTP)

Register	Offset	R/W	Description	Reset Value
CANFD_NBTP	CANFDn_BA+0x01C	R	Nominal Bit Timing & Prescaler Register (P*)	0x0600_0A03

31	30	29	28	27	26	25	24
NSJW							NBRP
23	22	21	20	19	18	17	16
NBRP							
15	14	13	12	11	10	9	8
NTSEG1							
7	6	5	4	3	2	1	0
Reserved	NTSEG2						

Bits	Description	
[31:25]	**NSJW**	**Nominal Re-Synchronization Jump Width** Valid values are 0 to 127. Should be smaller than NTSEG2. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used. t_{SJW} = (NSJW + 1) x tq.
[24:16]	**NBRP**	**Nominal Bit Rate Prescaler** The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Bit Rate Prescaler are 0 to 511. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used
[15:8]	**NTSEG1**	**Nominal Time segment before sample point** Valid values are 1 to 255. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. t_{BS1} = (NTSEG1 + 1) x tq.
[7]	**Reserved**	Reserved.
[6:0]	**NTSEG2**	**Nominal Time segment after sample point** 0x01-0x7F Valid values are 1 to 127. The actual interpretation by the hardware of this value is such that one more than the programmed value is used. t_{BS2} = (NTSEG2 + 1) x tq. **Note:** With a CAN Core clock (cclk) of 8 MHz, the reset value of 0x06000A03 configures the controller for a bit rate of 500 kBit/s.

Timestamp Counter Configuration Register (CANFD_TSCC)

Register	Offset	R/W	Description	Reset Value
CANFD_TSCC	CANFDn_BA+0x020	R	Timestamp Counter Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				TCP			
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						TSS	

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:16]	**TCP**	**Timestamp Counter Prescaler** Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[15:2]	**Reserved**	Reserved.
[1:0]	**TSS**	**Timestamp Select** 00 = Timestamp counter value always 0x0000. 01 = Timestamp counter value incremented according to TCP. 10 = Reserved 11 = Same as “00”.

Timestamp Counter Register (CANFD_TSCV)

Register	Offset	R/W	Description	Reset Value
CANFD_TSCV	CANFDn_BA+0x024	R	Timestamp Counter Value (C*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TSC							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TSC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TSC**	**Timestamp Counter** The internal Timestamp Counter value is captured on start of frame (both Rx and Tx). When TSS (CANFD_TSCC[[1:0]) = 2'b01, the Timestamp Counter is incremented in multiples of CAN bit times [1...16] depending on the configuration of TCP (CANFD_TSCC[19:16]). A wrap around sets interrupt flag TSW (CANFD_IR[16]). Write access resets the counter to zero. **Note:** A "wrap around" is a change of the Timestamp Counter value from non-zero to zero not caused by write access to CANFD_TSCV.

Timeout Counter Configuration Register (CANFD_TOCC)

Register	Offset	R/W	Description	Reset Value
CANFD_TOCC	CANFDn_BA+0x028	R	Timeout Counter Configuration (P*)	0xFFFF_0000

31	30	29	28	27	26	25	24
TOP							
23	22	21	20	19	18	17	16
TOP							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					TOS		ETOC

Bits	Description	
[31:16]	**TOP**	**Timeout Period** Start value of the Timeout Counter (down-counter). Configures the Timeout Period.
[15:3]	**Reserved**	Reserved.
[2:1]	**TOS**	**Timeout Select** When operating in Continuous mode, a write to CANFD_TOCV presets the counter to the value configured by TOP (CANFD_TOCC[31:16]) and continues down-counting. When the Timeout Counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP (CANFD_TOCC[31:16]). Down-counting is started when the first FIFO element is stored. 00 = Continuous operation. 01 = Timeout controlled by Tx Event FIFO. 10 = Timeout controlled by Rx FIFO 0. 11 = Timeout controlled by Rx FIFO 1.
[0]	**ETOC**	**Enable Timeout Counter** 0 = Timeout Counter disabled. 1 = Timeout Counter enabled. **Note:** For use of timeout function with CAN FD refer to 6.38.5.3.

Timeout Counter Value Register (CANFD_TOCV)

Register	Offset	R/W	Description	Reset Value
CANFD_TOCV	CANFDn_BA+0x02C	R	Timeout Counter Value (C*)	0x0000_FFFF

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TOC							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TOC**	**Timeout Counter** The filed is decremented in multiples of CAN bit times [1...16] depending on the configuration of TCP (CANFD_TSCC[19:16]). When decremented to zero, interrupt flag TOO (CANFD_IR[18]) is set and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS (CANFD_TOCC[1:0]).

Error Counter Register (CANFD_ECR)

Register	Offset	R/W	Description	Reset Value
CANFD_ECR	CANFDn_BA+0x040	R	Error Counter Register (X*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CEL							
15	14	13	12	11	10	9	8
RP	REC						
7	6	5	4	3	2	1	0
TEC							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**CEL**	**CAN Error Logging** The counter is incremented each time when a CAN protocol error causes the 8-bit Transmit Error Counter TEC or the 7-bit Receive Error Counter REC to be incremented. The counter is also incremented when the Bus_Off limit is reached. It is not incremented when only RP is set without changing REC. The increment of CEL follows after the increment of REC or TEC. The counter is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag ELO (CANFD_IR[22]).
[15]	**RP**	**Receive Error Passive** 0 = The Receive Error Counter is below the error passive level of 128. 1 = The Receive Error Counter has reached the error passive level of 128.
[14:8]	**REC**	**Receive Error Counter** Actual state of the Receive Error Counter, values between 0 and 127.
[7:0]	**TEC**	**Transmit Error Counter** Actual state of the Transmit Error Counter, values between 0 and 255. **Note:** When ASM (CANFD_CCCR[2]) is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

Protocol Status Register (CANFD_PSR)

Register	Offset	R/W	Description	Reset Value
CANFD_PSR	CANFDn_BA+0x044	R	Protocol Status Register (XS*)	0x0000_0707

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	TDCV						
15	14	13	12	11	10	9	8
Reserved	PXE	RFDF	RBRS	RESI	DLEC		
7	6	5	4	3	2	1	0
BO	EW	EP	ACT		LEC		

Bits	Description	
[31:23]	**Reserved**	Reserved.
[22:16]	**TDCV**	**Transmitter Delay Compensation Value** Position of the secondary sample point, defined by the sum of the measured delay from CANx_TXD to CANx_RXD and TDCO (CANFD_TDCR[[14:8]). The SSP position is, in the data phase, the number of minimum time quata (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.
[15]	**Reserved**	Reserved.
[14]	**PXE**	**Protocol Exception Event** 0 = No protocol exception event occurred since last read access. 1 = Protocol exception event occurred.
[13]	**RFDF**	**Received a CAN FD Message** This bit is set independent of acceptance filtering. 0 = Since this bit was reset by the CPU, no CAN FD message has been received. 1 = Message in CAN FD format with FDF flag set has been received. **Note:** Byte access: Reading byte 0 will reset RFDF, reading bytes 3/2/1 has no impact.
[12]	**RBRS**	**BRS flag of last received CAN FD Message** This bit is set together with RFDF, independent of acceptance filtering. 0 = Last received CAN FD message did not have its BRS flag set. 1 = Last received CAN FD message had its BRS flag set. **Note:** Byte access: Reading byte 0 will reset RBRS, reading bytes 3/2/1 has no impact.
[11]	**RESI**	**ESI flag of last received CAN FD Message** This bit is set together with RFDF, independent of acceptance filtering. 0 = Last received CAN FD message did not have its ESI flag set. 1 = Last received CAN FD message had its ESI flag set.
[10:8]	**DLEC**	**Data Phase Last Error Code** Type of last error that occurred in the data phase of a CAN FD format frame with its BRS flag set. Coding is the same as for LEC. This field will be cleared to zero when a CAN FD format frame with its BRS flag set has been transferred (reception or transmission) without error.

[7]	**BO**	**Bus_Off Status** 0 = The CAN FD controller is not Bus_Off. 1 = The CAN FD controller is in Bus_Off state.
[6]	**EW**	**Warning Status** 0 = Both error counters are below the Error_Warning limit of 96. 1 = At least one of error counter has reached the Error_Warning limit of 96.
[5]	**EP**	**Error Passive** 0 = The CAN FD controller is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected. 1 = The CAN FD controller is in the Error_Passive state.
[4:3]	**ACT**	**Activity** Monitors the module’s CAN communication state. 00 = Synchronizing - node is synchronizing on CAN communication. 01 = Idle - node is neither receiver nor transmitter. 10 = Receiver - node is operating as receiver. 11 = Transmitter - node is operating as transmitter.
[2:0]	**LEC**	**Last Error Code** The LEC indicates the type of the last error to occur on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error. 000 = No Error: No error occurred since LEC has been reset by successful reception or transmission. 001 = Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed. 010 = Form Error: A fixed format part of a received frame has the wrong format. 011 = AckError: The message transmitted by the CANFD CONTROLLER was not acknowledged by another node. 100 = Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant. 101 = Bit0Error : During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed). 110 = CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data. 111 = NoChange: Any read access to the Protocol Status Register re-initializes the LEC to 7.When the LEC shows the value 7, no CAN bus event was detected since the last CPU read access to the Protocol Status Register.

Note: When a frame in CAN FD format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) will be shown in DLEC instead of LEC. An error in a fixed stuff bit of a CAN FD CRC sequence will be shown as a Form Error, not Stuff Error.

Note: The Bus_Off recovery sequence (refer to ISO 11898-1:2015) cannot be shortened by setting or resetting INIT (CANFD_CCCR[0]). If the device goes Bus_Off, it will set INIT (CANFD_CCCR[0]) of its own accord, stopping all bus activities. Once INIT (CANFD_CCCR[0]) has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the Error Management Counters will be reset. During the waiting time after the resetting of INIT (CANFD_CCCR[0]), each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to LEC (CANFD_PSR[2:0]), enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. REC(CANFD_ECR[14:8]) is used to count these sequences.

Transmitter Delay Compensation Register (CANFD_TDCR)

Register	Offset	R/W	Description	Reset Value
CANFD_TDCR	CANFDn_BA+0x048	R	Transmitter Delay Compensation Register (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	TDCO						
7	6	5	4	3	2	1	0
Reserved	TDCF						

Bits	Description	
[31:15]	**Reserved**	Reserved.
[14:8]	**TDCO**	**Transmitter Delay Compensation SSP Offset** Offset value defining the distance between the measured delay from CANx_TXD to CANx_RXD and the secondary sample point. Valid values are 0 to 127 mtq.
[7]	**Reserved**	Reserved.
[6:0]	**TDCF**	**Transmitter Delay Compensation Filter Window Length** Defines the minimum value for the SSP position, dominant edges on CANx_RXD that would result in an earlier SSP position are ignored for transmitter delay measurement. The feature is enabled when TDCF is configured to a value greater than TDCO. Valid values are 0 to 127 mtq.

Interrupt Register (CANFD_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register.

Register	Offset	R/W	Description	Reset Value
CANFD_IR	CANFDn_BA+0x050	R/W	Interrupt Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		ARA	PED	PEA	WDI	BO	EW
23	22	21	20	19	18	17	16
EP	ELO	Reserved		DRX	TOO	MRAF	TSW
15	14	13	12	11	10	9	8
TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM
7	6	5	4	3	2	1	0
RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**ARA**	**Access to Reserved Address** 0 = No access to reserved address occurred. 1 = Access to reserved address occurred.
[28]	**PED**	**Protocol Error in Data Phase** 0 = No protocol error in data phase. 1= Protocol error in data phase detected (DLEC (CANFD_PSR[10:8]) no equal 0 or 7). **Note:** Data bit time is used
[27]	**PEA**	**Protocol Error in Arbitration Phase** 0 = No protocol error in arbitration phase. 1 = Protocol error in arbitration phase detected (LEC (CANFD_PSR[2:0]) no equal 0 or 7). **Note:** Nominal bit time is used
[26]	**WDI**	**Watchdog Interrupt** 0 = No Message RAM Watchdog event occurred. 1 = Message RAM Watchdog event due to missing READY.
[25]	**BO**	**Bus_Off Status** 0 = Bus_Off status unchanged. 1 = Bus_Off status changed.
[24]	**EW**	**Warning Status** 0 = Error_Warning status unchanged. 1 = Error_Warning status changed.
[23]	**EP**	**Error Passive**

		0 = Error_Passive status unchanged. 1 = Error_Passive status changed.
[22]	**ELO**	**Error Logging Overflow** 0 = CAN Error Logging Counter did not overflow. 1 = Overflow of CAN Error Logging Counter occurred.
[21:20]	**Reserved**	Reserved.
[19]	**DRX**	**Message stored to Dedicated Rx Buffer** The flag is set whenever a received message has been stored into a dedicated Rx Buffer. 0 = No Rx Buffer updated. 1 = At least one received message stored into an Rx Buffer.
[18]	**TOO**	**Timeout Occurred** 0 = No timeout. 1 = Timeout reached.
[17]	**MRAF**	**Message RAM Access Failure** The flag is set, when the Rx Handler • Has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx Handler starts processing of the following message. • Was not able to write a message to the Message RAM. In this case message storage is aborted. In both cases the FIFO put index is not updated resp. the New Data flag for a dedicated Rx Buffer is not set, a partly stored message is overwritten when the next message is stored to this location. The flag is also set when the Tx Handler was not able to read a message from the Message RAM in time. In this case message transmission is aborted. In case of a Tx Handler access failure the CAN FD controller is switched into Restricted Operation Mode (refer to **Restricted Operation Mode**). To leave Restricted Operation Mode, the Host CPU has to reset ASM (CANFD_CCCR[2]). 0 = No Message RAM access failure occurred. 1 = Message RAM access failure occurred.
[16]	**TSW**	**Timestamp Wraparound** 0 = No timestamp counter wrap-around. 1 = Timestamp counter wrapped around.
[15]	**TEFL**	**Tx Event FIFO Element Lost** 0 = No Tx Event FIFO element lost. 1 = Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
[14]	**TEFF**	**Tx Event FIFO Full** 0 = Tx Event FIFO is not full. 1 = Tx Event FIFO is full.
[13]	**TEFW**	**Tx Event FIFO Watermark Reached** 0 = Tx Event FIFO fill level below watermark. 1 = Tx Event FIFO fill level reached watermark.
[12]	**TEFN**	**Tx Event FIFO New Entry** 0 = Tx Event FIFO unchanged. 1 = Tx Handler wrote Tx Event FIFO element.
[11]	**TFE**	**Tx FIFO Empty** 0 = Tx FIFO non-empty. 1 = Tx FIFO empty.
[10]	**TCF**	**Transmission Cancellation Finished**

		0 = No transmission cancellation finished. 1 = Transmission cancellation finished.
[9]	**TC**	**Transmission Completed** 0 = No transmission completed. 1 = Transmission completed.
[8]	**HPM**	**High Priority Message** 0 = No high priority message received. 1 = High priority message received.
[7]	**RF1L**	**Rx FIFO 1 Message Lost** 0 = No Rx FIFO 1 message lost. 1 = Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero.
[6]	**RF1F**	**Rx FIFO 1 Full** 0 = Rx FIFO 1 is not full. 1 = Rx FIFO 1 is full.
[5]	**RF1W**	**Rx FIFO 1 Watermark Reached** 0 = Rx FIFO 1 fill level below watermark. 1 = Rx FIFO 1 fill level reached watermark.
[4]	**RF1N**	**Rx FIFO 1 New Message** 0 = No new message written to Rx FIFO 1. 1 = New message written to Rx FIFO 1.
[3]	**RF0L**	**Rx FIFO 0 Message Lost** 0 = No Rx FIFO 0 message lost. 1 = Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero.
[2]	**RF0F**	**Rx FIFO 0 Full** 0 = Rx FIFO 0 is not full. 1 = Rx FIFO 0 is full.
[1]	**RF0W**	**Rx FIFO 0 Watermark Reached** 0 = Rx FIFO 0 fill level below watermark. 1 = Rx FIFO 0 fill level reached watermark.
[0]	**RF0N**	**Rx FIFO 0 New Message** 0 = No new message written to Rx FIFO 0. 1 = New message written to Rx FIFO 0.

Interrupt Enable Register (CANFD_IE)

Register	Offset	R/W	Description	Reset Value
CANFD_IE	CANFDn_BA+0x054	R/W	Interrupt Enable	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		ARAE	PEDE	PEAE	WDIE	BOE	EWE
23	22	21	20	19	18	17	16
EPE	ELOE	BEUE	BECE	DRXE	TOOE	MRAFE	TSWE
15	14	13	12	11	10	9	8
TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME
7	6	5	4	3	2	1	0
RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**ARAE**	**Access to Reserved Address Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[28]	**PEDE**	**Protocol Error in Data Phase Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[27]	**PEAE**	**Protocol Error in Arbitration Phase Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[26]	**WDIE**	**Watchdog Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[25]	**BOE**	**Bus_Off Status Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[24]	**EWE**	**Warning Status Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[23]	**EPE**	**Error Passive Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[22]	**ELOE**	**Error Logging Overflow Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[21]	**BEUE**	**Bit Error Uncorrected Interrupt Enable**

		0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[20]	**BECE**	**Bit Error Corrected Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[19]	**DRXE**	**Message stored to Dedicated Rx Buffer Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[18]	**TOOE**	**Timeout Occurred Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[17]	**MRAFE**	**Message RAM Access Failure Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[16]	**TSWE**	**Timestamp Wraparound Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[15]	**TEFLE**	**Tx Event FIFO Event Lost Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[14]	**TEFFE**	**Tx Event FIFO Full Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[13]	**TEFWE**	**Tx Event FIFO Watermark Reached Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[12]	**TEFNE**	**Tx Event FIFO New Entry Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[11]	**TFEE**	**Tx FIFO Empty Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[10]	**TCFE**	**Transmission Cancellation Finished Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[9]	**TCE**	**Transmission Completed Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[8]	**HPME**	**High Priority Message Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[7]	**RF1LE**	**Rx FIFO 1 Message Lost Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.

[6]	**RF1FE**	**Rx FIFO 1 Full Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[5]	**RF1WE**	**Rx FIFO 1 Watermark Reached Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[4]	**RF1NE**	**Rx FIFO 1 New Message Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[3]	**RF0LE**	**Rx FIFO 0 Message Lost Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[2]	**RF0FE**	**Rx FIFO 0 Full Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[1]	**RF0WE**	**Rx FIFO 0 Watermark Reached Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[0]	**RF0NE**	**Rx FIFO 0 New Message Interrupt Enable** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.

Interrupt Line Select Register (CANFD_ILS)

This register assigns an interrupt generated by a specific interrupt flag from the CANFD_IR register to one of the two module interrupt lines.

Register	Offset	R/W	Description	Reset Value
CANFD_ILS	CANFDn_BA+0x058	R/W	Interrupt Line Select	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		ARAL	PEDL	PEAL	WDIL	BOL	EWL
23	22	21	20	19	18	17	16
EPL	ELOL	Reserved		DRXL	TOOL	MRAFL	TSWL
15	14	13	12	11	10	9	8
TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML
7	6	5	4	3	2	1	0
RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**ARAL**	**Access to Reserved Address Line**
[28]	**PEDL**	**Protocol Error in Data Phase Line**
[27]	**PEAL**	**Protocol Error in Arbitration Phase Line**
[26]	**WDIL**	**Watchdog Interrupt Line**
[25]	**BOL**	**Bus_Off Status Interrupt Line**
[24]	**EWL**	**Warning Status Interrupt Line**
[23]	**EPL**	**Error Passive Interrupt Line**
[22]	**ELOL**	**Error Logging Overflow Interrupt Line**
[21:20]	**Reserved**	Reserved.
[19]	**DRXL**	**Message stored to Dedicated Rx Buffer Interrupt Line**
[18]	**TOOL**	**Timeout Occurred Interrupt Line**
[17]	**MRAFL**	**Message RAM Access Failure Interrupt Line**
[16]	**TSWL**	**Timestamp Wraparound Interrupt Line**
[15]	**TEFLL**	**Tx Event FIFO Event Lost Interrupt Line**
[14]	**TEFFL**	**Tx Event FIFO Full Interrupt Line**
[13]	**TEFWL**	**Tx Event FIFO Watermark Reached Interrupt Line**
[12]	**TEFNL**	**Tx Event FIFO New Entry Interrupt Line**
[11]	**TFEL**	**Tx FIFO Empty Interrupt Line**

[10]	**TCFL**	**Transmission Cancellation Finished Interrupt Line**
[9]	**TCL**	**Transmission Completed Interrupt Line**
[8]	**HPML**	**High Priority Message Interrupt Line**
[7]	**RF1LL**	**Rx FIFO 1 Message Lost Interrupt Line**
[6]	**RF1FL**	**Rx FIFO 1 Full Interrupt Line**
[5]	**RF1WL**	**Rx FIFO 1 Watermark Reached Interrupt Line**
[4]	**RF1NL**	**Rx FIFO 1 New Message Interrupt Line**
[3]	**RF0LL**	**Rx FIFO 0 Message Lost Interrupt Line**
[2]	**RF0FL**	**Rx FIFO 0 Full Interrupt Line**
[1]	**RF0WL**	**Rx FIFO 0 Watermark Reached Interrupt Line**
[0]	**RF0NL**	**Rx FIFO 0 New Message Interrupt Line** 0 = Interrupt assigned to CAN interrupt line 0. 1 = Interrupt assigned to CAN interrupt line 1.

Interrupt Line Register (CANFD_ILE)

Register	Offset	R/W	Description	Reset Value
CANFD_ILE	CANFDn_BA+0x05C	R/W	Interrupt Line Enable	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**EINT1**	**EINT0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**EINT1**	**Enable Interrupt Line 1** 0 = Interrupt line m_can_int1 disabled. 1 = Interrupt line m_can_int1 enabled.
[0]	**EINT0**	**Enable Interrupt Line 0** 0 = Interrupt line m_can_int0 disabled. 1 = Interrupt line m_can_int0 enabled.

Global Filter Configuration Register (CANFD_GFC)

Register	Offset	R/W	Description	Reset Value
CANFD_GFC	CANFDn_BA+0x080	R	Global Filter Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		ANFS		ANFE		RRFS	RRFE

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:4]	**ANFS**	**Accept Non-matching Frames Standard** Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated. 00 = Accept in Rx FIFO 0. 01 = Accept in Rx FIFO 1. 10 = Reject. 11 = Reject.
[3:2]	**ANFE**	**Accept Non-matching Frames Extended** Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated. 00 = Accept in Rx FIFO 0. 01 = Accept in Rx FIFO 1. 10 = Reject. 11 = Reject.
[1]	**RRFS**	**Reject Remote Frames Standard** 0= Filter remote frames with 11-bit standard IDs. 1= Reject all remote frames with 11-bit standard IDs.
[0]	**RRFE**	**Reject Remote Frames Extended** 0= Filter remote frames with 29-bit extended IDs. 1= Reject all remote frames with 29-bit extended IDs.

Standard ID Filter Configuration Register (CANFD_SIDFC)

Register	Offset	R/W	Description	Reset Value
CANFD_SIDFC	CANFDn_BA+0x084	R	Standard ID Filter Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
LSS							
15	14	13	12	11	10	9	8
FLSSA							
7	6	5	4	3	2	1	0
FLSSA						Reserved	

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**LSS**	**List Size Standard** 0= No standard Message ID filter. 1-128 = Number of standard Message ID filter elements. >128= Values greater than 128 are interpreted as 128.
[15:2]	**FLSSA**	**Filter List Standard Start Address** Start address of standard Message ID filter list (32-bit word address, refer to Figure 6.38-11).
[1:0]	**Reserved**	Reserved.

Extended ID Filter Configuration Register (CANFD_XIDFC)

Register	Offset	R/W	Description	Reset Value
CANFD_XIDFC	CANFDn_BA+0x088	R	Extended ID Filter Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	LSE						
15	14	13	12	11	10	9	8
FLESA							
7	6	5	4	3	2	1	0
FLESA						Reserved	

Bits	Description	
[31:23]	**Reserved**	Reserved.
[22:16]	**LSE**	**List Size Extended** 0= No extended Message ID filter. 1-64= Number of extended Message ID filter elements. >64= Values greater than 64 are interpreted as 64.
[15:2]	**FLESA**	**Filter List Extended Start Address** Start address of extended Message ID filter list (32-bit word address, refer to Figure 6.38-11)
[1:0]	**Reserved**	Reserved.

Extended ID and Mask Register (CANFD_XIDAM)

Register	Offset	R/W	Description	Reset Value
CANFD_XIDAM	CANFDn_BA+0x090	R	Extended ID AND Mask (P*)	0x1FFF_FFFF

31	30	29	28	27	26	25	24
Reserved			EIDM				
23	22	21	20	19	18	17	16
EIDM							
15	14	13	12	11	10	9	8
EIDM							
7	6	5	4	3	2	1	0
EIDM							

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:0]	**EIDM**	**Extended ID Mask** For acceptance filtering of extended frames the Extended ID AND Mask is ANDed with the Message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to one the mask is not active.

High Priority Message Status Register (CANFD_HPMS)

Register	Offset	R/W	Description	Reset Value
CANFD_HPMS	CANFDn_BA+0x094	R	High Priority Message Status	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
FLST	FIDX						
7	6	5	4	3	2	1	0
MSI		BIDX					

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**FLST**	**Filter List** Indicates the filter list of the matching filter element. 0 = Standard Filter List. 1 = Extended Filter List.
[14:8]	**FIDX**	**Filter Index** Index of matching filter element. Range is 0 to LSS (CANFD_SIDFC[23:16]) - 1 or LSE (CANFD_XIDFC[22:16]) – 1
[7:6]	**MSI**	**Message Storage Indicator** 00 = No FIFO selected. 01 = FIFO message lost. 10 = Message stored in FIFO 0. 11 = Message stored in FIFO 1.
[5:0]	**BIDX**	**Buffer Index** Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

New Data 1 Register (CANFD_NDAT1)

Register	Offset	R/W	Description	Reset Value
CANFD_NDAT1	CANFDn_BA+0x098	R/W	New Data 1	0x0000_0000

31	30	29	28	27	26	25	24
NDn							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
NDn							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NDn							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NDn							

Bits	Description	
[31:0] n=0,1..31	**NDn**	**New Data** The register holds the New Data flags of Rx Buffers 0 to 31. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register. 0 = Rx Buffer not updated. 1 = Rx Buffer updated from new message.

New Data 2 Register (CANFD_NDAT2)

Register	Offset	R/W	Description	Reset Value
CANFD_NDAT2	CANFDn_BA+0x09C	R/W	New Data 2	0x0000_0000

31	30	29	28	27	26	25	24
NDn							
23	22	21	20	19	18	17	16
NDn							
15	14	13	12	11	10	9	8
NDn							
7	6	5	4	3	2	1	0
NDn							

Bits	Description	
[31:0] n=32,33..63	NDn	**New Data** The register holds the New Data flags of Rx Buffers 32 to 63. The flags are set when the respective Rx Buffer has been updated from a received frame. The flags remain set until the Host clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset will clear the register. 0 = Rx Buffer not updated. 1 = Rx Buffer updated from new message.

Rx FIFO 0 Configuration Register (CANFD_RXF0C)

Register	Offset	R/W	Description	Reset Value
CANFD_RXF0C	CANFDn_BA+0x0A0	R	Rx FIFO 0 Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
F0OM	F0WM						
23	22	21	20	19	18	17	16
Reserved	F0S						
15	14	13	12	11	10	9	8
F0SA							
7	6	5	4	3	2	1	0
F0SA						Reserved	

Bits	Description	
[31]	**F0OM**	**FIFO 0 Operation Mode** FIFO 0 can be operated in blocking or in overwrite mode (refer to **Rx FIFOs**). 0 = FIFO 0 blocking mode. 1 = FIFO 0 overwrite mode.
[30:24]	**F0WM**	**Rx FIFO 0 Watermark** 0= Watermark interrupt disabled. 1-64 = Level for Rx FIFO 0 watermark interrupt (RF0W(CANFD_IR[1]). >64 = Watermark interrupt disabled.
[23]	**Reserved**	Reserved.
[22:16]	**F0S**	**Rx FIFO 0 Size** 0= No Rx FIFO 0. 1-64= Number of Rx FIFO 0 elements. >64= Values greater than 64 are interpreted as 64. The Rx FIFO 0 elements are indexed from 0 to F0S-1
[15:2]	**F0SA**	**Rx FIFO 0 Start Address** Start address of Rx FIFO 0 in Message RAM (32-bit word address).
[1:0]	**Reserved**	Reserved.

Rx FIFO 0 Status Register (CANFD_RXF0S)

Register	Offset	R/W	Description	Reset Value
CANFD_RXF0S	CANFDn_BA+0x0A4	R	Rx FIFO 0 Status	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						RF0L	F0F
23	22	21	20	19	18	17	16
Reserved		F0PI					
15	14	13	12	11	10	9	8
Reserved		F0GI					
7	6	5	4	3	2	1	0
Reserved	F0FL						

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**RF0L**	**Rx FIFO 0 Message Lost** This bit is a copy of interrupt flagRF0L (CANFD_IR[3]). When CANFD_IR.RF0L is reset, this bit is also reset. 0 = No Rx FIFO 0 message lost. 1 = Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size zero. **Note:** Overwriting the oldest message when F0OM (CANFD_RXF0C[31]) = 1 will not set this flag.
[24]	**F0F**	**Rx FIFO 0 Full** 0= Rx FIFO 0 is not full. 1= Rx FIFO 0 is full.
[23:22]	**Reserved**	Reserved.
[21:16]	**F0PI**	**Rx FIFO 0 Put Index** Rx FIFO 0 write index pointer, range 0 to 63.
[15:14]	**Reserved**	Reserved.
[13:8]	**F0GI**	**Rx FIFO 0 Get Index** Rx FIFO 0 read index pointer, range 0 to 63.
[7]	**Reserved**	Reserved.
[6:0]	**F0FL**	**Rx FIFO 0 Fill Level** Number of elements stored in Rx FIFO 0, range 0 to 64

Rx FIFO 0 Acknowledge Register (CANFD_RXF0A)

Register	Offset	R/W	Description	Reset Value
CANFD_RXF0A	CANFDn_BA+0x0A8	R/W	Rx FIFO 0 Acknowledge	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**F0A**					

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**F0A**	**Rx FIFO 0 Acknowledge Index** After the Host has read a message or a sequence of messages from Rx FIFO 0 it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI. This will set the Rx FIFO 0 Get Index F0GI (CANFD_RXF0S[13:8]) to F0AI (CANFD_RXF0A[5:0]) + 1 and update the FIFO 0 Fill Level F0FL(CANFD_RXF0S[6:0]).

Rx Buffer Configuration Register (CANFD_RXBC)

Register	Offset	R/W	Description	Reset Value
CANFD_RXBC	CANFDn_BA+0x0AC	R	Rx Buffer Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RBSA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RBSA						**Reserved**	

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:2]	**RBSA**	**Rx Buffer Start Address** Configures the start address of the Rx Buffers section in the Message RAM (32-bit word address).
[1:0]	**Reserved**	Reserved.

Rx FIFO 1 Configuration Register (CANFD_RXF1C)

Register	Offset	R/W	Description	Reset Value
CANFD_RXF1C	CANFDn_BA+0x0B0	R	Rx FIFO 1 Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
F1OM	F1WM						
23	22	21	20	19	18	17	16
Reserved	F1S						
15	14	13	12	11	10	9	8
F1SA							
7	6	5	4	3	2	1	0
F1SA						Reserved	

Bits	Description	
[31]	**F1OM**	**FIFO 1 Operation Mode** FIFO 1 can be operated in blocking or in overwrite mode (refer to **Rx FIFOs**). 0= FIFO 1 blocking mode. 1= FIFO 1 overwrite mode.
[30:24]	**F1WM**	**Rx FIFO 1 Watermark** 0= Watermark interrupt disabled. 1-64 = Level for Rx FIFO 1 watermark interrupt (RF1W(CANFD_IR[5])). >64 = Watermark interrupt disabled.
[23]	**Reserved**	Reserved.
[22:16]	**F1S**	**Rx FIFO 1 Size** 0= No Rx FIFO 1. 1-64 = Number of Rx FIFO 1 elements. >64 = Values greater than 64 are interpreted as 64. The Rx FIFO 1 elements are indexed from 0 to F1S - 1
[15:2]	**F1SA**	**Rx FIFO 1 Start Address** Start address of Rx FIFO 1 in Message RAM (32-bit word address, refer to Figure 6.38-11).
[1:0]	**Reserved**	Reserved.

Rx FIFO 1 Status Register (CANFD_RXF1S)

Register	Offset	R/W	Description	Reset Value
CANFD_RXF1S	CANFDn_BA+0x0B4	R	Rx FIFO 1 Status	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						RF1L	F1F
23	22	21	20	19	18	17	16
Reserved		F1PI					
15	14	13	12	11	10	9	8
Reserved		F1GI					
7	6	5	4	3	2	1	0
Reserved	F1FL						

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**RF1L**	**Rx FIFO 1 Message Lost** This bit is a copy of interrupt flag RF1L(CANFD_IR[7]). When RF1L(CANFD_IR[7]) is reset, this bit is also reset. 0= No Rx FIFO 1 message lost. 1= Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size zero. **Note:** Overwriting the oldest message when F1OM (CANFD_RXF1C[31]) = 1 will not set this flag.
[24]	**F1F**	**Rx FIFO 1 Full** 0 = Rx FIFO 1 is not full. 1 = Rx FIFO 1 is full.
[23:22]	**Reserved**	**Rx FIFO 1 Get Index** Rx FIFO 1 read index pointer, range 0 to 63.
[21:16]	**F1PI**	**Rx FIFO 1 Fill Level** Number of elements stored in Rx FIFO 1, range 0 to 64.
[15:14]	**Reserved**	Reserved.
[13:8]	**F1GI**	**Rx FIFO 1 Get Index** Rx FIFO 1 read index pointer, range 0 to 63.
[7]	**Reserved**	Reserved.
[6:0]	**F1FL**	**Rx FIFO 1 Fill Level** Number of elements stored in Rx FIFO 1, range 0 to 64

Rx FIFO 1 Acknowledge Register (CANFD_RXF1A)

Register	Offset	R/W	Description	Reset Value
CANFD_RXF1A	CANFDn_BA+0x0B8	R/W	Rx FIFO 1 Acknowledge	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		F1AI					

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**F1AI**	**Rx FIFO 1 Acknowledge Index** After the Host has read a message or a sequence of messages from Rx FIFO 1 it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI. This will set the Rx FIFO 1 Get Index F1GI (CANFD_RXF1S[13:8]) to F1AI (CANFD_RXF1A[5:0]) + 1 and update the FIFO 1 Fill Level F1FL (CANFD_RXF1S[6:0]).

Rx Buffer / FIFO Element Size Configuration Register (CANFD_RXESC)

Register	Offset	R/W	Description	Reset Value
CANFD_RXESC	CANFDn_BA+0x0BC	R	Rx Buffer / FIFO Element Size Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					RBDS		
7	6	5	4	3	2	1	0
Reserved	F1DS			Reserved	F0DS		

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:8]	**RBDS**	**Rx Buffer Data Field Size** 000 = 8 byte data field. 001 = 12 byte data field. 010 = 16 byte data field. 011 = 20 byte data field. 100 = 24 byte data field. 101 = 32 byte data field. 110 = 48 byte data field. 111 = 64 byte data field.
[7]	**Reserved**	Reserved.
[6:4]	**F1DS**	**Rx FIFO 1 Data Field Size** 000 = 8 byte data field. 001 = 12 byte data field. 010 = 16 byte data field. 011 = 20 byte data field. 100 = 24 byte data field. 101 = 32 byte data field. 110 = 48 byte data field. 111 = 64 byte data field.
[3]	**Reserved**	Reserved.
[2:0]	**F0DS**	**Rx FIFO 0 Data Field Size** 000 = 8 byte data field. 001 = 12 byte data field. 010 = 16 byte data field. 011 = 20 byte data field. 100 = 24 byte data field.

		101 = 32 byte data field. 110 = 48 byte data field. 111 = 64 byte data field. **Note:** In case the data field size of an accepted CAN frame exceeds the data field size configured for the matching Rx Buffer or Rx FIFO, only the number of bytes as configured by CANFD_RXESC are stored to the Rx Buffer resp. Rx FIFO element. The rest of the frame data field is ignored.

Tx Buffer Configuration Register (CANFD_TXBC)

Register	Offset	R/W	Description	Reset Value
CANFD_TXBC	CANFDn_BA+0x0C0	R	Tx Buffer Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	TFQM	TFQS					
23	22	21	20	19	18	17	16
Reserved		NDTB					
15	14	13	12	11	10	9	8
TBSA							
7	6	5	4	3	2	1	0
TBSA						Reserved	

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**TFQM**	**Tx FIFO/Queue Mode** 0= Tx FIFO operation. 1= Tx Queue operation.
[29:24]	**TFQS**	**Transmit FIFO/Queue Size** 0= No Tx FIFO/Queue. 1-32= Number of Tx Buffers used for Tx FIFO/Queue. >32= Values greater than 32 are interpreted as 32.
[23:22]	**Reserved**	Reserved.
[21:16]	**NDTB**	**Number of Dedicated Transmit Buffers** 0= No Dedicated Tx Buffers. 1-32= Number of Dedicated Tx Buffers. >32= Values greater than 32 are interpreted as 32.
[15:2]	**TBSA**	**Tx Buffers Start Address** Start address of Tx Buffers section in Message RAM (32-bit word address, refer to Figure 6.38-11). **Note:** The sum of TFQS and NDTB may be not greater than 32. There is no check for erroneous configurations. The Tx Buffers section in the Message RAM starts with the dedicated Tx Buffers
[1:0]	**Reserved**	Reserved.

Tx FIFO / Queue Status Register (CANFD_TXFQS)

Register	Offset	R/W	Description	Reset Value
CANFD_TXFQS	CANFDn_BA+0x0C4	R	Tx FIFO/Queue Status	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		TFQF	TFQPI				
15	14	13	12	11	10	9	8
Reserved			TFGI				
7	6	5	4	3	2	1	0
Reserved		TFFL					

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**TFQF**	**Tx FIFO/Queue Full** 0= Tx FIFO/Queue is not full. 1= Tx FIFO/Queue is full.
[20:16]	**TFQP**	**Tx FIFO/Queue Put Index** Tx FIFO/Queue write index pointer, range 0 to 31.
[15:13]	**Reserved**	Reserved.
[12:8]	**TFG**	**Tx FIFO Get Index** Tx FIFO read index pointer, range 0 to 31. Read as zero when Tx Queue operation is configured (TFQM (CANFD_TXBC[30]) = 1).
[7:6]	**Reserved**	Reserved.
[5:0]	**TFFL**	**Tx FIFO Free Level** Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as zero when Tx Queue operation is configured (TFQM (CANFD_TXBC[30]) = 1)**.** **Note:** In case of mixed configurations where dedicated Tx Buffers are combined with a Tx FIFO or a Tx Queue, the Put and Get Indices indicate the number of the Tx Buffer starting with the first dedicated Tx Buffers. Example: For a configuration of 12 dedicated Tx Buffers and a Tx FIFO of 20 Buffers a Put Index of 15 points to the fourth buffer of the Tx FIFO.

Tx Buffer Element Size Configuration Register (CANFD_TXESC)

Register	Offset	R/W	Description	Reset Value
CANFD_TXESC	CANFDn_BA+0x0C8	R	Tx Buffer Element Size Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**TBDS**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**TBDS**	**Tx Buffer Data Field Size** 000 = 8 byte data field. 001 = 12 byte data field. 010 = 16 byte data field. 011 = 20 byte data field. 100 = 24 byte data field. 101 = 32 byte data field. 110 = 48 byte data field. 111 = 64 byte data field. **Note:** In case the data length code DLC of a Tx Buffer element is configured to a value higher than the Tx Buffer data field size TBDS(CANFD_TXESC[2:0]), the bytes not defined by the Tx Buffer are transmitted as 0xCC (padding bytes).

Tx Buffer Request Pending Register (CANFD_TXBRP)

Register	Offset	R/W	Description	Reset Value
CANFD_TXBRP	CANFDn_BA+0x0CC	R	Tx Buffer Request Pending	0x0000_0000

31	30	29	28	27	26	25	24
TRPn							
23	22	21	20	19	18	17	16
TRPn							
15	14	13	12	11	10	9	8
TRPn							
7	6	5	4	3	2	1	0
TRPn							

Bits	Description	
[31:0]	TRPn	**Transmission Request Pending** **Each Tx Buffer has its own Transmission Request Pending bit The bits are set via register CANFD_TXBAR The bits are reset after a requested transmission has completed or has been cancelled via register CANFD_TXBCR** CANFD_TXBRP bits are set only for those Tx Buffers configured via CANFD_TXBC. After a CANFD_TXBRP bit has been set, a Tx scan (refer to 6.38.5.5, Tx Handling) is started to check for the pending Tx request with the highest priority (Tx Buffer with lowest Message ID). A cancellation request resets the corresponding transmission request pending bit of register CANFD_TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding CANFD_TXBRP bit has been reset. After a cancellation has been requested, a finished cancellation is signaled via CANFD_TXBCF • after successful transmission together with the corresponding CANFD_TXBTO bit • when the transmission has not yet been started at the point of cancellation • when the transmission has been aborted due to lost arbitration • when an error occurred during frame transmission In DAR mode all transmissions are automatically cancelled if they are not successful. The corresponding CANFD_TXBCF bit is set for all unsuccessful transmissions. 0 = No transmission request pending. 1 = Transmission request pending. **Note:** CANFD_TXBRP bits which are set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx Buffer, this Add Request is cancelled immediately, the corresponding CANFD_TXBRP bit is reset.

Tx Buffer Add Request Register (CANFD_TXBAR)

Register	Offset	R/W	Description	Reset Value
CANFD_TXBAR	CANFDn_BA+0x0D0	R/W	Tx Buffer Add Request	0x0000_0000

31	30	29	28	27	26	25	24
ARn							
23	22	21	20	19	18	17	16
ARn							
15	14	13	12	11	10	9	8
ARn							
7	6	5	4	3	2	1	0
ARn							

Bits	Description	
[31:0]	ARn	**Add Request** **Each Tx Buffer has its own Add Request bit Writing a 1 will set the corresponding Add Request bit; writing a 0 has no impact This enables the Host to set transmission requests for multiple Tx Buffers with one write to CANFD_TXBAR CANFD_TXBAR bits are set only for those Tx Buffers configured via CANFD_TXBC When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed** 0 = No transmission request added. 1 = Transmission requested added. **Note:** If an add request is applied for a Tx Buffer with pending transmission request (corresponding CANFD_TXBRP bit already set), this add request is ignored.

Tx Buffer Cancellation Request Register (CANFD_TXBCR)

Register	Offset	R/W	Description	Reset Value
CANFD_TXBCR	CANFDn_BA+0x0D4	R/W	Tx Buffer Cancellation Request	0x0000_0000

31	30	29	28	27	26	25	24
CRn							
23	22	21	20	19	18	17	16
CRn							
15	14	13	12	11	10	9	8
CRn							
7	6	5	4	3	2	1	0
CRn							

Bits	Description	
[31:0]	CRn	**Cancellation Request** Each Tx Buffer has its own Cancellation Request bit. Writing a 1 will set the corresponding Cancellation Request bit; writing a 0 has no impact. This enables the Host to set cancellation requests for multiple Tx Buffers with one write to CANFD_TXBCR. CANFD_TXBCR bits are set only for those Tx Buffers configured via CANFD_TXBC. The bits remain set until the corresponding bit of CANFD_TXBRP is reset. 0 = No cancellation pending. 1 = Cancellation pending.

Tx Buffer Transmission Occurred Register (CANFD_TXBTO)

Register	Offset	R/W	Description	Reset Value
CANFD_TXBTO	CANFDn_BA+0x0D8	R	Tx Buffer Transmission Occurred	0x0000_0000

31	30	29	28	27	26	25	24
TOn							
23	22	21	20	19	18	17	16
TOn							
15	14	13	12	11	10	9	8
TOn							
7	6	5	4	3	2	1	0
TOn							

Bits	Description	
[31:0]	TOn	**Transmission Occurred** Each Tx Buffer has its own Transmission Occurred bit. The bits are set when the corresponding CANFD_TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register CANFD_TXBAR. 0 = No transmission occurred. 1 = Transmission occurred.

Tx Buffer Cancellation Finished Register (CANFD_TXBCF)

Register	Offset	R/W	Description	Reset Value
CANFD_TXBCF	CANFDn_BA+0x0DC	R	Tx Buffer Cancellation Finished	0x0000_0000

31	30	29	28	27	26	25	24
CFn							
23	22	21	20	19	18	17	16
CFn							
15	14	13	12	11	10	9	8
CFn							
7	6	5	4	3	2	1	0
CFn							

Bits	Description	
[31:0]	CFn	**Cancellation Finished** Each Tx Buffer has its own Cancellation Finished bit. The bits are set when the corresponding CANFD_TXBRP bit is cleared after a cancellation was requested via CANFD_TXBCR. In case the corresponding CANFD_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register CANFD_TXBAR. 0 = No transmit buffer cancellation. 1 = Transmit buffer cancellation finished.

Tx Buffer Transmission Interrupt Enable Register (CANFD_TXBTIE)

Register	Offset	R/W	Description	Reset Value
CANFD_TXBTIE	CANFDn_BA+0x0E0	R/W	Tx Buffer Transmission Interrupt Enable	0x0000_0000

31	30	29	28	27	26	25	24
TIEn							
23	22	21	20	19	18	17	16
TIEn							
15	14	13	12	11	10	9	8
TIEn							
7	6	5	4	3	2	1	0
TIEn							

Bits	Description	
[31:0]	TIEn	**Transmission Interrupt Enable** Each Tx Buffer has its own Transmission Interrupt enable bit. 0 = Transmission interrupt disabled. 1 = Transmission interrupt enable.

Tx Buffer Cancellation Finished Interrupt Enable Register (CANFD_TXBCIE)

Register	Offset	R/W	Description	Reset Value
CANFD_TXBCIE	CANFDn_BA+0x0E4	R/W	Tx Buffer Cancellation Finished Interrupt Enable	0x0000_0000

31	30	29	28	27	26	25	24
CFIEn							
23	22	21	20	19	18	17	16
CFIEn							
15	14	13	12	11	10	9	8
CFIEn							
7	6	5	4	3	2	1	0
CFIEn							

Bits	Description	
[31:0]	CFIEn	**Cancellation Finished Interrupt Enable** Each Tx Buffer has its own Cancellation Finished Interrupt Enable bit. 0 = Cancellation finished interrupt disabled. 1 = Cancellation finished interrupt enabled.

Tx Event FIFO Configuration Register (CANFD_TXEFC)

Register	Offset	R/W	Description	Reset Value
CANFD_TXEFC	CANFDn_BA+0x0F0	R	Tx Event FIFO Configuration (P*)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		EFWM					
23	22	21	20	19	18	17	16
Reserved		EFS					
15	14	13	12	11	10	9	8
EFSA							
7	6	5	4	3	2	1	0
EFSA						Reserved	

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**EFWM**	**Event FIFO Watermark** 0 = Watermark interrupt disabled. 1-32= Level for Tx Event FIFO watermark interrupt (TEFW (CANFD_IR[13])). >32= Watermark interrupt disabled.
[23:22]	**Reserved**	Reserved.
[21:16]	**EFS**	**Event FIFO Size** 0= Tx Event FIFO disabled. 1-32= Number of Tx Event FIFO elements. >32= Values greater than 32 are interpreted as 32. The Tx Event FIFO elements are indexed from 0 to EFS - 1
[15:2]	**EFSA**	**Event FIFO Start Address** Start address of Tx Event FIFO in Message RAM (32-bit word address, refer to Figure 6.38-11).
[1:0]	**Reserved**	Reserved.

Tx Event FIFO Status Register (CANFD_TXEFS)

Register	Offset	R/W	Description	Reset Value
CANFD_TXEFS	CANFDn_BA+0x0F4	R	Tx Event FIFO Status	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						TEFL	EFF
23	22	21	20	19	18	17	16
Reserved			EFPI				
15	14	13	12	11	10	9	8
Reserved			EFGI				
7	6	5	4	3	2	1	0
Reserved		EFFL					

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**TEFL**	**Tx Event FIFO Element Lost** This bit is a copy of interrupt flag TEFL (CANFD_IR[15]). When TEFL is reset, this bit is also reset. 0= No Tx Event FIFO element lost. 1= Tx Event FIFO element lost, also set after write attempt to Tx Event FIFO of size zero.
[24]	**EFF**	**Event FIFO Full** 0= Tx Event FIFO is not full. 1= Tx Event FIFO is full.
[23:21]	**Reserved**	Reserved.
[20:16]	**EFPI**	**Event FIFO Put Index** Tx Event FIFO write index pointer, range 0 to 31
[15:13]	**Reserved**	Reserved.
[12:8]	**EFGI**	**Event FIFO Get Index** Tx Event FIFO read index pointer, range 0 to 31
[7:6]	**Reserved**	Reserved.
[5:0]	**EFFL**	**Event FIFO Fill Level** Number of elements stored in Tx Event FIFO, range 0 to 32

Tx Event FIFO Acknowledge Register (CANFD_TXEFA)

Register	Offset	R/W	Description	Reset Value
CANFD_TXEFA	CANFDn_BA+0x0F8	R/W	Tx Event FIFO Acknowledge	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**EFAI**				

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**EFAI**	**Event FIFO Acknowledge Index** After the Host has read an element or a sequence of elements from the Tx Event FIFO it has to write the index of the last element read from Tx Event FIFO to EFAI. This will set the Tx Event FIFO Get Index EFGI (CANFD_TXEFS[12:8]) to EFAI + 1 and update the Event FIFO Fill Level EFFL (CANFD_TXEFS[5:0])

6.39 CRC Controller (CRC)

6.39.1 Overview

The Cyclic Redundancy Check (CRC) generator can perform CRC calculation with 8-bits, 16-bits and 32-bits configurable polynomials.

6.39.2 Features

- Supports 8-bits, 16-bits and 32-bits configurable polynomials
- Programmable seed value
- Supports programmable order reverse setting for input data and CRC checksum
- Supports programmable 1's complement setting for input data and CRC checksum
- Supports 8/16/32-bit of data width
  - 8-bit write mode: 1-AHB clock cycle operation
  - 16-bit write mode: 2-AHB clock cycle operation
  - 32-bit write mode: 4-AHB clock cycle operation
- Supports using PDMA to write data to perform CRC operation

6.39.3 Block Diagram

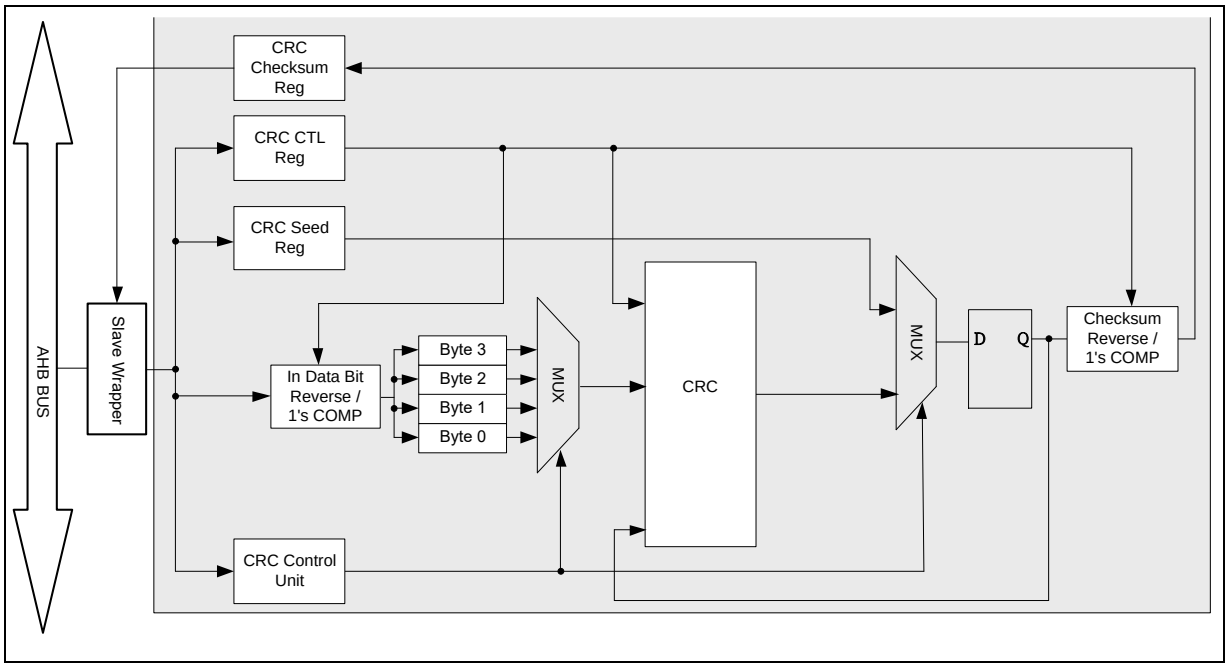


Figure 6.39-1 CRC Generator Block Diagram

6.39.4 Basic Configuration

- Clock Source Configuration
  - Enable CRC peripheral clock in CRCCKEN (CLK_AHBCLK0[7]).
- Reset Configuration
  - Reset CRC controller in CRCRST (SYS_IPRST0[7]).

6.39.5 Functional Description

CRC generator can perform CRC calculation with 8-bits, 6-bits and 32-bits configurable polynomial settings. User can choose the length of CRC polynomial by setting CRCMODE[1:0] (CRC_CTL[31:30] CRC Polynomial Mode) and program the value of CRC polynomial by setting POLYNOMIAL[31:0] (CRC_POLYNOMIAL[31:0] CRC Polynomial value).

For example, the user must write 0x07 to register CRC_POLYNOMIAL when the polynomial ($X^8 + X^2 + X + 1$) is used for CRC 8-bits calculation. If the polynomial ($X^{16} + X^{15} + X^2 + 1$) is used for CRC 16-bits calculation, the value of register CRC_POLYNOMIAL must be 0x8005. User must write 0x04C11DB7 to register CRC_POLYNOMIAL when the polynomial ($X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$) is used for CRC 32-bits calculation.

The following is a program sequence example.

1. Enable CRC generator by setting CRCEN (CRC_CTL[0] CRC Channel Enable Bit).
2. Configure the CRC polynomial mode and value.
    1) Configure the polynomial mode by setting CRCMODE (CRC_CTL[31:30] CRC Polynomial Mode).
    2) Configure the polynomial value by setting POLYNOMIAL (CRC_POLYNOMIAL [31:0] CRC Polynomial Value).
3. Initial setting for CRC calculation.
    1) Configure 1's complement for CRC checksum by setting CHKSFMT (CRC_CTL[27] Checksum 1's Complement).
    2) Configure bit order reverse for CRC checksum by setting CHKSREV (CRC_CTL[25] Checksum Bit Order Reverse). The functional block is also shown in Figure 6.39-2.
    3) Configure 1's complement for CRC write data by setting DATFMT (CRC_CTL[26] Write Data 1's Complement).
    4) Configure bit order reverse for CRC write data per byte by setting DATREV (CRC_CTL[24] Write Data Bit Order Reverse). The functional block is also shown in Figure 6.39-3.
4. Perform CHKSINIT (CRC_CTL[1] Checksum Initialization) to load the initial checksum value from CRC_SEED register value.
5. Write data to CRC_DAT register to calculate CRC checksum.
6. Get the CRC checksum result by reading CRC_CHECKSUM register.

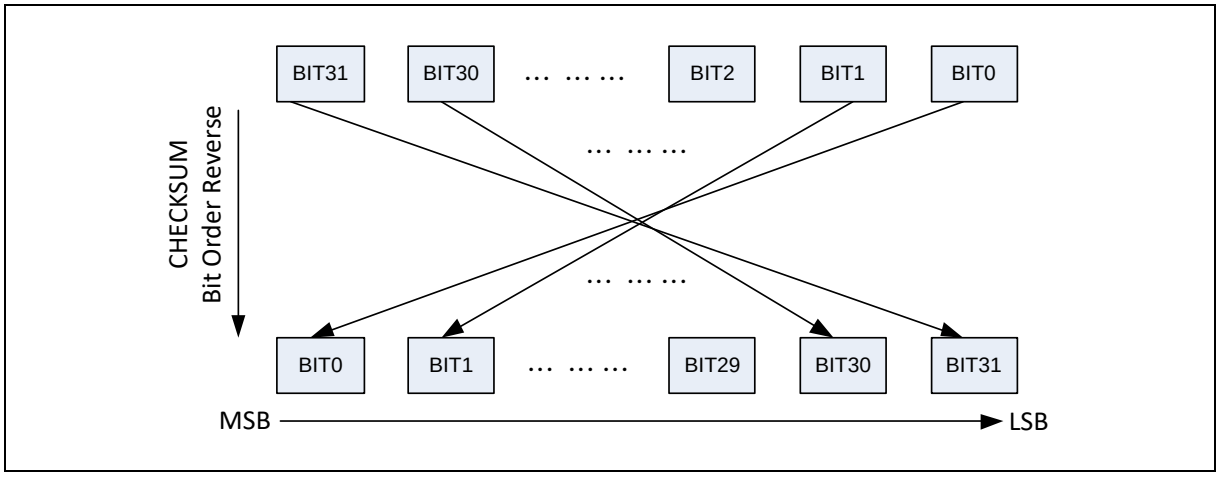


Figure 6.39-2 CHECKSUM Bit Order Reverse Functional Block

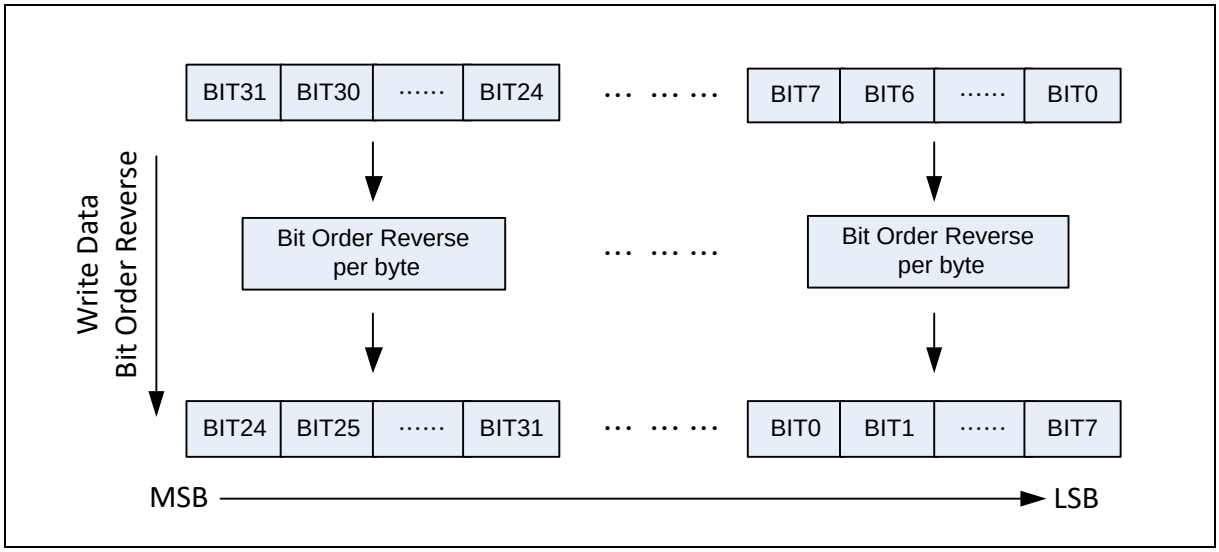


Figure 6.39-3 Write Data Bit Order Reverse Functional Block

6.39.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
CRC Base Address: **CRC_BA = 0x4003_1000**				
CRC_CTL	CRC_BA+0x00	R/W	CRC Control Register	0x6000_0000
CRC_DAT	CRC_BA+0x04	R/W	CRC Write Data Register	0x0000_0000
CRC_SEED	CRC_BA+0x08	R/W	CRC Seed Register	0xFFFF_FFFF
CRC_CHECKSUM	CRC_BA+0x0C	R	CRC Checksum Register	0xFFFF_FFFF
CRC_POLYNOMIAL	CRC_BA+0x10	R/W	CRC Polynomial Register	0x0000_0000

6.39.7 Register Description

CRC Control Register (CRC_CTL)

Register	Offset	R/W	Description	Reset Value
CRC_CTL	CRC_BA+0x00	R/W	CRC Control Register	0x6000_0000

31	30	29	28	27	26	25	24
CRCMODE		DATLEN		CHKSFMT	DATFMT	CHKSREV	DATREV
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						CHKSINIT	CRCEN

Bits	Description	
[31:30]	**CRCMODE**	**CRC Polynomial Mode** This field indicates the CRC operation polynomial mode. 00 = CRC-16 Polynomial mode. 01 = CRC-8 Polynomial mode. 10 = CRC-16 Polynomial mode. 11 = CRC-32 Polynomial mode. **Note:** User must program the polynomial value in CRC_POLYNOMIAL register to specify the polynomial used for CRC calculation.
[29:28]	**DATLEN**	**CPU Write Data Length** This field indicates the write data length. 00 = Data length is 8-bit mode. 01 = Data length is 16-bit mode. 10 = Data length is 32-bit mode. **Note:** When the write data length is 8-bit mode, the valid data in CRC_DAT register is only DATA[7:0] bits; if the write data length is 16-bit mode, the valid data in CRC_DAT register is only DATA[15:0].
[27]	**CHKSFMT**	**Checksum 1's Complement** This bit is used to enable the 1's complement function for checksum result in CRC_CHECKSUM register. 0 = 1's complement for CRC checksum Disabled. 1 = 1's complement for CRC checksum Enabled.
[26]	**DATFMT**	**Write Data 1's Complement** This bit is used to enable the 1's complement function for write data value in CRC_DAT register. 0 = 1's complement for CRC writes data in Disabled. 1 = 1's complement for CRC writes data in Enabled.

[25]	**CHKSREV**	**Checksum Bit Order Reverse** This bit is used to enable the bit order reverse function for checksum result in CRC_CHECKSUM register. 0 = Bit order reverse for CRC checksum Disabled. 1 = Bit order reverse for CRC checksum Enabled. **Note:** If the checksum result is 0xDD7B0F2E, the bit order reverse for CRC checksum is 0x74F0DEBB.
[24]	**DATREV**	**Write Data Bit Order Reverse** This bit is used to enable the bit order reverse function per byte for write data value in CRC_DAT register. 0 = Bit order reversed for CRC write data in Disabled. 1 = Bit order reversed for CRC write data in Enabled (per byte). **Note:** If the write data is 0xAABBCCDD, the bit order reverse for CRC write data in is 0x55DD33BB.
[23:2]	**Reserved**	Reserved.
[1]	**CHKSINIT**	**Checksum Initialization (Write Only)** 0 = No effect. 1 = Initial checksum value by auto reload CRC_SEED register value to CRC_CHECKSUM register value. **Note:** This bit will be cleared automatically.
[0]	**CRCEN**	**CRC Channel Enable Bit** 0 = No effect. 1 = CRC operation Enabled.

CRC Write Data Register (CRC_DAT)

Register	Offset	R/W	Description	Reset Value
CRC_DAT	CRC_BA+0x04	R/W	CRC Write Data Register	0x0000_0000

31	30	29	28	27	26	25	24
DATA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DATA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DATA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA							

Bits	Description	
[31:0]	**DATA**	**CRC Write Data Bits** User can write data directly by CPU mode or use PDMA function to write data to this field to perform CRC operation. **Note:** When the write data length is 8-bit mode, the valid data in CRC_DAT register is only DATA[7:0] bits; if the write data length is 16-bit mode, the valid data in CRC_DAT register is only DATA[15:0].

CRC Seed Register (CRC_SEED)

Register	Offset	R/W	Description	Reset Value
CRC_SEED	CRC_BA+0x08	R/W	CRC Seed Register	0xFFFF_FFFF

31	30	29	28	27	26	25	24
SEED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SEED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SEED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SEED							

Bits	Description	
[31:0]	**SEED**	**CRC Seed Value** This field indicates the CRC seed value. **Note:** This field will be reloaded as checksum initial value (CRC_CHECKSUM register) after performing CHKSINIT (CRC_CTL[1]).

CRC Checksum Register (CRC_CHECKSUM)

Register	Offset	R/W	Description	Reset Value
CRC_CHECKSUM	CRC_BA+0x0C	R	CRC Checksum Register	0xFFFF_FFFF

31	30	29	28	27	26	25	24
CHECKSUM							
23	22	21	20	19	18	17	16
CHECKSUM							
15	14	13	12	11	10	9	8
CHECKSUM							
7	6	5	4	3	2	1	0
CHECKSUM							

Bits	Description	
[31:0]	CHECKSUM	**CRC Checksum Results** This field indicates the CRC checksum result.

CRC Polynomial Register (CRC_POLYNOMIAL)

Register	Offset	R/W	Description	Reset Value
CRC_POLYNOMIAL	CRC_BA+0x10	R/W	CRC Polynomial Register	0x0000_0000

31	30	29	28	27	26	25	24
POLYNOMIAL							
23	22	21	20	19	18	17	16
POLYNOMIAL							
15	14	13	12	11	10	9	8
POLYNOMIAL							
7	6	5	4	3	2	1	0
POLYNOMIAL							

Bits	Description	
[31:0]	POLYNOMIAL	**CRC Polynomial Value** This field indicates the value of CRC polynomial.

6.40 Cryptographic Accelerator (CRYPTO)

6.40.1 Overview

The Crypto (Cryptographic Accelerator) includes a secure pseudo random number generator (PRNG) core and supports AES, SHA/HMAC, RSA, and ECC algorithms.

The PRNG core supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation. (283~571 bits are only generated for key store).

The AES accelerator is an implementation fully compliant with the AES (Advance Encryption Standard) encryption and decryption algorithm. The AES accelerator supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2, CBC-CS3, CCM and GCM mode.

The SHA accelerator is an implementation fully compliant with the SHA-160, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/t, SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 and SHAKE256. The SHA accelerator supports HMAC (Keyed-Hash Message Authentication Code) algorithms including HMAC-SHA-160, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512, HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, and HMAC-SHA3-512.

The ECC accelerator is an implementation fully compliant with elliptic curve cryptography by using polynomial basis in binary field and prime filed.

The RSA accelerator is an implementation fully compliant with RSA cryptography, CRT decryption algorithm and side-channel attack countermeasures algorithm.

The Crypto can get key from key store and/or put the key to key store determined by the function of each accelerator.

6.40.2 Features

- PRNG
  - Supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation (283~571 bits are only generated for key store).
  - Can take the true random number seed from TRNG
- AES
  - Supports FIPS NIST 197
  - Supports SP800-38A and addendum
  - Supports 128, 192, and 256 bits key
  - Supports both encryption and decryption
  - Supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2 and CBC-CS3 modes
  - Supports CCM mode, GCM mode and GMAC function
  - Supports key expander for key expansion in FIPS NIST 197
  - Supports one technique to improve side-channel attack protection ability
- SHA
  - Supports FIPS NIST 180, 180-2, 180-4
  - Supports SHA-160, SHA-224, SHA-256, SHA-384, SHA-512 and SHA-512/t
  - Supports SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 and SHAKE256
- HMAC
  - Supports FIPS NIST 180, 180-2, 180-4
  - Supports HMAC-SHA-160, HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and

HMAC-SHA-512

- Supports HMAC-SHA3-224, HMAC-SHA3-256, HMAC-SHA3-384, and HMAC-SHA3-512

- ECC
  - Supports both prime field GF(p) and binary filed GF(2^m)
  - Supports NIST P-192, P-224, P-256, P-384, and P-521
  - Supports NIST B-163, B-233, B-283, B-409, and B-571
  - Supports NIST K-163, K-233, K-283, K-409, and K-571
  - Supports Curve25519
  - Supports Public Key Cryptographic Algorithm SM2 Based on Elliptic Curves
  - Supports point multiplication, addition and doubling operations in GF(p) and GF(2^m)
  - Supports modulus division, multiplication, addition and subtraction operations in GF(p)
  - Supports four techniques to improve side-channel attack protection ability
- RSA
  - Supports both encryption and decryption with 1024, 2048, 3072 and 4096 bits
  - Supports CRT decryption with 2048, 3072 and 4096 bits
  - Supports three techniques to improve side-channel attack protection ability

6.40.3 Block Diagram

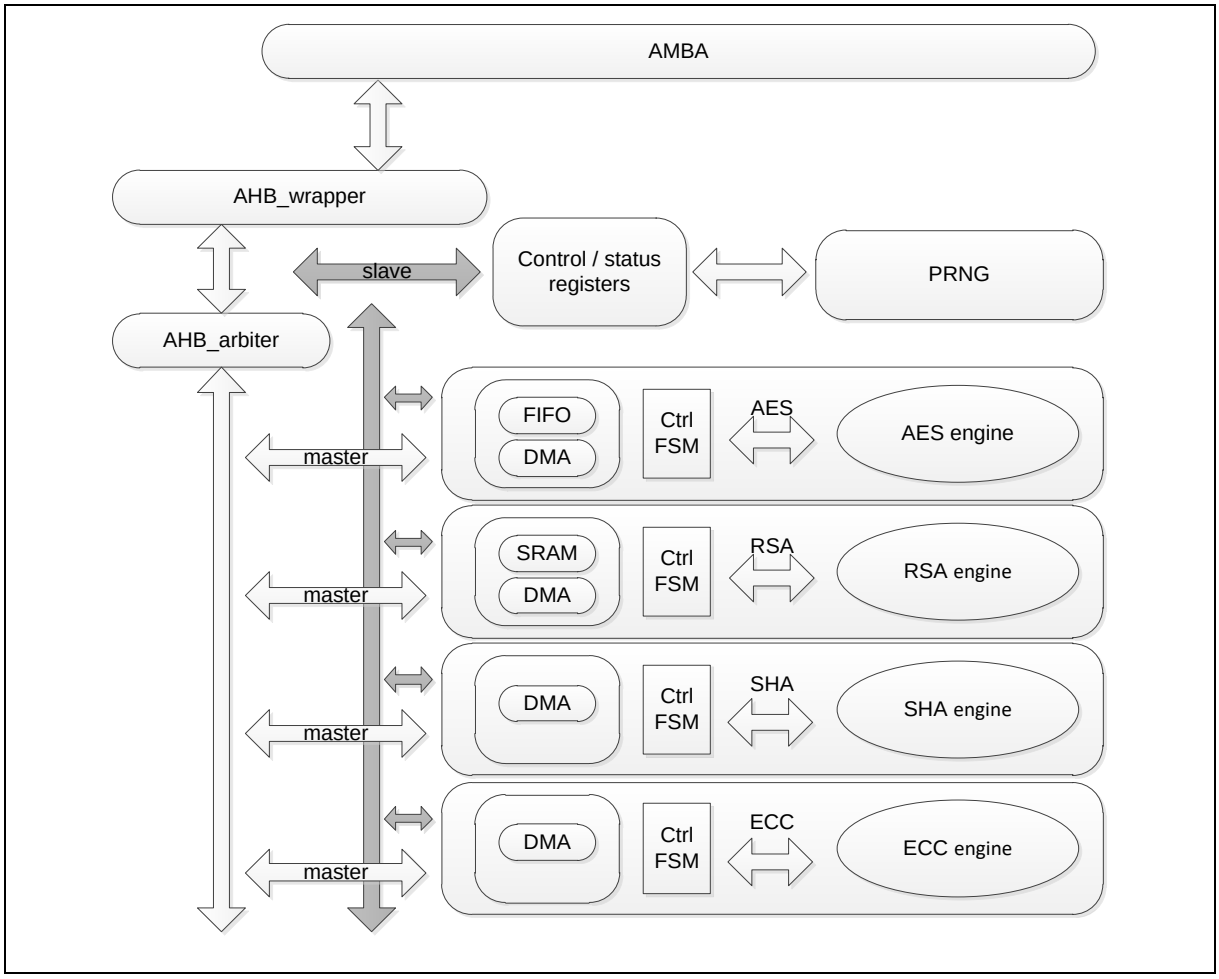


Figure 6.40-1 Cryptographic Accelerator Block Diagram

6.40.4 Basic Configuration

- Clock Source Configuration
  - Enable CRYPTO peripheral clock in CRPTCKEN(CLK_AHBCLK[12])
- Reset Configuration
  - Reset CRYPTO controller in CRPTRST(SYS_IPRST0[12])

6.40.5 Functional Description

The cryptographic accelerator includes a secure pseudo random number generator (PRNG) core and supports AES, SHA/HMAC, RSA, and ECC algorithms. The accelerator can be used in different data security applications, such as secure communications that need cryptographic protection and integrity.

1. The PRNG core supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation configured by KEYSZ. (283~571 bits are only generated for key store).
2. The AES accelerator is a fully compliant implementation of the AES (Advance Encryption Standard) encryption and decryption algorithm. The AES accelerator supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2, CBC-CS3, CCM and GCM mode. The AES accelerator provides the DMA function to reduce the CPU intervention, and supports three burst lengths, sixteen-words, eight-words, and four-words.

3. The SHA/HMAC accelerator is a fully compliant implementation of the SHA1 (i.e., SHA-160), SHA2 (i.e., SHA-224, SHA-256, SHA-384, SHA-512), SHA3 (i.e., SHA3-224, SHA3-256, SHA3-384, SHA3-512, SHAKE128 and SHAKE256) and corresponding HMAC algorithm (except for SHAKE128 and SHAKE256). The SHA/HMAC accelerator also supports the DMA function to reduce the CPU intervention. It supports three burst lengths, sixteen-words, eight-words, and four-words.
4. The ECC accelerator is a fully compliant implementation of the prime field GF(p) and binary field GF(2^m) algorithm in polynomial basis. The prime field GF(p) supports NIST P-192, P-224, P-256, P-384, P-521 and Curve25519. The binary field GF(2^m) supports NIST B-163, B-233, B-283, B-409, B-571 and NIST K-163, K-233, K-283, K-409 and K-571. Besides, Public Key Cryptographic Algorithm SM2 is also implemented in this accelerator.
5. The RSA accelerator is a fully compliant implementation of RSA-1024, RSA-2048, RSA-3072 and RSA-4096. The CRT mode supports RSA-2048, RSA-3072 and RSA-4096.

Software can control the data flow by enabling the CRYPTO_INTEN, and monitor the accelerator status by checking the CRYPTO_INTSTS status register. When any engine happened operation error or buffer error, the corresponding error flag will be set to 1 and inform to CPU if error interrupt enable bit is set to 1. If users want to detail error condition, software can check status flag register of each engine.

Engine	Error Interrupt Enable Bit	Error Interrupt Flag	Error Conditions
PRNG	PRNGEIEN (CRYPTO_INTEN[17])	PRNGEIF (CRYPTO_INTSTS[17])	KSERR/KCTLERR
AES	AESEIEN (CRYPTO_INTEN[1])	AESEIF (CRYPTO_INTSTS[1])	KSERR/INBUFERR/OUTBUFERR/BUSERR
SHA/HMAC	HMACEIEN (CRYPTO_INTEN[25])	HMACEIF (CRYPTO_INTSTS[25])	KSERR/DMAERR/BUSERR
ECC	ECCEIEN (CRYPTO_INTEN[23])	ECCEIF (CRYPTO_INTSTS[23])	KSERR/BUSERR
RSA	RSAEIEN (CRYPTO_INTEN[31])	RSAEIF (CRYPTO_INTSTS[31])	KSERR/CTLERR/BUSERR

Table 6.40-1 Each Engine Error Conditions and Error Flag

The cryptographic accelerator supports the following features to enhance the performance.

DMA Mode

Once DMA source address register, destination address register, and byte count register are configured by CPU, moving data from and to accelerator is done by DMA logic totally. This mode can off-load the loading from the CPU. The cryptographic accelerator embeds one hardware DMA channel for AES engine, one hardware DMA channel for SHA/HMAC engine, one hardware DMA channel for RSA engine, and one hardware DMA channel for ECC engine. The RSA engine only supports DMA mode.

Engine	DMA Enable Bit
AES	DMAEN (CRYPTO_AES_CTL[7])
SHA/HMAC	DMAEN (CRYPTO_HMAC_CTL[7])
ECC	DMAEN (CRYPTO_ECC_CTL[7])

Table 6.40-2 DMA Enable Bit Table

DMA Cascade Mode

In the case that the data SRAM resource is tight, or another peripheral is scheduled to switch, the data source or sink needs an update, while the setting for the accelerator operation is planned to be kept. In this mode, software can update DMA source address register, destination address register, and byte count register during a cascade operation, without finishing the accelerator operation. Because AES-CCM, AES-GCM, SHA3 and SHAKE operations require storing more information to continue the

operation, AES and SHA/HMAC provide FBOUT and FBIN bits in their control register to trigger DMA engine to automatically output and input all required information according to their DMA feedback address register (CRYPTO_AES_FBADDR or CRYPTO_HMAC_FBADDR). Please note that the last two blocks of AES-CBC-CS1/2/3 must be in the last cascade operation.

Engine	DMA Cascade Bit
AES	DMACSCAD (CRYPTO_AES_CTL[6])
SHA/HMAC	DMACSCAD (CRYPTO_HMAC_CTL[6])

Table 6.40-3 DMA Cascade Bit Table

Non-DMA Mode

In the case that the input data is small in size, DMA mode is not preferred. This mode can reduce the processing time for the accelerator, since no DMA related register needs a configuration, and no latency in DMA logic is introduced. Input data was feeding to cryptographic engine via writing to data input register. The RSA engine doses not support Non-DMA mode.

Channel Expansion Mode

In this mode, several virtual channels are feasible in AES mode. The intermediate data (or called feedback information) from feedback registers (CRYPTO_AES_FDBCKx) should be stored temporarily in data SRAM and switched to another configuration setting of accelerator operation that includes operational mode, encryption/decryption, key, key size, IV, and other parameters. Once switching back, the intermediate data from feedback registers should be written to initial vectors (CRYPTO_AES_IVx) for the accelerator to continue the operation with the original configuration setting. Finally, user must configure AES control register (CRYPTO_AES_CTL) to complete accelerator operation. Note that, in ECB mode, there is no need to move the intermediate data from feedback registers to IV. Because the feedback information of CCM and GCM mode are 18 words, above method are not suitable for CCM and GCM mode. To switch channels and continue the operation in CCM and GCM, AES provides FBOUT and FBIN bits in CRYPTO_AES_CTL to trigger AES DMA engine to automatically output and input all feedback information according to DMA feedback address register (CRYPTO_AES_FBADDR).

6.40.5.1 PRNG (Pseudo Random Number Generator)

The PRNG block diagram is depicted in Figure 6.41-2. The core supports 128, 163, 192, 224, 233, 255, 256, 283, 384, 409, 512, 521 and 571 bits random number generation configured by KEYSZ(CRYPTO_PRNG_CTL[5:2]). (283~571 bits are only generated for key store).

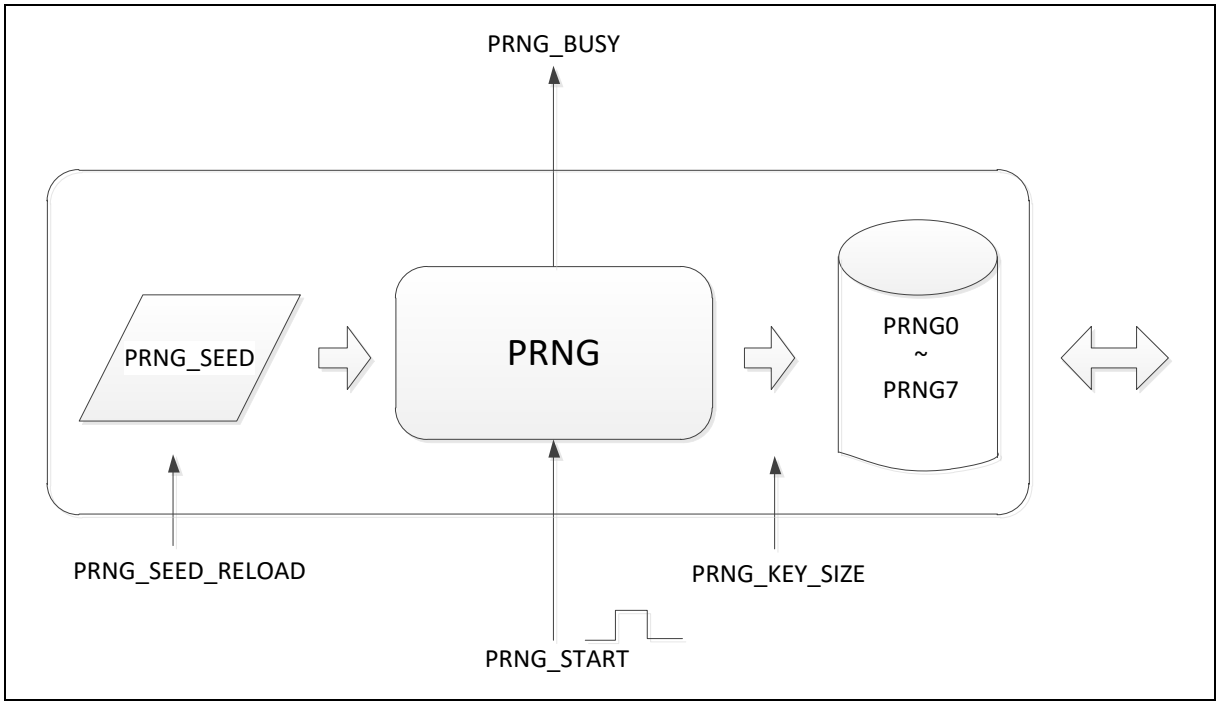


Figure 6.40-2 PRNG Function Diagram

Programming steps to get the **pseudo** random number are depicted below.

1. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
2. Write a random seed to CRYPTO_PRNG_SEED. Note that CRYPTO_PRNG_SEED should be initialized since it is not initialized as the chip powers up.
3. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
4. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then software can read the output random numbers from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7.

Programming steps to get the **pseudo** random number to **key store** are depicted below.

1. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
2. Write a random seed to CRYPTO_PRNG_SEED. Note that CRYPTO_PRNG_SEED should be initialized since it is not initialized as the chip is powered up.
3. Configure PRNG key control register CRYPTO_PRNG_KSCTL for key owner(CRYPTO_PRNG_KSCTL[26:24]), write key store destination(CRYPTO_PRNG_KSCTL[23:22]), write key destination(CRYPTO_PRNG_KSCTL[21]), ECDSA selection bit (CRYPTO_PRNG_KSCTL[20]), ECDH selection bit (CRYPTO_PRNG_KSCTL[19]) and trusted key(CRYPTO_PRNG_KSCTL[16]) in key store.
4. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
5. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then

software can read key NUMBER(CRYPTO_PRNG_KSSTS[4:0]).

Programming steps to get the random number with the **true** random number seed from TRNG are depicted below.

1. Select TRNG peripheral clock frequency in CLKPSC (TRNG_CTL[5:2]).
2. Enable ACT (TRNG_ACT[7]) bit.
3. Wait READY (TRNG_CTL[7]) bit to be set to 1.
4. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
5. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
6. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then software can read the output random numbers from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7.

Programming steps to get the random number with the **true** random number seed from TRNG to **key store** are depicted below.

6. Select TRNG peripheral clock frequency in CLKPSC (TRNG_CTL[5:2]).
7. Enable ACT (TRNG_ACT[7]) bit.
8. Wait READY (TRNG_CTL[7]) bit to be set to 1.
9. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
10. Configure PRNG key control register CRYPTO_PRNG_KSCTL for key owner(CRYPTO_PRNG_KSCTL[26:24]), write key store destination(CRYPTO_PRNG_KSCTL[23:22]), write key destination(CRYPTO_PRNG_KSCTL[21]), ECDSA selection bit (CRYPTO_PRNG_KSCTL[20]), ECDH selection bit (CRYPTO_PRNG_KSCTL[19]) and trusted key(CRYPTO_PRNG_KSCTL[16]) in key store.
11. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
12. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then software can read key NUMBER(CRYPTO_PRNG_KSSTS[4:0]).

Some Notices of PRNG:

1. If users want to generate key for key store, must to execute key store initialization (please refer to the Key Store (KS) section).
2. If users don't generate key for key store (CRYPTO_PRNG_KSCTL[21] is '0'), PRNG cannot generate more than 256 bits key (CRYPTO_PRNG_CTL[5:2]).
3. If users want to write key for ECDSA or ECDH to key store, PRNG seed must from TRNG and key is must written to the SRAM of key store (WSDST, CRYPTO_PRNG_KSCTL[23:22] must set to '00'). Otherwise, KCTLERR(CRYPTO_PRNG_KSSTS[16]) will become '1'.
4. If users want to write key for ECDSA or ECDH to key store, key must in the interval [1, n-1] (the parameter n is from ECC). The value of n cannot be 0 or 1, otherwise, PRNG will always keep busy.

6.40.5.2 AES (Advanced Encryption Standard)

The AES accelerator supports AES algorithm from U.S NIST. This accelerator provides 10 block cipher modes includes ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2, CBC-CS3, CCM and GCM mode. The detail of above block cipher modes are described in the following:

Electronic Codebook Mode

The Electronic Codebook (ECB) mode is a confidentiality mode that features the assignment of a fixed ciphertext block to each plaintext block, for a given key. It's analogous to the assignment of code words in a codebook.

In ECB encryption, each block of the plaintext is applied to the forward cipher function $CIPH_k$ directly and independently. The resulting sequence of output blocks is the ciphertext. In ECB decryption, each block of the ciphertext is applied to the inverse cipher function $CIPH^{-1}_k$ directly and independently. The resulting sequence of output blocks is the plaintext.

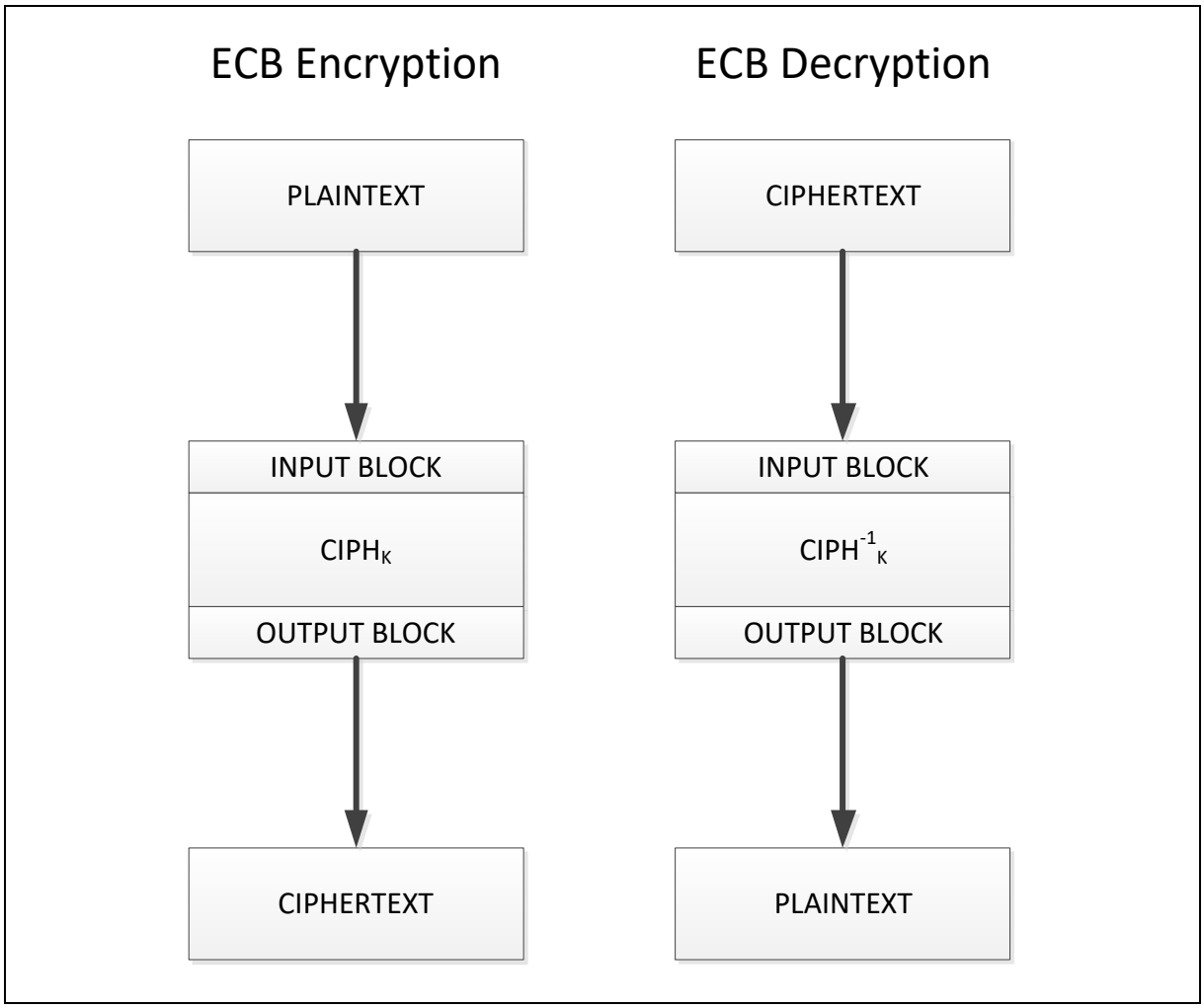


Figure 6.40-3 Electronic Codebook Mode

In ECB mode, any given plaintext block always gets encrypted to the same ciphertext block under a given key. If this property is undesirable in a particular application, the ECB mode should not be used.

Cipher Block Chaining Mode

The Cipher Block Chaining (CBC) mode is a confidentiality mode whose encryption process features

the combining chaining of the plaintext blocks with the previous ciphertext blocks. The CBC mode requires an initialization vector (IV) to combine with the first plaintext block. The IV does not need to be secret, but it must be unpredictable.

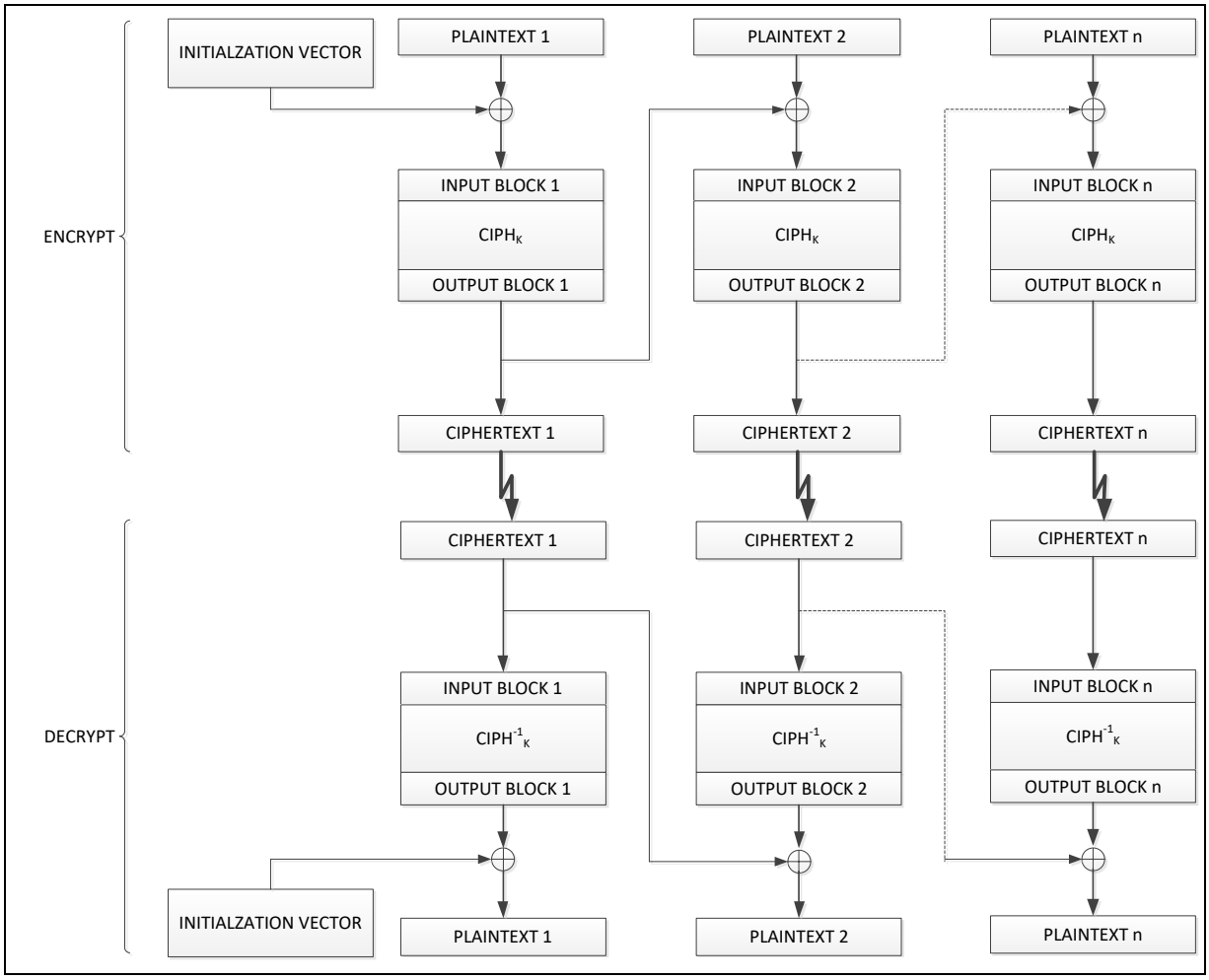


Figure 6.40-4 Cipher Block Chaining Mode

Cipher Feedback Mode (CFB)

The Cipher Feedback (CFB) mode is a confidentiality mode that features the feedback of successive ciphertext segments into the input blocks of the forward cipher to generate output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The CFB mode requires an IV as the initial input block. The IV need not be secret, but it must be unpredictable. The AES only supports 128-bit segment length CFB mode.

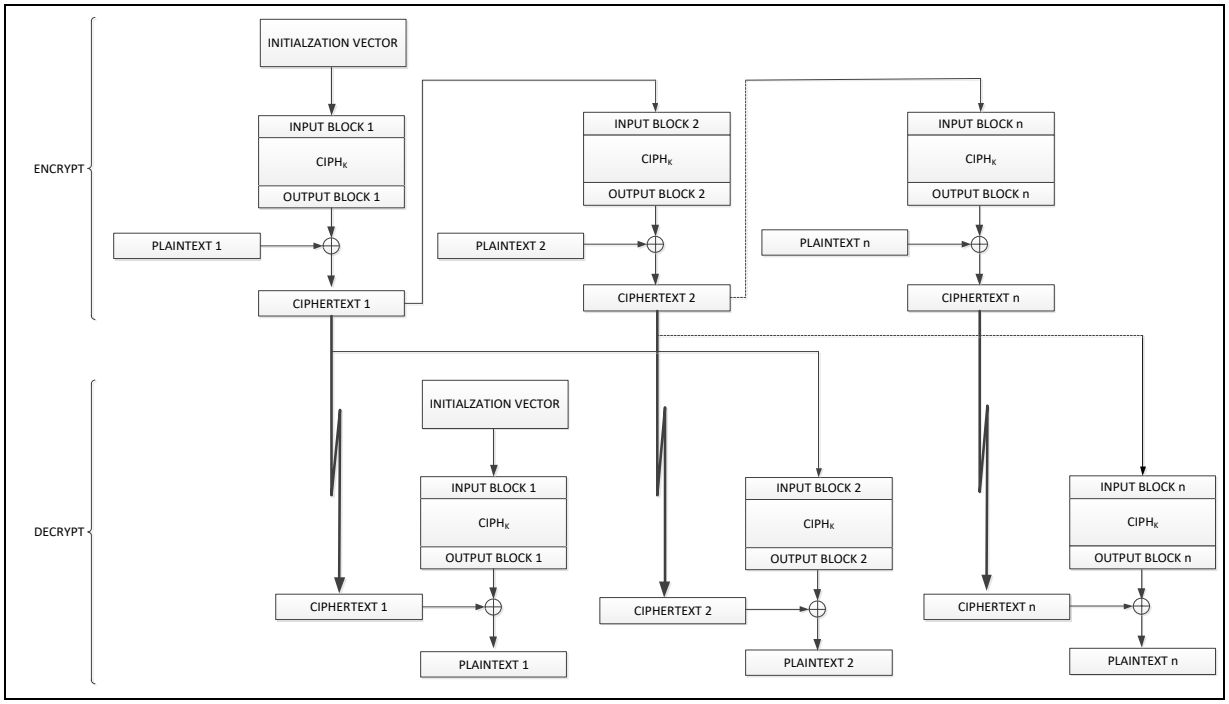


Figure 6.40-5 Cipher Feedback Mode

Output Feedback Mode

The Output Feedback (OFB) mode is a confidentiality mode that features the iteration of the forward cipher on an IV to generate a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The OFB mode requires that the IV is a nonce, i.e., the IV must be unique for each execution of the mode under the given key.

The OFB mode requires a unique IV for every message that is ever encrypted under the given key. If, contrary to this requirement, the same IV is used for the encryption of more than one message, then the confidentiality of those messages may be compromised. Confidentiality may be similarly be compromised if any of the input blocks to the forward cipher function for the encryption of a message is designated as the IV for the encryption of another message under the given key.

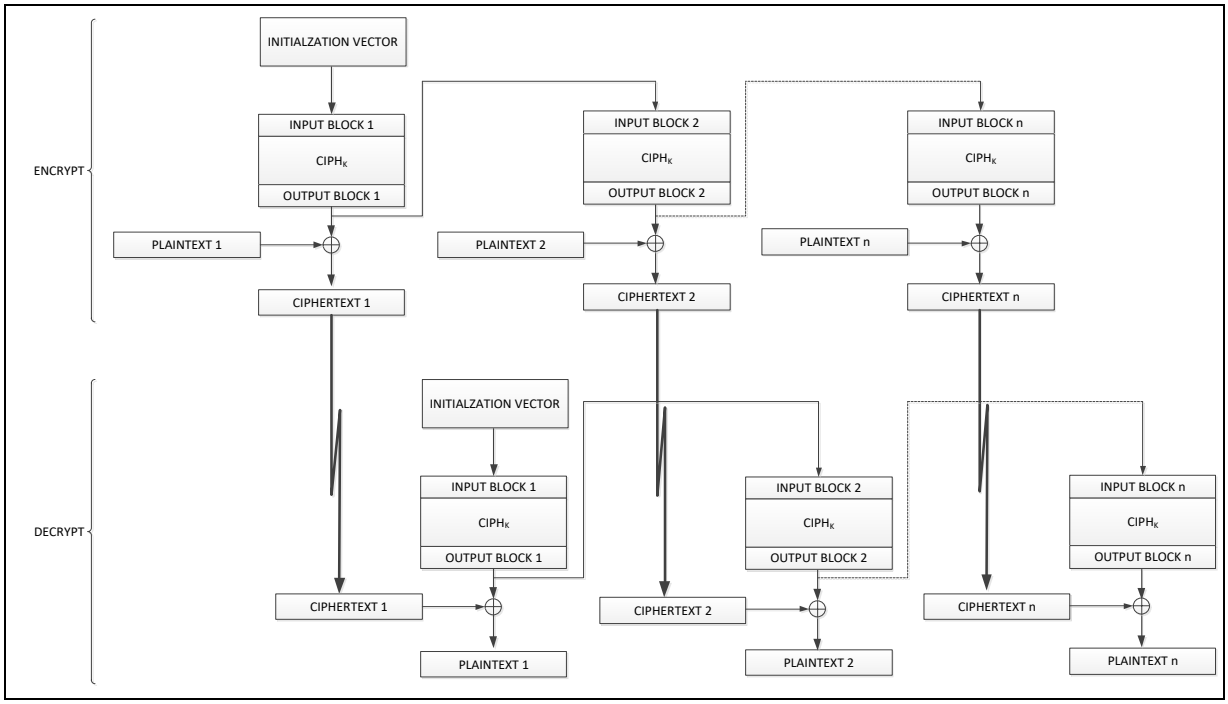


Figure 6.40-6 Output Feedback Mode

Counter Mode (CTR)

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The sequence of counters must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single message: across all of the messages that are encrypted under the given key, all of the counters must be distinct.

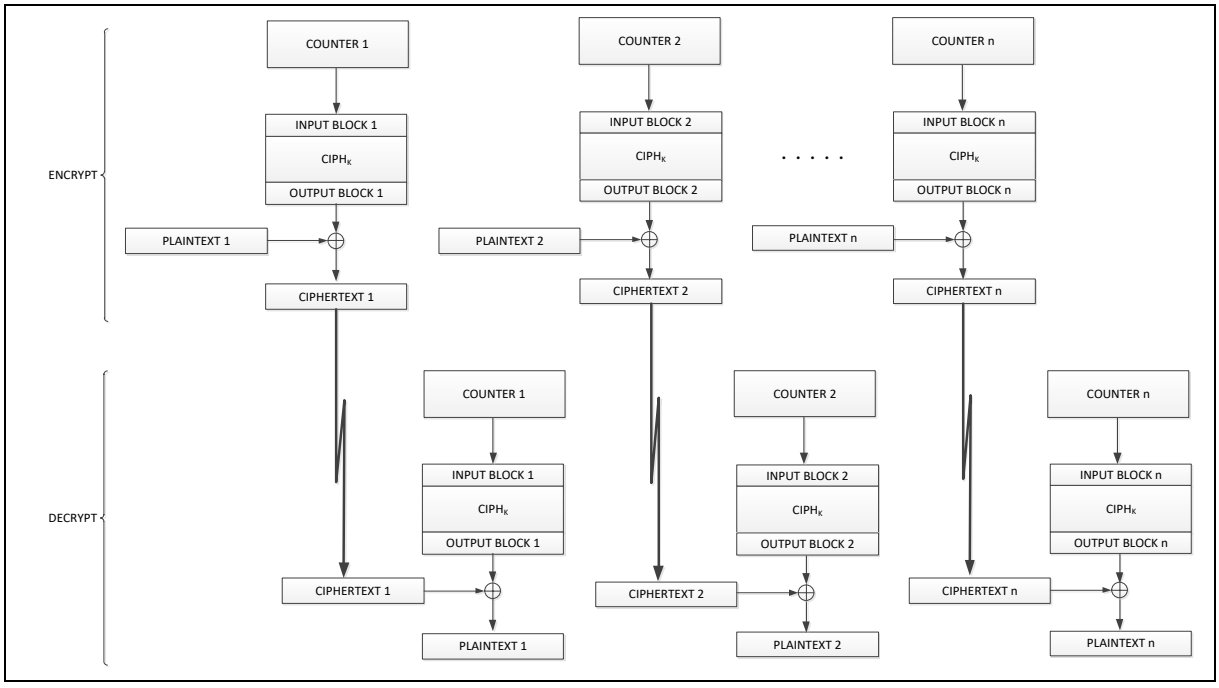


Figure 6.40-7 Counter Mode

CBC Ciphertext-Stealing 1 Mode (CBC-CS1)

Figure 6.41-8 illustrates the CBC-CS1-Encrypt algorithm for the case that P_n^* is a partial block. The cryptographic accelerator would append P_n^* with '0' to form a complete block P_n.

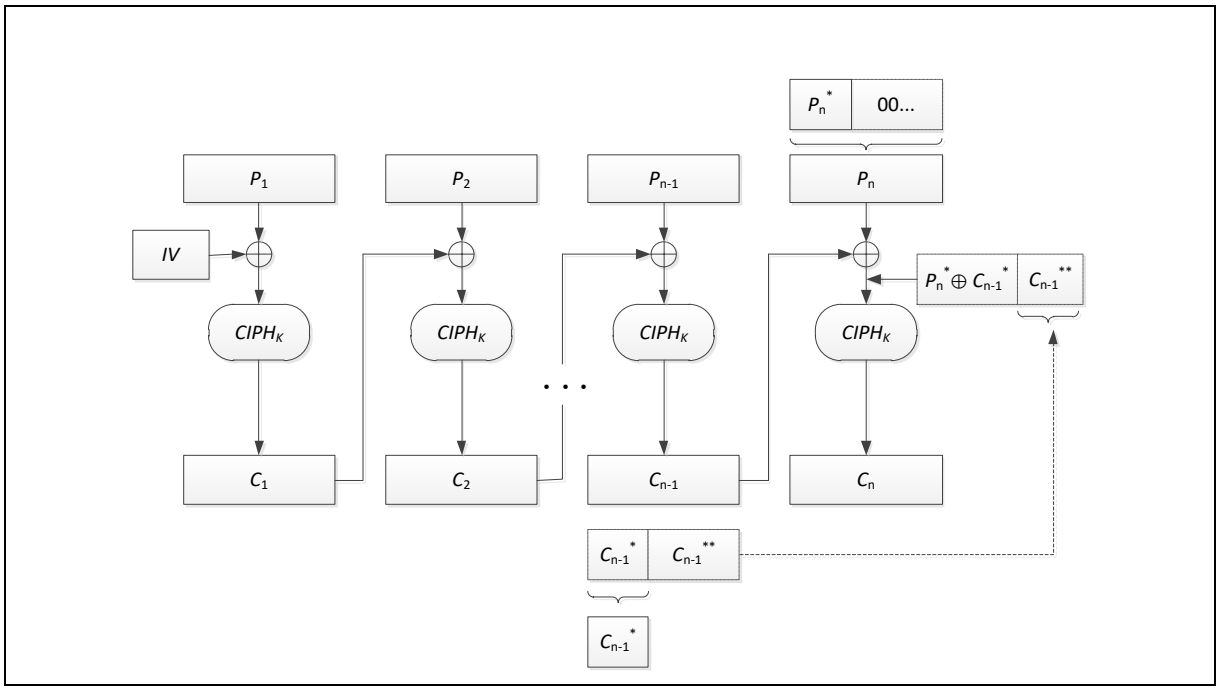


Figure 6.40-8 CBC-CS1 Encryption

Figure 6.41-9 illustrates the CBC-CS1-Decrypt algorithm for the case that C_{n-1}^* is a partial block.

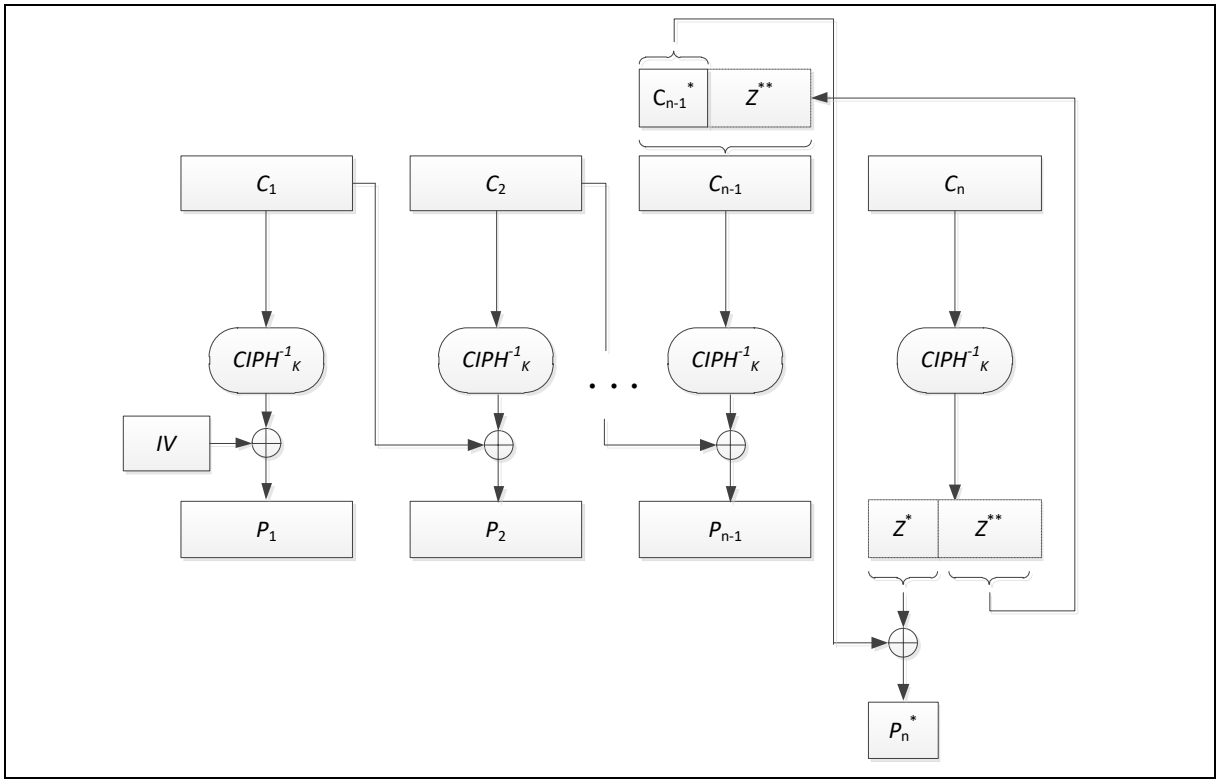


Figure 6.40-9 CBC-CS1 Decryption

CBC Ciphertext-Stealing 2 Mode (CBC-CS2)

When P_n^* is a partial block, then CBC-CS2-Encrypt and CBC-CS1-Encrypt differ only in the ordering of C_{n-1}^* and C_n.

CBC Ciphertext-Stealing 3 Mode (CBC-CS3)

C_{n-1}^* and C_n are unconditionally swapped, i.e., even when C_{n-1}^* is a complete block; therefore, CBC-CS3 is not strictly an extension of CBC mode. In the other case, i.e., when C_{n-1}^* is a nonempty partial block, CBC-CS3-Encrypt is equivalent to CBC-CS2-Encrypt.

Counter with CBC-MAC Mode (CCM)

The Counter with Cipher Block Chaining-Message Authentication Coder (CCM) mode is an algorithm for authenticated encryption/decryption with associated data. It provides the assurance of the confidentiality of data via CTR mode and the authenticity of the confidential data via CBC-MAC function. The input data of AES CCM includes plaintext/ciphertext (denoted P/C in CCM) in encryption/decryption, additional associated data (denoted A in CCM) and nonce (denoted N in CCM). The output data of CCM are C/P in encryption/decryption. In CCM, the authentication tag (denoted T in CCM) is encrypted in the last block of ciphertext C and decrypted from the same block. Figure 6.40-10 illustrates the CCM encryption and decryption flow. User can refer the document “Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality” to obtain the definition of all symbols. As shown in Figure 6.40-10, the CCM encryption and decryption has four main steps. To simplify hardware architecture, this accelerator only supports step 2 of encryption, step 3 of encryption, step 1 of decryption and step 3 of decryption. Thus, the input blocks structure of AES CCM in this accelerator are restricted to the following rules

1. Input blocks of encryption = {B_0 || … || B_r || B_{r-m} || B_{r-m+1} || … || B_r} ; where B_0 to B_r are generated from Formatting function.
2. Input blocks of decryption = {B_0 || … || B_r || B_{r-m} || B_{r-m+1} || … || B_r} ; where B_0 to B_{r-m} are generated from Formatting function and B_{r-m+1} to B_r is replaced by the ciphertext C (without the $T \oplus S_0$ block).

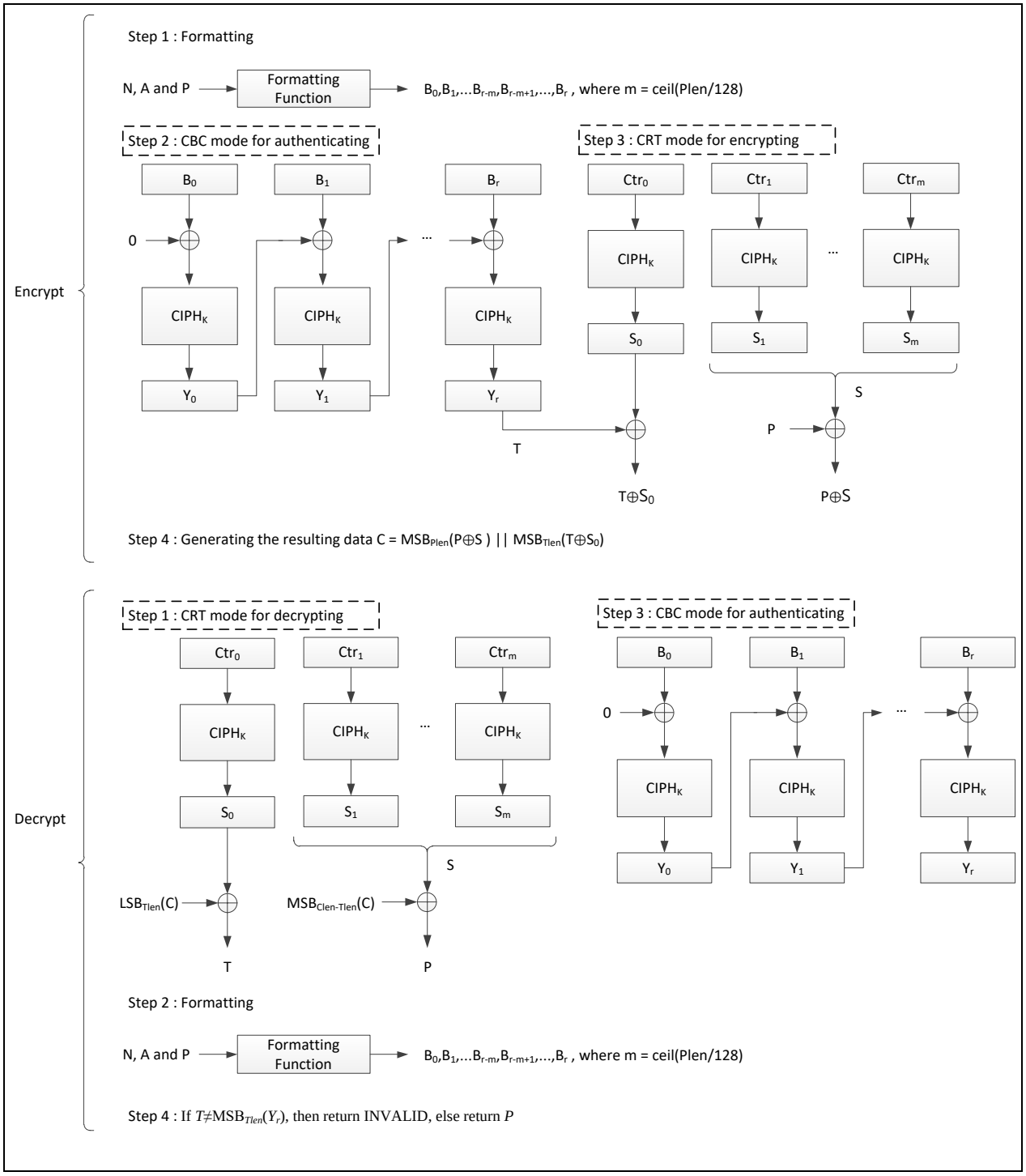


Figure 6.40-10 CCM Encryption and Decryption Flow

The output blocks structure of AES CCM are defined by the following rule

3. Output blocks of encryption = { MSB_{Plen} (P⊕S) || (m*128-Plen) bits redundant data || T⊕S_0}
4. Output blocks of decryption = { MSB_{Plen} (C⊕S) || (m*128-Plen) bits redundant data || Y_r⊕S_0}

Both input and output blocks in CCM must be block alignment (128 bits), except for the last input block

in the plaintext or ciphertext. Before starting every AES CCM operation, users have to write the byte count of A (i.e., (r-m+1)*16 bytes in Figure 6.40-10) and P (i.e., Plen/8 bytes in Figure 6.40-10) to the registers CRPT_AES_GCM_ACNT0/1 and CRPT_AES_GCM_PCNT0/1, respectively. Afterward, users must write initial vectors (i.e., Ctr_0 in Figure 6.40-10) to registers CRYPTO_AES_IV0 ~ CRYPTO_AES_IV3. After finishing CCM mode, users can obtain the ciphertext (i.e., $MSB_{Plen}(P \oplus S)$) or plaintext (i.e., $MSB_{Plen}(C \oplus S)$) with an authentication tag (i.e., $T \oplus S_0$ or $Y_r \oplus S_0$). In the end of CCM decryption, users can compare the last output block (i.e., $MSB_{Tlen}(T \oplus S_0)$) with the last input block of ciphertext C (i.e., $MSB_{Tlen}(C_{m+1})$) to make sure the authenticity of the confidential data in step 4. When adopting the DMA cascade mode in CCM mode, users must use DMA automatically feedback function by controlling the FBOUT and FBIN bits in CRYPTO_AES_CTL to trigger DMA engine to automatically write and read all required information according to the DMA feedback address register (i.e., CRYPTO_AES_FBADDR). Note that users must allocate 18 words continued memory space from the address CRYPTO_AES_FBADDR at least.

Galois/Counter Mode (GCM)

The Galois/Counter (GCM) mode is an algorithm for authenticated encryption/decryption with associated data. It provides the assurance of the confidentiality of data via CTR mode and the authenticity of the confidential data via GHASH function (i.e., binary Galois finite field hash function). The input data of GCM includes plaintext/ciphertext (denoted P/C in GCM) in encryption/decryption, additional authenticated data (denoted A in GCM) and initialization vector (denoted IV in GCM). The output data of GCM are C/P in encryption/decryption and an authentication tag (denoted T in GCM). Figure 6.40-11 illustrates the GCM encryption/decryption algorithm. User can refer the document "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC" to obtain the definition of all symbols in this figure. In Figure 6.40-11, the rectangle with gray color means input blocks, and the rectangle with dotted line represents output blocks. The input blocks structure of AES GCM for this cryptographic accelerator are restricted to the following rules

1. Input blocks = {IV || 0^{31} || 1 || A || 0^{V} || P/C || 0^{U}} ; if len(IV) == 96
2. Input blocks = {IV || 0^{S} || 0^{64} || len(IV) || A || 0^{V} || P/C || 0^{U}} ; if len(IV) != 96
3. The output blocks structure of AES GCM are defined by the following rule
4. Output blocks = {C/P || U bits redundant data || T/T'}

Both input and output blocks in GCM must be block alignment (128 bits). Before starting every AES GCM operation, users have to write the byte count of IV, A and P to the registers CRPT_AES_GCM_IVCNT0/1, CRPT_AES_GCM_ACNT0/1 and CRPT_AES_GCM_PCNT0/1, respectively. Besides, this cryptographic accelerator also provides the GHASH function of AES GCM. The input blocks structure of AES GHASH are restricted to the block alignment. After finishing AES GHASH operation, cryptographic accelerator will output a hash value (128-bits). In the end of GCM decryption, users can compare the value of T' with the one of T to make sure the authenticity of the confidential data. When adopting the DMA cascade mode in GCM mode, users must use DMA automatically feedback function by controlling the FBOUT and FBIN bits in CRYPTO_AES_CTL to trigger DMA engine to automatically write and read all required information according to the DMA feedback address register (i.e., CRYPTO_AES_FBADDR). Note that users must allocate 18 words continued memory space at least from the address CRYPTO_AES_FBADDR.

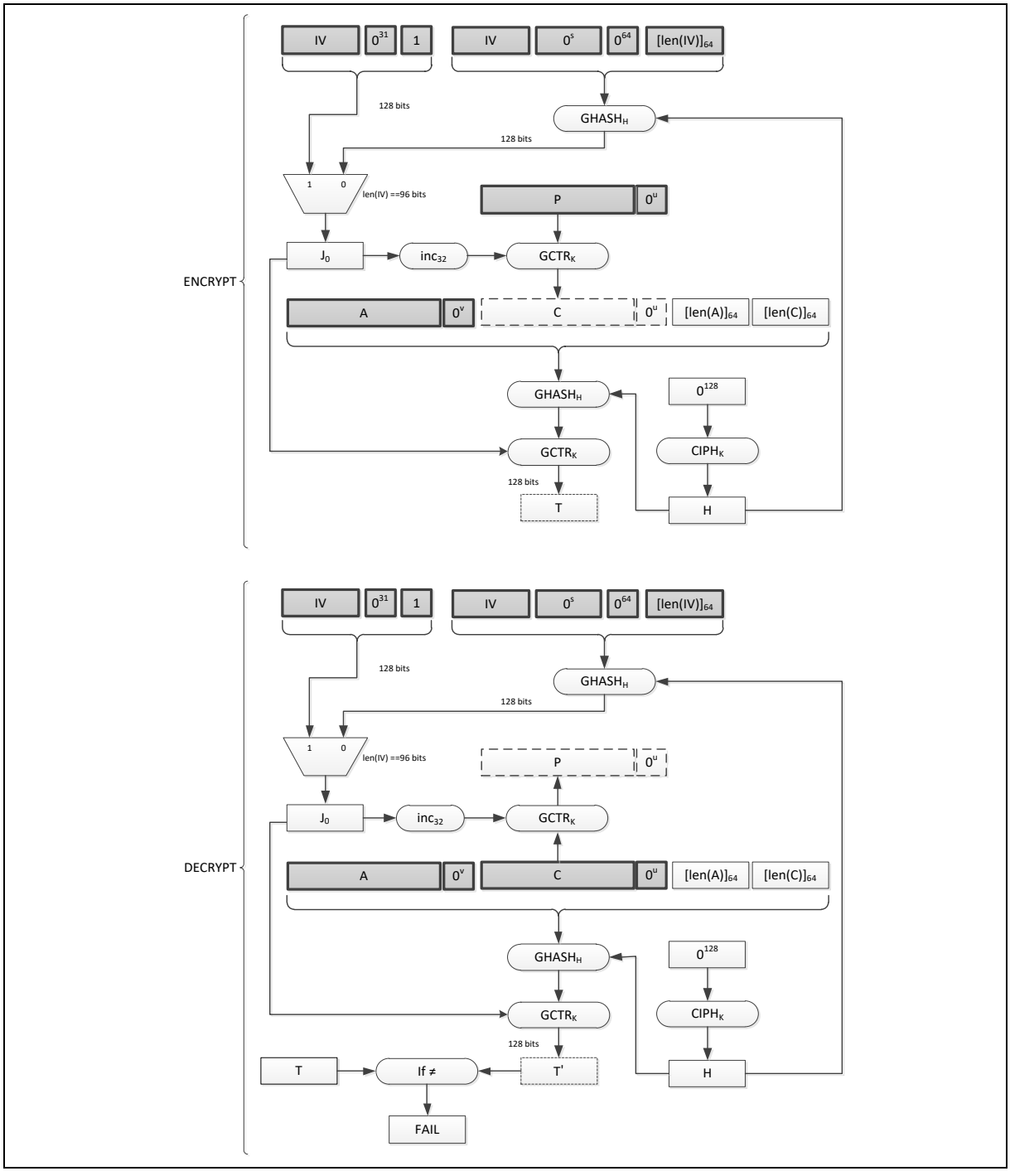


Figure 6.40-11 GCM Encryption and Decryption Flow

Refer to the following programming steps for how to program the AES related registers.

AES DMA Mode Programming Flow

1. Write 1 to AESIEN (CRYPTO_INTEN[0]) to enable AES interrupt if needed.
2. Program AES key information to register CRYPTO_AES_KSCTL when key is from the key store.
3. Program AES key to registers CRYPTO_AES_KEY0 ~ CRYPTO_AES_KEY7 when AES key

is not from the key store.

4. Program initial vectors to registers CRYPTO_AES_IV0 ~ CRYPTO_AES_IV3.
5. Program the total byte count of IV, A and P to the registers CRPT_AES_GCM_IVCNT0/1, CRPT_AES_GCM_ACNT0/1 and CRPT_AES_GCM_PCNT0/1 in AES CCM or GCM, respectively. Note that IV is not necessary for AES CCM.
6. Program DMA source address to register CRYPTO_AES_SADDR.
7. Program DMA destination address to register CRYPTO_AES_DADDR.
8. Program DMA byte count to register CRYPTO_AES_CNT.
9. Program DMA feedback address to register CRYPTO_AES_FBADDR (18 words memory space must be reserved for feedback information) if DMA automatically feedback function is needed.
10. Program AES control register CRYPTO_AES_CTL for FBOUT and FBIN to trigger AES DMA engine to write and read feedback information according to CRYPTO_AES_FBADDR if needed.
11. Configure AES control register CRYPTO_AES_CTL for key protection(KEYPRT), DMA input/output swap(INSWAP/OUTSWAP), encryption/decryption(ENCRYPTO), operational mode(OPMODE), DMA mode enable (DMAEN), DMA cascade control(DMACSCAD) and key size(KEYSZ). Write 1 to DMALAST (CRYPTO_AES_CTL[5]) if enabled DMACSCAD and current operation is last cascade operation.
12. Write input data to DMA source address with selected DMA byte count.
13. Write 1 to START(CRYPTO_AES_CTL[0]) to start AES encryption/decryption.
14. Wait for the AES interrupt flag AESIF (CRYPTO_INTSTS[0]) to be set or AES busy flag (CRYPTO_AES_STS[0]) to be set to zero.
15. Read output data from DMA destination address with selected DMA byte count.
16. Repeat step 6 to step 12 until all data processed if enabled DMACSCAD (CRYPTO_AES_CTL[6]).

AES Non-DMA Mode Programming Flow

1. Write 1 to AESIEN (CRYPTO_INTEN[0]) to enable AES interrupt if needed.
2. Program AES key information to register CRYPTO_AES_KSCTL when key is from the key store.
3. Program AES key to register CRYPTO_AES_KEY0 ~ CRYPTO_AES_KEY7 when AES key is not from the key store.
4. Program initial vectors to register CRYPTO_AES_IV0 ~ CRYPTO_AES_IV3.
5. Program the total byte count of IV, A and P to registers CRPT_AES_GCM_IVCNT0/1, CRPT_AES_GCM_ACNT0/1 and CRPT_AES_GCM_PCNT0/1 in AES CCM and GCM, respectively. Note that IV is not necessary for AES CCM.
6. Configure AES control register CRYPTO_AES_CTL for key protection (KEYPRT), input/output swap (INSWAP/OUTSWAP), encryption/decryption (ENCRYPTO), operational mode (OPMODE), DMA cascade control and key size(KEYSZ).
7. Write 1 to START(CRYPTO_AES_CTL[0]) to start AES encryption/decryption.
8. Polling INBUFFULL(CRYPTO_AES_STS[9]) and OUTBUFEMPTY(CRYPTO_AES_STS[16]). If INBUFFULL(CRYPTO_AES_STS[9]) is 0, write 32 bits input data to CRYPTO_AES_DATIN. If OUTBUFEMPTY(CRYPTO_AES_STS[16]) is 0, read 32 bits data from CRYPTO_AES_DATOUT.
9. Repeat step 8 until 128 bits data (16 bytes) are written to and read from AES engine.

10. Write 1 to DMALAST(CRYPTO_AES_CTL[5]) if current operation is last operation.
11. Write data byte count of last operation to register CRYPTO_AES_CNT if current operation is last operation.
12. Repeat Steps 7 to 11 until all data processed.

Note: To bring the side-channel protection of AES into full play, it is suggested that user must execute "PRNG get the random number with the true random number seed from TRNG" at least once before starting the first AES operation after every CRYPTO or system reset.

6.40.5.3 SHA (Secure Hash Algorithm)

SHA accelerator supports SHA1, SHA2 and SHA3 algorithm from U.S NIST.

User can refer to the following steps to understand how to program the SHA/HMAC related registers.

SHA DMA Mode Programming Flow

1. Write 1 to HMACIEN (CRYPTO_INTEN[24]) to enable SHA interrupt if needed.
2. Configure SHA control register CRYPTO_HMAC_CTL for SHA engine input/output data swap(INSWAP/OUTSWAP), DMA mode(DMAEN), SHA3 engine enable bit(SHA3EN) and SHA operation mode(OPMODE). Clear HMACEN (CRYPTO_HMAC_CTL[4]) to select SHA mode.
3. This step is only for cascade mode. Write 1 to DMACSCAD (CRYPTO_HMAC_CTL[6]) and DMAFIRST (CRYPTO_HMAC_CTL[4]) in the first time of cascade mode computation. Write 1 to DMACSCAD (CRYPTO_HMAC_CTL[6]) and DMALAST (CRYPTO_HMAC_CTL[5]) in the final time of cascade mode computation. In other cases of cascade mode computation, only write 1 to DMACSCAD(CRYPTO_HMAC_CTL[6]).
4. This step is only for manual cascade mode. Write the feedback information of the last cascade mode computation to feedback registers (CRYPTO_HMAC_FDBCK0~53 for SHA1/2 and CRYPTO_HMAC_FDBCK0~87 for SHA3).
5. This step is only for automatic cascade mode by DMA automatically feedback function. Program DMA feedback address to register CRYPTO_HMAC_FBADDR for reading or writing feedback information in SRAM (users must reserve 54 words space for SHA1/2 feedback information and 88 words space for SHA3 feedback information).
6. This step is only for automatic cascade mode by DMA automatically feedback function. Program HMAC control register CRYPTO_HMAC_CTL for FBOUT and FBIN to trigger HMAC DMA engine to output and input feedback information according to CRYPTO_HMAC_FBADDR.
7. Program DMA source address to register CRYPTO_HMAC_SADDR.
8. Program DMA byte count to register CRYPTO_HMAC_DMACNT.
9. Write input data to DMA source address with selected DMA byte count.
10. Write 1 to START(CRYPTO_HMAC_CTL[0]) to start SHA encryption.
11. Wait for the SHA interrupt flag HMACIF (CRYPTO_INTSTS[24]) to be set.
12. Read output digest (SHA-160: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4, SHA-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, SHA-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, SHA-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, SHA-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15, SHA3-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, SHA3-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, SHA3-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, SHA3-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15, SHAKE-128: CRYPTO_HMAC_SHAKEDGST0 ~

CRYPTO_HMAC_SHAKEDGST41, SHAKE-256: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST33).

13. If users need to produce more SHAKE digests, then jump to step14 or else step 17.
14. Write 1 to NEXTDGST(CRYPTO_HMAC_CTL[24]).
15. Wait for the SHAKE busy flag SHAKEBUSY(CRYPTO_HMAC_STS[2]) to be 0.
16. Read output digest (SHAKE-128: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST41, SHAKE-256: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST33) and back to step 14 if more SHAKE digests are required.
17. Write 1 to FINISHDGST(CRYPTO_HMAC_CTL[25]) to finish SHAKE function.

Note: The KEYCNT(CRYPTO_HMAC_KEYCNT[31:0]) keeps the byte count of key that SHA engine operates. In SHA with DMA cascade mode, the CRYPTO_HMAC_DMACNT must keep block size alignment, except for the final time of cascade mode computation. The block size of HMAC-SHA-512 and HMAC-SHA-384 are 128 bytes; HMAC-SHA-256, HMAC-SHA-224 and HMAC-SHA-160 are 64 bytes; HMAC-SHA3-512 is 72 bytes; HMAC-SHA3-384 is 104 bytes; HMAC-SHA3-256 is 136 bytes; HMAC-SHA3-224 is 144 bytes; SHAKE128 is 168 bytes; SHAKE256 is 136 bytes. After finishing every cascade mode computation, user needs to store the feedback information from all HMAC feedback registers if the further computation will cascade the current one. Both SHAKE128 and SHAKE256 do not support DMA cascade mode.

SHA Non-DMA mode programming flow:

1. Configure SHA control register CRYPTO_HMAC_CTL for SHA engine input/output data swap(INSWAP/OUTSWAP), SHA3 engine enable bit(SHA3EN) and SHA operation mode(OPMODE). Clear HMACEN (CRYPTO_HMAC_CTL[4]) to select SHA mode.
2. If it is the last input word, set DMALAST(CRYPTO_HMAC_CTL[5]).
3. Write 1 to START(CRYPTO_HMAC_CTL[0]) to start SHA encryption.
4. Wait for the SHA data input request DATINREQ(CRYPTO_HMAC_STS[16]) to be set.
5. Write one word of input data to CRYPTO_HMAC_DATIN.
6. Repeat Step 2 to 5 until all input words are written into SHA engine.
7. Wait for the BUSY (CRYPTO_HMAC_STS[0]) to be cleared.
8. Read output digest (SHA-160: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4, SHA-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, SHA-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, SHA-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, SHA-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15, SHA3-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, SHA3-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, SHA3-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, SHA3-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15, SHAKE-128: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST41, SHAKE-256: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST33).
9. If users need to produce more SHAKE digests, then jump to Step10 or else Step 13.
10. Write 1 to NEXTDGST(CRYPTO_HMAC_CTL[24]) if more SHAKE digests are required.
11. Wait for the SHAKE busy flag SHAKEBUSY(CRYPTO_HMAC_STS[2]) to be 0.
12. Read output digest (SHAKE-128: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST41, SHAKE-256: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST33) and back to Step 9.
13. Write 1 to FINISHDGST(CRYPTO_HMAC_CTL[25]) to finish SHAKE function.

Note: The KEYCNT(CRYPTO_HMAC_KEYCNT[31:0]) keeps the byte count of key that SHA engine operates.

6.40.5.4 HMAC (Keyed-Hash Message Authentication Code)

The Keyed-Hash Message Authentication Code is a specific construction for calculating a message authentication code involving a cryptographic hash function in combination with a secret cryptographic key. Any cryptographic hash function, such as SHA-256, may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-SHA-256 accordingly. Please note that the minimum size of message authentication code is 1 byte and the minimum size of key is 4 bytes in this HMAC accelerator. User can refer to the following steps to understand how to program the SHA/HMAC related registers.

HMAC DMA Mode Programming Flow

1. Write 1 to HMACIEN(CRYPTO_INTEN[24]) to enable HMAC interrupt if needed.
2. Program HMAC key information to register CRPT_HMAC_KSCTL when key is from the key store.
3. Configure SHA/HMAC control register CRYPTO_HMAC_CTL for HMAC engine input/output data swap(INSWAP/OUTSWAP), DMA mode(DMAEN), SHA3 engine enable bit(SHA3EN) and HMAC operation mode(OPMODE). Set HMACEN(CRYPTO_HMAC_CTL[4]) to select HMAC mode.
4. This step is only for cascade mode. Write 1 to DMACSCAD (CRYPTO_HMAC_CTL[6]) and DMAFIRST (CRYPTO_HMAC_CTL[4]) in the first time of cascade mode computation. Write 1 to DMACSCAD (CRYPTO_HMAC_CTL[6]) and DMALAST (CRYPTO_HMAC_CTL[5]) in the final time of cascade mode computation. In other cases of cascade mode computation, only write 1 to DMACSCAD(CRYPTO_HMAC_CTL[6]).
5. This step is only for manual cascade mode. Write the feedback information of the last cascade mode computation to feedback registers (CRYPTO_HMAC_FDBCKx)
6. This step is only for automatic cascade mode by DMA automatically feedback function. Program DMA feedback address to register CRYPTO_HMAC_FBADDR for reading or writing feedback information in SRAM (user must reserve 54 words space for feedback information).
7. This step is only for automatic cascade mode by DMA automatically feedback function. Program HMAC control register CRYPTO_HMAC_CTL for FBOUT and FBIN to trigger HMAC DMA engine to output and input feedback information according to CRYPTO_HMAC_FBADDR.
8. Program DMA source address to register CRYPTO_HMAC_SADDR.
9. Program DMA byte count to register CRYPTO_HMAC_DMACNT.
10. Write input data to DMA source address with selected DMA byte count.
11. Write 1 to START(CRYPTO_HMAC_CTL[0]) to start HMAC encryption.
12. Wait for the HMAC interrupt flag HMACIF(CRYPTO_INTSTS[24]) to be set.
13. Read output digest (HMAC-SHA-160: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4, HMAC-SHA-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, HMAC-SHA-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, HMAC-SHA-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, HMAC-SHA-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15, SHA3-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, SHA3-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, SHA3-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, SHA3-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15, SHAKE-128: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST41, SHAKE-256: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST33).

Note: The KEYCNT(CRYPTO_HMAC_KEYCNT[31:0]) keeps the byte count of key that HMAC engine operates. In HMAC DMA with cascade mode, the CRYPTO_HMAC_DMACNT must keep block size alignment, except for the final time of cascade mode computation. Do not merge the key and data in the same block when executing HMAC DMA with cascade mode computation. If the KEYCNT is more than one block size and not block size alignment, user must padding 0x0 in the end of key to ensure that its length is block size alignment. In above case, user must add the length of padding bits to the CRYPTO_HMAC_DMACNT, not to KEYCNT. For example, user wants to execute a HMAC-SHA-256 (block size is 64 bytes) with DMA cascade mode computation, 96 bytes key size and 96 bytes data size. In this case, user must allocate two block size memory space for HMAC key (not contiguous memory allocation). The first block is the 64 bytes of HMAC key[511:0] and the second one is 32 bytes of HMAC key [767:512] and 32 bytes zero padding. If the first cascade computation is only for HMAC key, user must set CRYPTO_HMAC_DMACNT to 0x80 and KEYCNT to 0x60. After finishing every cascade mode computation, user needs to store the feedback information from all HMAC feedback registers if the further computation will cascade the current one. The size of HMAC key in key store must be block size (i.e., The block size of HMAC-SHA-512 and HMAC-SHA-384 are 128 bytes; HMAC-SHA-256, HMAC-SHA-224 and HMAC-SHA-160 are 64 bytes; HMAC-SHA3-512 is 72 bytes; HMAC-SHA3-384 is 104 bytes; HMAC-SHA3-256 is 136 bytes; HMAC-SHA3-224 is 144 bytes; SHAKE128 is 168 bytes; SHAKE256 is 136 bytes). Both SHAKE128 and SHAKE256 do not support HMAC algorithm in this accelerator.

HMAC Non-DMA Mode Programming Flow

1. Configure SHA/HMAC control register CRYPTO_HMAC_CTL for SHA/HMAC engine input/output data swap(INSWAP/OUTSWAP), SHA3 engine enable bit(SHA3EN) and HMAC operation mode(OPMODE). Set HMACEN(CRYPTO_HMAC_CTL[4]) to select HMAC mode.
2. If it is the last input word, set DMALAST(CRYPTO_HMAC_CTL[5]).
3. Program HMAC key information to register CRPT_HMAC_KSCTL.
4. Write 1 to START(CRYPTO_HMAC_CTL[0]) to start HMAC encryption.
5. Wait for the HMAC data input request DATINREQ(CRYPTO_HMAC_STS[16]) to be set.
6. Write one word of input data to CRYPTO_HMAC_DATIN.
7. Repeat Step 2 to 5 until all input words are written into SHA engine.
8. Wait for the BUSY (CRYPTO_HMAC_STS[0]) to be cleared.
9. Read output digest (HMAC-SHA-160: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4, HMAC-SHA-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, HMAC-SHA-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, HMAC-SHA-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, HMAC-SHA-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15, SHA3-224: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6, SHA3-256: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7, SHA3-384: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11, SHA3-512: CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15, SHAKE-128: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST41, SHAKE-256: CRYPTO_HMAC_SHAKEDGST0 ~ CRYPTO_HMAC_SHAKEDGST33).

Note: The KEYCNT(CRYPTO_HMAC_KEYCNT[31:0]) keeps the byte count of key that HMAC engine operates. The maximum size of HMAC key in the key store is block size (i.e., HMAC-SHA-512 and HMAC-SHA-384 are 128 bytes; others are 64 bytes).

6.40.5.5 ECC (Elliptic Curve Cryptography)

Elliptic Curve Cryptography (ECC) is a famous approach of public-key cryptosystems. Recently, many protocols and applications utilize the algebraic cyclic group characters of elliptic curves over finite field to build cryptographic systems. All points of an elliptic curve will follow the formula of elliptic curve: $y^2 \equiv x^3+A*x+B \pmod N$ or $B*y^2 \equiv x^3+A*x^2+x \pmod N$ in GF(p) and $y^2+x*y \equiv x^3+A*x^2+B \pmod N$ in GF(2^m).

Thus, ECC accelerator can also support SM2 algorithm from CHINA state cryptography administration. Figure 6.40-12 exhibits the main hierarchy chart of ECC applications. The often appeared parameters and corresponding registers are shown in Table 6.40-4.

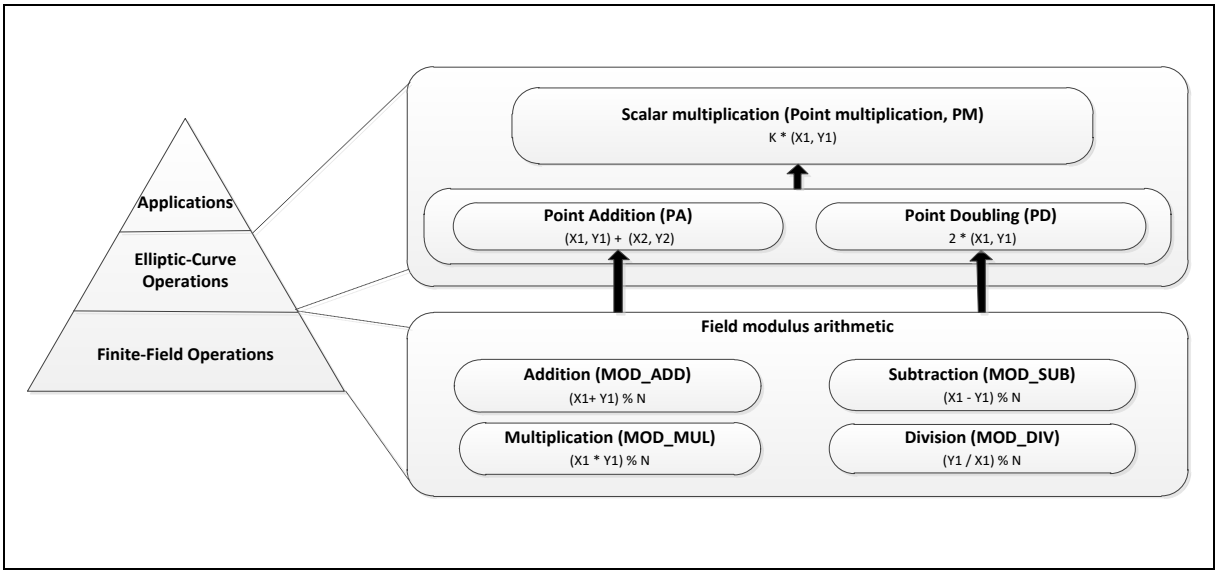


Figure 6.40-12 Main Hierarchy Chart of ECC

Parameter	Description	Corresponding Register
X1	The x-coordinate of point1	CRYPTO_ECC_X1_00~ CRYPTO_ECC_X1_17
Y1	The y-coordinate of point1	CRYPTO_ECC_Y1_00~ CRYPTO_ECC_Y1_17
X2	The x-coordinate of point2	CRYPTO_ECC_X2_00~ CRYPTO_ECC_X2_17
Y2	The y-coordinate of point2	CRYPTO_ECC_Y2_00~ CRYPTO_ECC_Y2_17
A	The curve parameter A	CRYPTO_ECC_A_00~ CRYPTO_ECC_A_17
B	The curve parameter B	CRYPTO_ECC_B_00~ CRYPTO_ECC_B_17
N	The curve parameter N	CRYPTO_ECC_N_00~ CRYPTO_ECC_N_17
M	The curve length	CRYPTO_ECC_CTL[31:22]
K	The scalar constant	CRYPTO_ECC_K_00~ CRYPTO_ECC_K_17

Table 6.40-4 ECC Parameters and Corresponding Registers Table

Scalar multiplication (point multiplication) is the core operation in ECC applications. The computation of scalar multiplication is composed of point addition and point doubling operations. Moreover, there are many finite field modulus arithmetic operations in the formula of point addition and doubling operation. To accelerate ECC applications, an elliptic curve cryptographic accelerator is provided, which can process not only three point operations in both GF(p) and GF(2^m) but also four modulus operations in GF(p). Before starting ECC accelerator, user must provide the required input data of ECC operation include point coordinates (X1, Y1, X2, Y2), curve parameters (A, B, N, M) and scalar data (K) in Table 6.40-5 .The mark "√" means that the input data is necessary for this operation. The detail definition of input data and the corresponding registers in the ECC accelerator are exhibited in the next section Register Map. When executing point multiplication (PM) with side-channel attack protection (SCAP = CRYPTO_ECC_CTL[14] =1), users must write the order value of elliptic curve into the registers from CRYPTO_ECC_X2_00 to CRYPTO_ECC_X2_17. To bring the side-channel protection of ECC into full play, it is suggested that user must execute "PRNG gets the random number with the true random

number seed from TRNG" at least once before starting the first ECC operation after every CRYPTO or system reset.

After ECC accelerator is finished, all point operations will generate an output point includes x-coordinate in the registers from CRYPTO_ECC_X1_00 to CRYPTO_ECC_X1_17 and y-coordinate in the registers from CRYPTO_ECC_Y1_00 to CRYPTO_ECC_Y1_17. In all modulus operations, ECC accelerator will only produce an output result in the registers from CRYPTO_ECC_X1_00 to CRYPTO_ECC_X1_17.

Operation	PM	PA	PD	MOD_DIV	MOD_MUL	MOD_ADD	MOD_SUB	ECDSAS
ECCOP[1:0]	00	10	11	01	01	01	01	01
MODOP[1:0]	XX	XX	XX	00	01	10	11	XX
ECDSAS	X	X	X	X	X	X	X	1
X1	√	√	√	√	√	√	√	√
Y1	√	√	√	√	√	√	√	√
X2		√						√
Y2		√						√
A	√	√	√					
B	√		√					
N	√	√	√	√	√	√	√	√
M	√	√	√		√			√
K	√							

Table 6.40-5 Required Input Data of Various Operations

User can refer to the following steps to understand how to program the ECC related registers.

ECC DMA Mode Programming Flow

1. Write 1 to ECCIEN(CRYPTO_INTEN[22]) to enable ECC interrupt if needed.
2. Program DMA source address to register CRYPTO_ECC_SADDR.
3. Program DMA destination address to register CRYPTO_ECC_DADDR.
4. Program DMA word count to register CRYPTO_ECC_WORDCNT.
5. Program the starting register address of all input data in Table 6.40-5 that will update to the register CRYPTO_ECC_STARTRE.G.
6. Write input data to DMA source address with selected DMA word count.
7. Program ECC key information to register CRYPTO_ECC_KSCTL and CRYPTO_ECC_KSXY when key is form the key store.
8. Configure ECC control register CRYPTO_ECC_CTL for ECC accelerator, such as the start signal of ECC accelerator (START), ECDSA signature S generation (ECDSAS), ECDSA signature R generation (ECDSAR), differential fault attack protection (DFAP), DMA mode enable signal (DMAEN), field selection (FSEL), point operation mode (ECCOP), modulus operation mode (MODOP), curve selection (CSEL), side-channel attack protection (SCAP), the control signals of all input data registers (LDA, LDB, LDN, LDK, LDP1, LDP2), and the key length of elliptic curve (CURVEM).
9. Wait for the ECC interrupt flag ECCIF(CRYPTO_INTSTS[22]) to be set.

10. Read output data and then clear ECC interrupt flag ECCIF.

ECC Non-DMA Mode Programming Flow

1. Write all necessary input data in the corresponding registers according to in Table 6.40-5, such as CURVEA, CURVEB, CURVEN, SCALARK.
2. Program ECC key information to register CRYPTO_ECC_KSCTL and CRYPTO_ECC_KSXY when key is from the key store.
3. Configure ECC control register CRYPTO_ECC_CTL for ECC accelerator, such as the start signal of ECC accelerator (START), differential fault attack protection (DFAP), field selection (FSEL), point operation mode (ECCOP), modulus operation mode (MODOP), curve selection (CSEL), side-channel attack protection (SCAP), the control signals of all input data registers (LDA, LDB, LDN, LDK, LDP1, LDP2), and the key length of elliptic curve (CURVEM).
4. Wait for the BUSY (CRYPTO_ECC_STS[0]) to be cleared.
5. Read output digest and then clear ECC interrupt flag ECCIF.

Some Notices of ECC Accelerator

1. The key length support of ECC accelerator is from 163 to 571 bits.
2. All input and output data must be positive. (If the input data is negative, it must be added N).
3. Only when START and DMAEN (CRYPTO_ECC_CTL[0] and [7]) are assigned to 1 simultaneously, ECC DMA mode will be active.
4. When ECC engine is active (i.e., BUSY is 1 and DMABUSY (CRYPTO_ECC_STS[1]) is 0), user cannot modify all input data registers (CRYPTO_ECC_X1_00 ~ CRYPTO_ECC_K_17).
5. If user wants to stop ECC accelerator, please configures the STOP (CRYPTO_ECC_CTL[1]) to 1. **Note:** To avoid the transmission error of the next operation, BUSY signal will not be cleared immediately until the action of DMA is done.
6. The modulus operation not support for binary field.
7. The input data of modulus multiplication and division operation for PF must be less than N.
8. K is private key, so this register is write only.
9. When side-channel attack protection (i.e., CRYPTO_ECC_CTL[14] =1) is active, users must write the order value of elliptic curve into the registers from CRYPTO_ECC_X2_00 to CRYPTO_ECC_X2_17 beforehand.

The following describes the method of application about Modular operations, key pair generation, ECDSA, ECDSA with key store, ECDH and ECDH with key store.

Modular Operations

Modular addition in prime field: X1 = X1 + Y1 (mod N)

1. Write the modulus to N register.
2. Write the summand to X1 register and the addend to Y1 register.
3. Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
4. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
5. Set FSEL(CRYPTO_ECC_CTL[8]) to 1
6. Set START(CRYPTO_ECC_CTL[0]) to 1
7. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared

8. Read result from X1 registers

Modular subtraction in prime field: X1 = X1 - Y1 (mod N)

9. Write the modulus to N register.
10. Write the minuend to X1 register and the subtrahend to Y1 register.
11. Set MODOP(CRYPTO_ECC_CTL[12:11]) to 11
12. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
13. Set FSEL(CRYPTO_ECC_CTL[8]) to 1
14. Set START(CRYPTO_ECC_CTL[0]) to 1
15. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
16. Read result from X1 registers

Modular multiplication in prime field: X1 = X1 * Y1 (mod N)

17. Write the modulus to N register and the bit length of multiplication to register M.
18. Write the multiplicand to X1 register and the multiplier to Y1 register.
19. Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
20. Set MODOP(CRYPTO_ECC_CTL[12:11]) to 01
21. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
22. Set FSEL(CRYPTO_ECC_CTL[8]) to 1
23. Set START(CRYPTO_ECC_CTL[0]) to 1
24. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
25. Read result from X1 registers

Modular division in prime field: X1 = Y1 / X1 (mod N)

26. Write the modulus to N register.
27. Write the divisor to X1 register and the dividend to Y1 register.
28. Set MODOP(CRYPTO_ECC_CTL[12:11]) to 00
29. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
30. Set FSEL(CRYPTO_ECC_CTL[8]) to 1
31. Set START(CRYPTO_ECC_CTL[0]) to 1
32. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
33. Read result from X1 registers

Key Pair Generation

Public key generation function: Q = dG (mod N)

1. Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4.
2. Write the point G(x, y) to X1, Y1 registers according to Table 6.40-4 or program the key information of G(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from the key store
3. Write the private key d to K register according to Table 6.40-4 or program the key information

of d to register CRYPTO_ECC_KSCTL and set RSRCK (CRYPTO_ECC_KSCTL[5]) to 1 for updating K register from the key store.

4. Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
5. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
6. Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field
7. Set START(CRYPTO_ECC_CTL[0]) to 1
8. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
9. Read public key Q from X1, Y1 registers

SM2 Digital Signature Algorithm (SM2 DSA)

SM2 DSA signature generation steps:

1. If Z_A exits, go to step 3.
2. Calculate $Z_A = HASH_{256}(ENTL \| ID \| A \| B \| x_G \| y_G \| x_Q \| y_Q)$, where HASH is a cryptographic hashing algorithm, (i.e. SM3 or SHA-256)
    1) Prepare input data include the length of user's ID (ENTL), user's ID (ID), curve parameter (i.e., A and B), base point G (x_G, y_G) and public key Q (x_Q, y_Q)
    2) Use SM3 or SHA-256 to calculate Z_A
    3) Keep the leftmost 256 bits of Z_A
3. Calculate $e = HASH_{256}(Z_A \| m)$, where HASH is a cryptographic hashing algorithm, (i.e. SM3 or SHA-256)
    1) Prepare input data include Z_A and message m
    2) Use SM3 or SHA-256 to calculate e
    3) Keep the leftmost 256 bits of e
4. Calculate e = e (mod n)
    1) Write the curve order and curve length to N, M registers according to Table 6.40-4
    2) Write the e to X1 register and 0x0 to Y1 register.
    3) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
    4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
    5) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
    6) Set START(CRYPTO_ECC_CTL[0]) to 1
    7) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
    8) Read result e from X1 registers
    9) If $e \geq n$, go to step 4
5. Select a random integer k form [1, n-1]
    1) Note that n is order, not prime modulus or irreducible polynomial function
6. Compute x1 = x1 (mod n), where (x1, y1) = k * G.
    1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.40-4

2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.40-4
3) Write the base point G(x, y) to X1, Y1 registers according to Table 6.40-4
4) Write the random integer k to K register according to Table 6.40-4
5) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
6) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
7) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field
8) Set START(CRYPTO_ECC_CTL[0]) to 1
9) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
10) Write the curve order and curve length to N, M registers according to Table 6.40-4
11) Write 0x0 to Y1 registers
12) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 10
13) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
14) Set START(CRYPTO_ECC_CTL[0]) to 1
15) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
16) Read X1 registers to get x1

7. Conpute r = (e + x1) (mod n). If r = 0 or (r + k) = n, go to step 5
   1) Write the curve order and curve length to N, M registers according to Table 6.40-4
   2) Write the e to X1 register and x1 to Y1 register.
   3) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   5) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
   6) Set START(CRYPTO_ECC_CTL[0]) to 1
   7) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   8) Read result r from X1 registers
   9) If r = 0, go to step 5
   10) Write the curve order and curve length to N, M registers according to Table 6.40-4
   11) Write the r to X1 register and k to Y1 register.
   12) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
   13) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   14) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
   15) Set START(CRYPTO_ECC_CTL[0]) to 1
   16) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   17) Read result rk from X1 registers
   18) If rk = 0, go to step 5
8. Compute $s = (1 + d)^{-1} \times (k - d \times r) \pmod{n}$. If s = 0, go to step 5
   1) Write the curve order and curve length to N, M registers according to Table 6.40-4
   2) Write the d to X1 register and r to Y1 register.

3) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 01
4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
5) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
6) Set START(CRYPTO_ECC_CTL[0]) to 1
7) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
8) Read temporary result dr from X1 registers
9) Write the curve order and curve length to N, M registers according to Table 6.40-4
10) Write the k to X1 register and dr to Y1 register.
11) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 11
12) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
13) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
14) Set START(CRYPTO_ECC_CTL[0]) to 1
15) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
16) Read temporary result k_dr from X1 registers
17) Write the curve order and curve length to N, M registers according to Table 6.40-4
18) Write the d to X1 register and 0x1 to Y1 register.
19) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
20) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
21) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
22) Set START(CRYPTO_ECC_CTL[0]) to 1
23) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
24) Read temporary result d1 from X1 registers
25) Write the curve order and curve length to N, M registers according to Table 6.40-4
26) Write the d1 to X1 register and k_dr to Y1 register.
27) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 00
28) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
29) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
30) Set START(CRYPTO_ECC_CTL[0]) to 1
31) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
32) Read result s from X1 registers

9. The signature is the pair (r, s)

SM2 DSA signature verification steps:

1. Verify that r and s are integers in the interval [1, n-1]. If not, the signature is invalid
   1) Note that n is order, not prime modulus or irreducible polynomial function
2. Calculate e = $HASH_{256}(Z_A \| m)$, where HASH is a cryptographic hashing algorithm, (i.e. SM3 or SHA-256)
   1) Prepare input data include Z_A and message m
   2) Use SM3 or SHA-256 to calculate e

3) Keep the leftmost 256 bits of e

3. Calculate e = e (mod n)
   1) Write the curve order and curve length to N, M registers according to Table 6.40-4
   2) Write the e to X1 register and 0x0 to Y1 register.
   3) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   5) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
   6) Set START(CRYPTO_ECC_CTL[0]) to 1
   7) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   8) Read result e from X1 registers
   9) If e ≥ n, go to step 3
4. Compute t = r + s (mod n)
   1) Write the curve order and curve length to N, M registers according to Table 6.40-4
   2) Write the r to X1 register and s to Y1 register.
   3) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   5) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
   6) Set START(CRYPTO_ECC_CTL[0]) to 1
   7) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   8) Read result t from X1 registers
5. Compute X' (x1', y1') = s * G + t * Q
   1) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4
   2) Write the point G(x, y) to X1, Y1 registers
   3) Write s to K registers
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
   5) Set START(CRYPTO_ECC_CTL[0]) to 1
   6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   7) Read X1, Y1 registers to get s*G
   8) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4
   9) Write the public key Q(x,y) to X1, Y1 registers
   10) Write t to K registers
   11) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
   12) Set START(CRYPTO_ECC_CTL[0]) to 1
   13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   14) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4

15) Write the result data s*G to X2, Y2 registers
16) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 10
17) Set START(CRYPTO_ECC_CTL[0]) to 1
18) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
19) Read X1, Y1 registers to get X'(x1', y1')
20) Write the curve order and curve length to N, M registers according to Table 6.40-4
21) Write x1' to X1 registers
22) Write 0x0 to Y1 registers
23) Set MOPOP(CRYPTO_ECC_CTL[12:11]) to 10
24) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
25) Set START(CRYPTO_ECC_CTL[0]) to 1
26) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
27) Read X1 registers to get x1' (mod n)

6. Compute R = e + x1' (mod n)
   1) Write the curve order and curve length to N, M registers according to Table 6.40-4
   2) Write the x1' to X1 register and e to Y1 register.
   3) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   5) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
   6) Set START(CRYPTO_ECC_CTL[0]) to 1
   7) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   8) Read result R from X1 registers
7. The signature is valid if R = r, otherwise it is invalid.

Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA signature generation steps:

1. Calculate e = HASH(m), where HASH is a cryptographic hashing algorithm, (i.e. SHA-160)
   1) Use SHA to calculate e
   2) Keep the leftmost L_{order} bits of e, where L_{order} is the bit length of order n
2. Select a random integer k form [1, n-1]
   1) Note that n is the order, not prime modulus or irreducible polynomial function.
3. Compute r = x1 (mod n), where (x1, y1) = k * G. If r = 0, go to Step 2.
   1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.40-4
   2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.40-4
   3) Write the point G(x, y) to X1, Y1 registers according to Table 6.40-4

4) Write the random integer k to K register according to Table 6.40-4
5) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
6) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
7) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field
8) Set START(CRYPTO_ECC_CTL[0]) to 1
9) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
10) Write the curve order and curve length to N, M registers according to Table 6.40-4
11) Write 0x0 to Y1 registers
12) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
13) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
14) Set START(CRYPTO_ECC_CTL[0]) to 1
15) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
16) Read X1 registers to get r

4. Compute $s = k^{-1} \times (e + d \times r) \pmod{n}$. If s = 0, go to Step 2
    1) Write the curve order to N registers according to Table 6.40-4
    2) Write the k to X1 register and the d to Y1 register according to Table 6.40-4
    3) Write the r to X2 register and the e to Y2 register according to Table 6.40-4
    4) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length
    5) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
    6) Set ECDSAS(CRYPTO_ECC_CTL[4]) to 1
    7) Set START(CRYPTO_ECC_CTL[0]) to 1
    8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
    9) Read X1 registers to get s
5. The signature is the pair (r, s)

ECDSA signature verification steps:

1. Verify that r and s are integers in the interval [1, n-1]. If not, the signature is invalid
2. Compute e = HASH (m), where HASH is the hashing algorithm in signature generation
    1) Use SHA to calculate e
    2) Keep the leftmost L_{order} bits of e, where L_{order} is the bit length of order n
3. Compute $w = s^{-1} \pmod{n}$
    1) Write the curve order to N registers according to Table 6.40-4
    2) Write s to X1 registers according to Table 6.40-4
    3) Write 0x1 to Y1 registers according to Table 6.40-4
    4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
    5) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 00
    6) Set FSEL(CRYPTO_ECC_CTL[8]) to 1
    7) Set START(CRYPTO_ECC_CTL[0]) to 1

8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
9) Read X1 registers to get w

4. Compute u1 = e × w (mod n) and u2 = r × w (mod n)
   1) Write the curve order and curve length to N, M registers according to Table 6.40-4
   2) Write e, w to X1, Y1 registers
   3) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   4) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 01
   5) Set START(CRYPTO_ECC_CTL[0]) to 1
   6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   7) Read X1 registers to get u1
   8) Write the curve order and curve length to N, M registers according to Table 6.40-4
   9) Write r, w to X1, Y1 registers
   10) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
   11) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 01
   12) Set START(CRYPTO_ECC_CTL[0]) to 1
   13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   14) Read X1 registers to get u2
5. Compute X' (x1', y1') = u1 * G + u2 * Q
   1) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4
   2) Write the point G(x, y) to X1, Y1 registers
   3) Write u1 to K registers
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
   5) Set START(CRYPTO_ECC_CTL[0]) to 1
   6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   7) Read X1, Y1 registers to get u1*G
   8) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4
   9) Write the public key Q(x,y) to X1, Y1 registers
   10) Write u2 to K registers
   11) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00
   12) Set START(CRYPTO_ECC_CTL[0]) to 1
   13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
   14) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4
   15) Write the result data u1*G to X2, Y2 registers
   16) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 10
   17) Set START(CRYPTO_ECC_CTL[0]) to 1

18) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
19) Read X1, Y1 registers to get X'(x1', y1')
20) Write the curve order and curve length to N, M registers according to Table 6.40-4
21) Write x1' to X1 registers
22) Write 0x0 to Y1 registers
23) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01
24) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10
25) Set START(CRYPTO_ECC_CTL[0]) to 1
26) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared
27) Read X1 registers to get x1' (mod n)

6. The signature is valid if x1' = r, otherwise it is invalid.

Elliptic Curve Digital Signature Algorithm (ECDSA) With Key Store

ECDSA signature generation steps:

1. Calculate e = HASH(m), where HASH is a cryptographic hashing algorithm, (i.e. SHA-160)
   1) Use SHA to calculate e
   2) Keep the leftmost L_{order} bits of e, where L_{order} is the bit length of order n
2. Select a random integer k form [1, n-1] via PRNG with TRNG and store in the key store
   1) Note that n is the order, not prime modulus or irreducible polynomial function.
   2) Write the curve order n to ECC N registers for checking PNRG key.
   3) Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0
   4) Configure PRNG key control register CRYPTO_PRNG_KSCTL. Set OWNER to 0x100, WSDST to 0x0, WDST to 0x1 and ECDSA to 0x1 to the user defined value in key store
   5) Configure PRNG control register CRYPTO_PRNG_CTL for KEYSZ to key length, and SEEDSRC to 0x0, SEEDRLD to 0x1, and START to 0x1.
   6) Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (must enable PRNGIEN (CRYPTO_INTEN[16])). Then software can read the NUM (CRYPTO_PRNG_KSSTS[4:0]) of k.
3. Compute r = x1 (mod n), where (x1, y1) = k * G. If r = 0, go to Step 2
   1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.40-4
   2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.40-4.
   3) Write the point G(x, y) to X1 and Y1 registers according to Table 6.40-4 or program the key information of G(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY. (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from the key store.
   4) Write the key information of k to register CRYPTO_ECC_KSCTL and set RSRCK (CRYPTO_ECC_KSCTL[5]) to 1 for updating K register from the key store.
   5) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length.
   6) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00.

7) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field.
8) Set ECDSAR(CRYPTO_ECC_CTL[5]) to 1.
9) Set START(CRYPTO_ECC_CTL[0]) to 1.
10) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
11) Write the curve order and curve length to N, M registers according to Table 6.40-4.
12) Write 0x0 to Y1 registers.
13) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01.
14) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10.
15) Set START(CRYPTO_ECC_CTL[0]) to 1.
16) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
17) Read X1 registers to get r.

4. Compute $s = k^{-1} \times (e + d \times r)(\text{mod } n)$. If s = 0, go to Step 2.
   1) Write the curve order to N registers according to Table 6.40-4.
   2) Write RSSRCY (CRYPTO_ECC_KSXY[15:14]) and NUMY (CRYPTO_ECC_KSXY[12:8]) according to the key information of d.
   3) Write RSRCXY (CRYPTO_ECC_KSXY[5]) to 1.
   4) Write RSSRCX (CRYPTO_ECC_KSXY[7:6]) and NUMX (CRYPTO_ECC_KSXY[4:0]) according to the key information of k.
   5) Write the r to X2 registers and the e to Y2 registers according to Table 6.40-4.
   6) Set CURVEM (CRYPTO_ECC_CTL[31:22]) to key length.
   7) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01.
   8) Set ECDSAS(CRYPTO_ECC_CTL[4]) to 1.
   9) Set START(CRYPTO_ECC_CTL[0]) to 1.
   10) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
   11) Read X1 registers to get s.
5. The signature is the pair (r, s).

ECDSA signature verification steps:

1. Verify that r and s are integers in the interval [1, n-1]. If not, the signature is invalid
2. Compute e = HASH (m), where HASH is the hashing algorithm in signature generation
   1) Use SHA to calculate e.
   2) Keep the leftmost L_{order} bits of e, where L_{order} is the bit length of order n.
3. Compute $w = s^{-1} (\text{mod } n)$
   1) Write the curve order to N registers according to Table 6.40-4.
   2) Write s to X1 registers according to Table 6.40-4.
   3) Write 0x1 to Y1 registers according to Table 6.40-4.
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01.
   5) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 00.
   6) Set FSEL(CRYPTO_ECC_CTL[8]) to 1.

7) Set START(CRYPTO_ECC_CTL[0]) to 1.
8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
9) Read X1 registers to get w.

4. Compute u1 = e × w (mod n) and u2 = r × w (mod n)
   1) Write the curve order and curve length to N, M registers according to Table 6.40-4.
   2) Write e, w to X1, Y1 registers.
   3) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01.
   4) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 01.
   5) Set START(CRYPTO_ECC_CTL[0]) to 1.
   6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
   7) Read X1 registers to get u1.
   8) Write the curve order and curve length to N, M registers according to Table 6.40-4.
   9) Write r, w to X1, Y1 registers.
   10) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01.
   11) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 01.
   12) Set START(CRYPTO_ECC_CTL[0]) to 1.
   13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
   14) Read X1 registers to get u2.
5. Compute X' (x1', y1') = u1 * G + u2 * Q
   1) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4.
   2) Write the point G(x, y) to X1 and Y1 registers according to Table 6.40-4 or program the key information of G(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from the key store.
   3) Write u1 to K registers.
   4) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00.
   5) Set START(CRYPTO_ECC_CTL[0]) to 1.
   6) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
   7) Read X1, Y1 registers to get u1*G
   8) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4.
   9) Write the public key Q(x,y) to X1 and Y1 registers according to Table 6.40-4 or program the key information of Q(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from the key store.
   10) Write u2 to K registers.
   11) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00.
   12) Set START(CRYPTO_ECC_CTL[0]) to 1.
   13) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
   14) Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4.

15) Write the result data u1*G to X2, Y2 registers.
16) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 10.
17) Set START(CRYPTO_ECC_CTL[0]) to 1.
18) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
19) Read X1, Y1 registers to get X'(x1', y1').
20) Write the curve order and curve length to N, M registers according to Table 6.40-4.
21) Write x1' to X1 registers.
22) Write 0x0 to Y1 registers.
23) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 01.
24) Set MODOP(CRYPTO_ECC_CTL[12:11]) to 10.
25) Set START(CRYPTO_ECC_CTL[0]) to 1.
26) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
27) Read X1 registers to get x1' (mod n).

6. The signature is valid if x1' = r, otherwise it is invalid.

Elliptic Curve Diffie-Hellman

Share secret generation function: Z is the x-coordinate of Q where Q = dG (mod N)

1. Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4.
2. Write the public key of receiving party G(x, y) to X1, Y1 registers according to Table 6.40-4.
3. Write (cofactor h * my private key d) to K register according to Table 6.40-4.
   1) h=1 in P-192, P-224, P-256, P-384 and P-521
   2) h=2 in B-163, B-233, B-283, B-409, B-571 and K-163
   3) h=4 in K-233, K-283, K-409 and K-571
4. Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00.
5. Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field.
6. Set START(CRYPTO_ECC_CTL[0]) to 1.
7. Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
8. Read public key Q from X1, Y1 registers.

Hash-based key derivation function: DerivedKeyingMaterial = KDF(Z, OtherInput)

Step1 For i = 1 to reps, where reps = ceil((the length of Z, OtherInput)/hash length)

1. Write the curve parameter A, B, N, and curve length M to corresponding registers according to Table 6.40-4.
2. Write the public key of receiving party G(x, y) to X1, Y1 registers according to Table 6.40-4.

Step 2 Return DerivedKeyingMaterial = $Hash_1$ || $Hash_2$ || … || $Hash_{reps}$

Note:

1. For the details of OtherInput, please refer to the page 46 in NIST SP 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”.
2. For the details of cofactor, please refer to “RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE”.

Elliptic Curve Diffie-Hellman With Key Store

Exchange a share key via RNG, ECC and key store.

1. Generate private key d_A via PRNG with TRNG to key store.
    1) Write the curve order n to ECC N registers for checking PNRG key.
    2) Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
    3) Configure PRNG key control register CRYPTO_PRNG_KSCTL. Set OWNER to 0x100, WSDST to 0x0, WDST to 0x1 and ECDH to 0x1 to the user defined value in the key store.
    4) Configure PRNG control register CRYPTO_PRNG_CTL for KEYSZ to key length, and SEEDSRC to 0x0, SEEDRLD to 0x1, and START to 0x1.
    5) Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF (CRYPTO_INTSTS[16]) (must enable PRNGIEN (CRYPTO_INTEN[16])). Then software can read the NUMBER (CRYPTO_PRNG_KSSTS[4:0]) of d_A.
2. Generate public key Q_A according to d_A via ECC and key store
    1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.40-4.
    2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.40-4.
    3) Write the public point G(x, y) to X1, Y1 registers according to Table 6.40-4 or program the key information of G(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY (CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from the key store.
    4) Program the key information of dA to register CRYPTO_ECC_KSCTL. Set the RSRCK to 0x1 and the NUMK to the NUM (CRYPTO_PRNG_KSSTS[4:0]) of dA. Set the RSSRCK to the WSDST (CRYPTO_PRNG_KSCTL[23:22]). Set others bits to 0x0.
    5) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00.
    6) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field.
    7) Set START(CRYPTO_ECC_CTL[0]) to 1.
    8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
    9) Read X1 and Y1 registers to get public key Q_A.
3. Send our public key Q_A to the opposite side and receive their public key Q_B.
4. Generate the share key according to d_A and Q_B via ECC and key store.
    1) Write the curve parameter A, B, N and curve length M to corresponding registers according to Table 6.40-4.
    2) Write the prime modulus or irreducible polynomial function to N registers according to Table 6.40-4.
    3) Write the public key QB(x, y) to X1, Y1 registers according to Table 6.40-4 or program the key information of QB(x, y) to register CRYPTO_ECC_KSXY and set RSRCXY

(CRYPTO_ECC_KSXY[5]) to 1 for updating X1, Y1 registers from the key store.

4) Program ECC key information to register CRYPTO_ECC_KSCTL. Set the RSRCK to 0x1 and the NUMK to the NUMBER (CRYPTO_PRNG_KSSTS[4:0]) of dA. Set the RSSRCK to the WSDST (CRYPTO_PRNG_KSCTL[23:22]). Set RSRCXY to 0x0. Set ECDH to 0x1. Set XY, WDST, WSDST and OWNER to the user defined value.
5) Set ECCOP(CRYPTO_ECC_CTL[10:9]) to 00.
6) Set FSEL(CRYPTO_ECC_CTL[8]) according to used curve of prime field or binary field.
7) Set START(CRYPTO_ECC_CTL[0]) to 1.
8) Wait for BUSY(CRYPTO_ECC_STS[0]) to be cleared.
9) Read NUM(CRPT_ECC_KSSTS[4:0]) to get the number of share key in the key store.

Note:

5. To generate the share key with key store support safely, it is suggested that user should make a new random number instead of the adopted private key when executing ECDH with key store every time.
6. When setting ECDH (CRYPTO_PRNG_KSCTL[19]) to generate a new d_A of ECDH flow by PRNG, the maximum read times of d_A key is 2 in key store. The first time is for Step 2 and the second one is for Step 4. Thus, ECC will output the result of point multiplication in X1 and Y1 register or the DMA destination address only when the d_A key is read for the first time. When the d_A key is read second time, the only destination of point multiplication is key store.
7. It is suggested that user can erase the d_A key in key store after generating the share key.

6.40.5.6 RSA (Rivest, Shamir and Adleman) Cryptography With CRT (Chinese Remainder Theorem) and SCA (Side-Channel Attack)

RSA is the first asymmetric cryptosystem (public-key cryptosystems) designed and named by Ron Rivest, Adi Shamir and Leonard Adleman from MIT in 1977. In the past years, many protocols and applications utilize the complexity of the prime factoring problem in the large semi-prime number to build cryptographic systems. The main computation of encryption and description in RSA is modulus exponentiation operation with very large bit length (as least 2048 bits from NIST's suggestion). To accelerate RSA applications, an RSA accelerator is provided, which can quickly compute the complex modulus exponentiation operation.

Chinese Remainder Theorem (CRT) is a theorem of number theory, which is widely used for computing with large integers. As it allows replacing a computation for which one knows a bound on the size of the result by several similar computations on small integers. If users want to repeatedly execute decryption with the same private key, they can switch normal RSA decryption mode to Chinese Remainder Theorem mode to improve performance.

Side-channel attack (SCA) is any attack based on information gained from the implementation of a computer system, rather than weaknesses in the implemented algorithm itself. Then, power consumption analysis is a method of side-channel attack most often used. If users want to against simple power analysis (SPA) and differential power analysis (DPA), they can use the side channel attack protection.

Before starting RSA accelerator with non-CRT operation, users must provide the required input data of modulus exponentiation operation include the base of exponentiation (abbreviated to M) in the memory address SADDR0, the base of modulus operation (abbreviated to N) in the memory address SADDR1 and the exponent of exponentiation (abbreviated to E) in the memory address SADDR2. Moreover, user still reserve a memory address DADDR for storing the Intermediate temporary value from RSA accelerator. After finishing non-CRT operation, RSA will store the result of modulus exponentiation operation in the memory address DADDR.

Before starting RSA accelerator with CRT operation, users must provide the required input data of modulus exponentiation operation include the base of exponentiation (abbreviated to M) in the memory address SADDR0, the base of modulus operation (abbreviated to N) in the memory address SADDR1 and the exponent of exponentiation (abbreviated to E) in the memory address SADDR2, and the factors of modulus operation (abbreviated to p and abbreviated to q) in memory address SADDR3 and SADDR4. Moreover, user still reserve seven memory address MADDR0~MADDR5 and DADDR for storing these intermediate temporary value from RSA accelerator. After finishing CRT operation, RSA will store the result of modulus exponentiation operation in the memory address DADDR. If the RSA accelerator executes CRT operation with key from the key store, key number of these intermediate temporary values (MADDR0~MADDR5) are stored in Key Number2~7 of CRYPTO_RSA_KSSTS0 and CRYPTO_RSA_KSSTS1.

However, if users want to execute side channel attack(SCA) protection and non-CRT mode, users must provide the required input data of modulus exponentiation operation include the base of exponentiation (abbreviated to M) in the memory address SADDR0, the base of modulus operation (abbreviated to N) in the memory address SADDR1 and the exponent of exponentiation (abbreviated to E) in the memory address SADDR2, and the factors of modulus operation (abbreviated to p and abbreviated to q) in memory address SADDR3 and SADDR4. Moreover, users still reserve two memory address DADDR and MADDR6 for storing these intermediate temporary value from RSA accelerator. After finishing non-CRT operation, RSA will store the result of modulus exponentiation operation in the memory address DADDR.

If users want to decrypt repeatedly with the same key, they can execute CRT bypass mode(CRTBYP) after the first time CRT decryption, they only need to provide the new required input data M in the memory address SADDR0. But users cannot change the data in the memory address SADDR1~SADDR4 (key, N, Key Number0~1 in CRYPTO_RSA_KSSTS0) and MADDR2~MADDR5 (Key Number4~7 in CRYPTO_RSA_KSSTS1). If key and data are stored in key store, users also cannot erase or revoke them. Otherwise, CRT bypass mode will be failed. moreover, they cannot execute CRTBYP in non-CRT mode.

RSA input data definition: M<N, input data (the base of exponentiation in the memory address SADDR0, abbreviated to M) must less than modulus (the base of modulus operation in the memory address SADDR1, abbreviated to N).

Users need to make sure that the memory space from SADDR0~SADDR2, MADDR0~MADDR1(Key Number2~3 in CRYPTO_RSA_KSSTS0), MADDR4~MADDR5(Key Number6~7 in CRYPTO_RSA_KSSTS1) memory address to its address +key length are reserved for RSA accelerator. The memory space from SADDR3~SADDR4(Key Number0~1 in CRYPTO_RSA_KSSTS0), MADDR2~MADDR3 (Key Number4~5 in CRYPTO_RSA_KSSTS1) memory address to its address +half key length are reserved for RSA accelerator. If executing side channel attack(SCA) protection in non-CRT mode, users should make sure that the memory space from MADDR6 to its address +key length+128-bits are reserved for RSA accelerator. (**Note:** The data endianness is little-endian. If the data bit-width is smaller than the keeping memory space, users should write zero to not enough part. For example, if key length is 8-bits and the data just 4-bits(1011), users should write 0000_1011 to the memory.)

After starting RSA accelerator operation, users cannot change input data in the memory address SADDR0~ SADDR4, MADDR0~MADDR6 and DADDR until the operation is finishing. Otherwise, users will get wrong result of modulus exponentiation operation.

Users need to allocate the space which size is 1 × key length (for p, q) in SRAM of key store in advance before executing the SCA protection mode with key store. If users want to execute SCA protection and no-CRT mode, users need to allocate another space which size is 1 × key length (for blind private key) in SRAM of key store, and it should be set un-readable. (If users want to execute RSA with key store, users need to operate these register: CRYPTO_RSA_KSCTL, CRYPTO_RSA_KSSTS0 and CRYPTO_RSA_KSSTS1)

Users need to allocate the space which size is 6 × key length in SRAM of key store in advance before executing the CRT mode with key store (middle data only read from / write to the SRAM of key store).

If users want to execute a new encryption or decryption with RSA accelerator, they must to rewrite input data (the base of exponentiation in the memory address SADDR0, abbreviated to M). Otherwise, they may get wrong input message and wrong answer.

To bring the side-channel protection of RSA into full play, it is suggested that user must execute “PRNG get the random number with the true random number seed from TRNG” at least once before starting the first RSA operation after every CRYPTO or system reset.

The RSA accelerator will execute correctly in the support situation. The support situation as shown in Table 6.40-6. If users use the error combination, CTLERR(CRYPTO_RSA_STS[17]) will be set to 1 (even user unset the START(CRYPTO_RSA_CTL[0])), and RSA accelerator will not start in the unsupported situation.

CRT: Chinese remainder theorem; CRT_BYP: CRT bypass; SCAP: side channel attack protection; xxx = { SCAP, CRT_BYP, CRT }, 1 = enable, 0 = disable. (‘✓’means RSA available, ‘×’ means RSA unavailable and CTLERR(CRYPTO_RSA_STS[17]) will become ‘1’)

KEY LENGTH	Normal (Xxx=000)	CRT (Xxx=001)	CRT_BYP (Xxx=010)	SCAP (Xxx=100)	CRT + CRT_BYP (Xxx=011)	CRT + SCAP (Xxx=101)	CRT_BYP + SCA (Xxx=110)	CRT + SCAP + CRT_BYP (Xxx=111)
1024	✓	×	×	✓	×	×	×	×
2048	✓	✓	×	✓	✓	✓	×	✓
3072	✓	✓	×	✓	✓	✓	×	✓
4096	✓	✓	×	✓	✓	✓	×	✓

Table 6.40-6 RSA Accelerator Support Situation

RSA DMA Mode Programming Flow

1. Write 1 to RSAIEN(CRYPTO_INTEN [30]) to enable RSA interrupt if needed.
2. Write input data to DMA source address.
3. Program DMA source address to register CRYPTO_RSA_SADDR0~ SADDR4.
4. If executing SCA protection and no-CRT mode, program DMA middle address to register CRYPTO_RSA_MADDR6.
5. Program DMA destination address to register CRYPTO_RSA_DADDR.
6. Configure RSA control register CRYPTO_RSA_CTL for RSA accelerator, such as the start signal of RSA accelerator(START), the stop signal of RSA accelerator(STOP), CRT enable control(CRT), CRT bypass enable control (CRTBYP), side channel attack protection enable control (SCAP), and the key length of RSA(KEYLENG).
7. Wait for the RSA interrupt flag RSAIF(CRYPTO_INTSTS[30]) to be set.
8. Read output data and then clear RSA interrupt flag RSAIF.

RSA CRT DMA Mode Programming Flow

1. Write 1 to RSAIEN(CRYPTO_INTEN [30]) to enable RSA interrupt if needed.
2. Write input data to DMA source address.
3. Program DMA source address to register CRYPTO_RSA_SADDR0~ SADDR4.
4. Program DMA middle address to register CRYPTO_RSA_MADDR0~ MADDR5.

5. Program DMA destination address to register CRYPTO_RSA_DADDR.

6. Configure RSA control register CRYPTO_RSA_CTL for RSA accelerator, such as the start signal of RSA accelerator(START), the stop signal of RSA accelerator(STOP), CRT enable control(CRT), CRT bypass enable control (CRTBYP), side channel attack protection enable control (SCAP), and the key length of RSA(KEYLENG).

7. Wait for the RSA interrupt flag RSAIF(CRYPTO_INTSTS[30]) to be set.

8. Read output data and then clear RSA interrupt flag RSAIF.

RSA DMA Mode with Key Store Programming Flow

1. Write 1 to RSAIEN(CRYPTO_INTEN [30]) to enable RSA interrupt if needed.

2. Write message, N and public key to DMA, and write others data (private key, p, q) to key store.

3. If executing SCA protection and no-CRT mode, allocate the space SRAM of key store for exponent blind key in advance before executing the SCA protection and no-CRT mode with key store. Its OWNER (KS_METADATA[18:16]) bits should be set to '010' and it should also be set un-readable (KS_METADATA[2] is set to '0').

4. Configure RSA key control register CRYPTO_RSA_KSCTL for Key Store Source (RSSRC), key Source (RSRC), exponent blind key number of key (BKNUM) and key number of key (NUM) in the key store.

5. Program DMA source address to register CRYPTO_RSA_SADDR0~ SADDR2.

6. Configure RSA key status register CRYPTO_RSA_KSSTS0 for each key number of others data (NUM0~1) in the key store.

7. Program DMA destination address to register CRYPTO_RSA_DADDR.

8. Configure RSA control register CRYPTO_RSA_CTL for RSA accelerator, such as the start signal of RSA accelerator(START), the stop signal of RSA accelerator(STOP), CRT enable control(CRT), CRT bypass enable control (CRTBYP), side channel attack protection enable control (SCAP), and the key length of RSA(KEYLENG).

9. Wait for the RSA interrupt flag RSAIF(CRYPTO_INTSTS[30]) to be set.

10. Read output data and then clear RSA interrupt flag RSAIF.

RSA CRT DMA Mode with Key Store Programming Flow

1. Write 1 to RSAIEN(CRYPTO_INTEN [30]) to enable RSA interrupt if needed.

2. Write message, N and public key to DMA, and write others data (private key, p, q) to key store.

3. Allocate the space SRAM of key store for Key Number2~7 (in CRYPTO_RSA_KSSTS0 and CRYPTO_RSA_KSSTS1) in advance before executing the CRT mode with key store (middle data only read from / write to the SRAM of key store).

4. Configure RSA key control register CRYPTO_RSA_KSCTL for key store Source (RSSRC), key Source (RSRC) and key number of key (NUM) in key store.

5. Program DMA source address to register CRYPTO_RSA_SADDR0~ SADDR2.

6. Configure RSA key status register CRYPTO_RSA_KSSTS0, CRYPTO_RSA_KSSTS1 for each key number of others data (Key Number0~7) in key store.

7. Program DMA destination address to register CRYPTO_RSA_DADDR.

8. Configure RSA control register CRYPTO_RSA_CTL for RSA accelerator, such as the start signal of RSA accelerator(START), the stop signal of RSA accelerator(STOP), CRT enable control(CRT), CRT bypass enable control (CRTBYP), side channel attack protection enable control (SCAP), and the key length of RSA(KEYLENG).

9. Wait for the RSA interrupt flag RSAIF(CRYPTO_INTSTS[30]) to be set.

10. Read output data and then clear RSA interrupt flag RSAIF.

Some Notices of RSA Accelerator

1. The key length support of RSA accelerator for both non-CRT encryption and decryption is 1024, 2048, 3072 and 4096 bits. The key length support of RSA accelerator for CRT decryption is 2048, 3072 and 4096 bits.

2. All input and output data must be positive.

3. When RSA engine is active (i.e., BUSY is 1 and DMABUSY (CRYPTO_RSA_STS[1]) is 0), user cannot modify any input and output data registers.

4. If user wants to stop RSA accelerator, please configures the STOP (CRYPTO_RSA_CTL[1]) to 1. **Note:** To avoid the transmission error of the next operation, BUSY signal will not be cleared immediately until the action of DMA is done.

5. User must rewrite input data (the base of exponentiation in the memory address SADDR0, abbreviated to M) in every encryption or decryption.

6. If user wants to execute CRT bypass mode (CRTBYP) after the first time CRT decryption mode, they cannot change the data in the memory address SADDR1~SADDR4(key, N, Key Number0~1 in CRYPTO_RSA_KSSTS0) and MADDR2~MADDR5(Key Number4~7 in CRYPTO_RSA_KSSTS1). If key and data are stored in key store, user also cannot erase or revoke them. Otherwise, CRT bypass mode will be failed.

7. To bring the side-channel protection of RSA into full play, it is suggested that user must execute “PRNG get the random number with the true random number seed from TRNG” at least once before starting the first RSA operation after every CRYPTO or system reset.

6.40.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
CRYPTO Base Address: **CRYPTO_BA = 0x4003_2000**				
CRYPTO_INTEN	CRYPTO_BA+0x000	R/W	Crypto Interrupt Enable Control Register	0x0000_0000
CRYPTO_INTSTS	CRYPTO_BA+0x004	R/W	Crypto Interrupt Flag	0x0000_0000
CRYPTO_PRNG_CTL	CRYPTO_BA+0x008	R/W	PRNG Control Register	0x0000_0000
CRYPTO_PRNG_SEED	CRYPTO_BA+0x00C	W	Seed for PRNG	Undefined
CRYPTO_PRNG_KEY0	CRYPTO_BA+0x010	R	PRNG Generated Key0	Undefined
CRYPTO_PRNG_KEY1	CRYPTO_BA+0x014	R	PRNG Generated Key1	Undefined
CRYPTO_PRNG_KEY2	CRYPTO_BA+0x018	R	PRNG Generated Key2	Undefined
CRYPTO_PRNG_KEY3	CRYPTO_BA+0x01C	R	PRNG Generated Key3	Undefined
CRYPTO_PRNG_KEY4	CRYPTO_BA+0x020	R	PRNG Generated Key4	Undefined
CRYPTO_PRNG_KEY5	CRYPTO_BA+0x024	R	PRNG Generated Key5	Undefined
CRYPTO_PRNG_KEY6	CRYPTO_BA+0x028	R	PRNG Generated Key6	Undefined
CRYPTO_PRNG_KEY7	CRYPTO_BA+0x02C	R	PRNG Generated Key7	Undefined
CRYPTO_PRNG_STS	CRYPTO_BA+0x030	R	PRNG Status Register	0x0000_0000
CRYPTO_AES_FDBCK0	CRYPTO_BA+0x050	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK1	CRYPTO_BA+0x054	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK2	CRYPTO_BA+0x058	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK3	CRYPTO_BA+0x05C	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_GCM_IVCNT0	CRYPTO_BA+0x080	R/W	AES GCM IV Byte Count Register 0	0x0000_0000
CRYPTO_AES_GCM_IVCNT1	CRYPTO_BA+0x084	R/W	AES GCM IV Byte Count Register 1	0x0000_0000
CRYPTO_AES_GCM_ACNT0	CRYPTO_BA+0x088	R/W	AES GCM A Byte Count Register 0	0x0000_0000
CRYPTO_AES_GCM_ACNT1	CRYPTO_BA+0x08C	R/W	AES GCM A Byte Count Register 1	0x0000_0000
CRYPTO_AES_GCM_PCNT0	CRYPTO_BA+0x090	R/W	AES GCM P Byte Count Register 0	0x0000_0000
CRYPTO_AES_GCM_PCNT1	CRYPTO_BA+0x094	R/W	AES GCM P Byte Count Register 1	0x0000_0000
CRYPTO_AES_FBADDR	CRYPTO_BA+0x0A0	R/W	AES DMA Feedback Address Register	0x0000_0000
CRYPTO_AES_CTL	CRYPTO_BA+0x100	R/W	AES Control Register	0x0000_0000
CRYPTO_AES_STS	CRYPTO_BA+0x104	R	AES Engine Flag	0x0001_0100
CRYPTO_AES_DATIN	CRYPTO_BA+0x108	R/W	AES Engine Data Input Port Register	0x0000_0000
CRYPTO_AES_DATOUT	CRYPTO_BA+0x10C	R	AES Engine Data Output Port Register	0x0000_0000
CRYPTO_AES_KEY0	CRYPTO_BA+0x110	R/W	AES Key Word 0 Register	0x0000_0000

CRYPTO_AES_KEY1	CRYPTO_BA+0x114	R/W	AES Key Word 1 Register	0x0000_0000
CRYPTO_AES_KEY2	CRYPTO_BA+0x118	R/W	AES Key Word 2 Register	0x0000_0000
CRYPTO_AES_KEY3	CRYPTO_BA+0x11C	R/W	AES Key Word 3 Register	0x0000_0000
CRYPTO_AES_KEY4	CRYPTO_BA+0x120	R/W	AES Key Word 4 Register	0x0000_0000
CRYPTO_AES_KEY5	CRYPTO_BA+0x124	R/W	AES Key Word 5 Register	0x0000_0000
CRYPTO_AES_KEY6	CRYPTO_BA+0x128	R/W	AES Key Word 6 Register	0x0000_0000
CRYPTO_AES_KEY7	CRYPTO_BA+0x12C	R/W	AES Key Word 7 Register	0x0000_0000
CRYPTO_AES_IV0	CRYPTO_BA+0x130	R/W	AES Initial Vector Word 0 Register	0x0000_0000
CRYPTO_AES_IV1	CRYPTO_BA+0x134	R/W	AES Initial Vector Word 1 Register	0x0000_0000
CRYPTO_AES_IV2	CRYPTO_BA+0x138	R/W	AES Initial Vector Word 2 Register	0x0000_0000
CRYPTO_AES_IV3	CRYPTO_BA+0x13C	R/W	AES Initial Vector Word 3 Register	0x0000_0000
CRYPTO_AES_SADDR	CRYPTO_BA+0x140	R/W	AES DMA Source Address Register	0x0000_0000
CRYPTO_AES_DADDR	CRYPTO_BA+0x144	R/W	AES DMA Destination Address Register	0x0000_0000
CRYPTO_AES_CNT	CRYPTO_BA+0x148	R/W	AES Byte Count Register	0x0000_0000
CRYPTO_HMAC_CTL	CRYPTO_BA+0x300	R/W	SHA/HMAC Control Register	0x0000_0000
CRYPTO_HMAC_STS	CRYPTO_BA+0x304	R	SHA/HMAC Status Flag	0x0000_0000
CRYPTO_HMAC_DGST0	CRYPTO_BA+0x308	R	SHA/HMAC Output Feedback Data 0	0x0000_0000
CRYPTO_HMAC_DGST1	CRYPTO_BA+0x30C	R	SHA/HMAC Output Feedback Data 1	0x0000_0000
CRYPTO_HMAC_DGST2	CRYPTO_BA+0x310	R	SHA/HMAC Output Feedback Data 2	0x0000_0000
CRYPTO_HMAC_DGST3	CRYPTO_BA+0x314	R	SHA/HMAC Output Feedback Data 3	0x0000_0000
CRYPTO_HMAC_DGST4	CRYPTO_BA+0x318	R	SHA/HMAC Output Feedback Data 4	0x0000_0000
CRYPTO_HMAC_DGST5	CRYPTO_BA+0x31C	R	SHA/HMAC Output Feedback Data 5	0x0000_0000
CRYPTO_HMAC_DGST6	CRYPTO_BA+0x320	R	SHA/HMAC Output Feedback Data 6	0x0000_0000
CRYPTO_HMAC_DGST7	CRYPTO_BA+0x324	R	SHA/HMAC Output Feedback Data 7	0x0000_0000
CRYPTO_HMAC_DGST8	CRYPTO_BA+0x328	R	SHA/HMAC Output Feedback Data 8	0x0000_0000
CRYPTO_HMAC_DGST9	CRYPTO_BA+0x32C	R	SHA/HMAC Output Feedback Data 9	0x0000_0000
CRYPTO_HMAC_DGST10	CRYPTO_BA+0x330	R	SHA/HMAC Output Feedback Data 10	0x0000_0000
CRYPTO_HMAC_DGST11	CRYPTO_BA+0x334	R	SHA/HMAC Output Feedback Data 11	0x0000_0000
CRYPTO_HMAC_DGST12	CRYPTO_BA+0x338	R	SHA/HMAC Output Feedback Data 12	0x0000_0000
CRYPTO_HMAC_DGST13	CRYPTO_BA+0x33C	R	SHA/HMAC Output Feedback Data 13	0x0000_0000
CRYPTO_HMAC_DGST14	CRYPTO_BA+0x340	R	SHA/HMAC Output Feedback Data 14	0x0000_0000
CRYPTO_HMAC_DGST15	CRYPTO_BA+0x344	R	SHA/HMAC Output Feedback Data 15	0x0000_0000
CRYPTO_HMAC_KEYCNT	CRYPTO_BA+0x348	R/W	SHA/HMAC Key Byte Count Register	0x0000_0000
CRYPTO_HMAC_SADDR	CRYPTO_BA+0x34C	R/W	SHA/HMAC DMA Source Address Register	0x0000_0000
CRYPTO_HMAC_DMACNT	CRYPTO_BA+0x350	R/W	SHA/HMAC Byte Count Register	0x0000_0000

CRYPTO_HMAC_DATIN	CRYPTO_BA+0x354	R/W	SHA/HMAC Engine Non-DMA Mode Data Input Port Register	0x0000_0000
CRYPTO_HMAC_FDBCK0	CRYPTO_BA+0x358	R/W	SHA/HMAC Output Feedback Data 0 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK1	CRYPTO_BA+0x35C	R/W	SHA/HMAC Output Feedback Data 1 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK2	CRYPTO_BA+0x360	R/W	SHA/HMAC Output Feedback Data 2 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK3	CRYPTO_BA+0x364	R/W	SHA/HMAC Output Feedback Data 3 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK4	CRYPTO_BA+0x368	R/W	SHA/HMAC Output Feedback Data 4 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK5	CRYPTO_BA+0x36C	R/W	SHA/HMAC Output Feedback Data 5 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK6	CRYPTO_BA+0x370	R/W	SHA/HMAC Output Feedback Data 6 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK7	CRYPTO_BA+0x374	R/W	SHA/HMAC Output Feedback Data 7 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK8	CRYPTO_BA+0x378	R/W	SHA/HMAC Output Feedback Data 8 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK9	CRYPTO_BA+0x37C	R/W	SHA/HMAC Output Feedback Data 9 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK10	CRYPTO_BA+0x380	R/W	SHA/HMAC Output Feedback Data 10 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK11	CRYPTO_BA+0x384	R/W	SHA/HMAC Output Feedback Data 11 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK12	CRYPTO_BA+0x388	R/W	SHA/HMAC Output Feedback Data 12 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK13	CRYPTO_BA+0x38C	R/W	SHA/HMAC Output Feedback Data 13 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK14	CRYPTO_BA+0x390	R/W	SHA/HMAC Output Feedback Data 14 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK15	CRYPTO_BA+0x394	R/W	SHA/HMAC Output Feedback Data 15 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK16	CRYPTO_BA+0x398	R/W	SHA/HMAC Output Feedback Data 16 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK17	CRYPTO_BA+0x39C	R/W	SHA/HMAC Output Feedback Data 17 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK18	CRYPTO_BA+0x3A0	R/W	SHA/HMAC Output Feedback Data 18 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK19	CRYPTO_BA+0x3A4	R/W	SHA/HMAC Output Feedback Data 19 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK20	CRYPTO_BA+0x3A8	R/W	SHA/HMAC Output Feedback Data 20 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK21	CRYPTO_BA+0x3AC	R/W	SHA/HMAC Output Feedback Data 21 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK22	CRYPTO_BA+0x3B0	R/W	SHA/HMAC Output Feedback Data 22 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK23	CRYPTO_BA+0x3B4	R/W	SHA/HMAC Output Feedback Data 23 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK24	CRYPTO_BA+0x3B8	R/W	SHA/HMAC Output Feedback Data 24 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK25	CRYPTO_BA+0x3BC	R/W	SHA/HMAC Output Feedback Data 25 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK26	CRYPTO_BA+0x3C0	R/W	SHA/HMAC Output Feedback Data 26 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK27	CRYPTO_BA+0x3C4	R/W	SHA/HMAC Output Feedback Data 27 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK28	CRYPTO_BA+0x3C8	R/W	SHA/HMAC Output Feedback Data 28 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK29	CRYPTO_BA+0x3CC	R/W	SHA/HMAC Output Feedback Data 29 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK30	CRYPTO_BA+0x3D0	R/W	SHA/HMAC Output Feedback Data 30 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK31	CRYPTO_BA+0x3D4	R/W	SHA/HMAC Output Feedback Data 31 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK32	CRYPTO_BA+0x3D8	R/W	SHA/HMAC Output Feedback Data 32 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK33	CRYPTO_BA+0x3DC	R/W	SHA/HMAC Output Feedback Data 33 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK34	CRYPTO_BA+0x3E0	R/W	SHA/HMAC Output Feedback Data 34 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK35	CRYPTO_BA+0x3E4	R/W	SHA/HMAC Output Feedback Data 35 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK36	CRYPTO_BA+0x3E8	R/W	SHA/HMAC Output Feedback Data 36 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK37	CRYPTO_BA+0x3EC	R/W	SHA/HMAC Output Feedback Data 37 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK38	CRYPTO_BA+0x3F0	R/W	SHA/HMAC Output Feedback Data 38 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK39	CRYPTO_BA+0x3F4	R/W	SHA/HMAC Output Feedback Data 39 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK40	CRYPTO_BA+0x3F8	R/W	SHA/HMAC Output Feedback Data 40 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK41	CRYPTO_BA+0x3FC	R/W	SHA/HMAC Output Feedback Data 41 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK42	CRYPTO_BA+0x400	R/W	SHA/HMAC Output Feedback Data 42 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK43	CRYPTO_BA+0x404	R/W	SHA/HMAC Output Feedback Data 43 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK44	CRYPTO_BA+0x408	R/W	SHA/HMAC Output Feedback Data 44 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK45	CRYPTO_BA+0x40C	R/W	SHA/HMAC Output Feedback Data 45 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK46	CRYPTO_BA+0x410	R/W	SHA/HMAC Output Feedback Data 46 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK47	CRYPTO_BA+0x414	R/W	SHA/HMAC Output Feedback Data 47 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK48	CRYPTO_BA+0x418	R/W	SHA/HMAC Output Feedback Data 48 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK49	CRYPTO_BA+0x41C	R/W	SHA/HMAC Output Feedback Data 49 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK50	CRYPTO_BA+0x420	R/W	SHA/HMAC Output Feedback Data 50 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK51	CRYPTO_BA+0x424	R/W	SHA/HMAC Output Feedback Data 51 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK52	CRYPTO_BA+0x428	R/W	SHA/HMAC Output Feedback Data 52 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK53	CRYPTO_BA+0x42C	R/W	SHA/HMAC Output Feedback Data 53 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK54	CRYPTO_BA+0x430	R/W	SHA/HMAC Output Feedback Data 54 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK55	CRYPTO_BA+0x434	R/W	SHA/HMAC Output Feedback Data 55 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK56	CRYPTO_BA+0x438	R/W	SHA/HMAC Output Feedback Data 56 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK57	CRYPTO_BA+0x43C	R/W	SHA/HMAC Output Feedback Data 57 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK58	CRYPTO_BA+0x440	R/W	SHA/HMAC Output Feedback Data 58 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK59	CRYPTO_BA+0x444	R/W	SHA/HMAC Output Feedback Data 59 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK60	CRYPTO_BA+0x448	R/W	SHA/HMAC Output Feedback Data 60 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK61	CRYPTO_BA+0x44C	R/W	SHA/HMAC Output Feedback Data 61 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK62	CRYPTO_BA+0x450	R/W	SHA/HMAC Output Feedback Data 62 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK63	CRYPTO_BA+0x454	R/W	SHA/HMAC Output Feedback Data 63 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK64	CRYPTO_BA+0x458	R/W	SHA/HMAC Output Feedback Data 64 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK65	CRYPTO_BA+0x45C	R/W	SHA/HMAC Output Feedback Data 65 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK66	CRYPTO_BA+0x460	R/W	SHA/HMAC Output Feedback Data 66 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK67	CRYPTO_BA+0x464	R/W	SHA/HMAC Output Feedback Data 67 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK68	CRYPTO_BA+0x468	R/W	SHA/HMAC Output Feedback Data 68 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK69	CRYPTO_BA+0x46C	R/W	SHA/HMAC Output Feedback Data 69 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK70	CRYPTO_BA+0x470	R/W	SHA/HMAC Output Feedback Data 70 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK71	CRYPTO_BA+0x474	R/W	SHA/HMAC Output Feedback Data 71 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK72	CRYPTO_BA+0x478	R/W	SHA/HMAC Output Feedback Data 72 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK73	CRYPTO_BA+0x47C	R/W	SHA/HMAC Output Feedback Data 73 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK74	CRYPTO_BA+0x480	R/W	SHA/HMAC Output Feedback Data 74 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK75	CRYPTO_BA+0x484	R/W	SHA/HMAC Output Feedback Data 75 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK76	CRYPTO_BA+0x488	R/W	SHA/HMAC Output Feedback Data 76 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK77	CRYPTO_BA+0x48C	R/W	SHA/HMAC Output Feedback Data 77 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK78	CRYPTO_BA+0x490	R/W	SHA/HMAC Output Feedback Data 78 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK79	CRYPTO_BA+0x494	R/W	SHA/HMAC Output Feedback Data 79 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK80	CRYPTO_BA+0x498	R/W	SHA/HMAC Output Feedback Data 80 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK81	CRYPTO_BA+0x49C	R/W	SHA/HMAC Output Feedback Data 81 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK82	CRYPTO_BA+0x4A0	R/W	SHA/HMAC Output Feedback Data 82 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK83	CRYPTO_BA+0x4A4	R/W	SHA/HMAC Output Feedback Data 83 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK84	CRYPTO_BA+0x4A8	R/W	SHA/HMAC Output Feedback Data 84 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK85	CRYPTO_BA+0x4AC	R/W	SHA/HMAC Output Feedback Data 85 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK86	CRYPTO_BA+0x4B0	R/W	SHA/HMAC Output Feedback Data 86 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK87	CRYPTO_BA+0x4B4	R/W	SHA/HMAC Output Feedback Data 87 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_SHA512T	CRYPTO_BA+0x4F8	R/W	SHA/HMAC SHA512/t Control Register	0x0000_0000
CRYPTO_HMAC_FBADDR	CRYPTO_BA+0x4FC	R/W	SHA/HMAC DMA Feedback Address Register	0x0000_0000
CRYPTO_HMAC_SHAKEDGST0	CRYPTO_BA+0x500	R	SHA/HMAC SHAKE Digest Message 0	0x0000_0000
CRYPTO_HMAC_SHAKEDGST1	CRYPTO_BA+0x504	R	SHA/HMAC SHAKE Digest Message 1	0x0000_0000
CRYPTO_HMAC_SHAKEDGST2	CRYPTO_BA+0x508	R	SHA/HMAC SHAKE Digest Message 2	0x0000_0000

CRYPTO_HMAC_SHAKEDGST3	CRYPTO_BA+0x50C	R	SHA/HMAC SHAKE Digest Message 3	0x0000_0000
CRYPTO_HMAC_SHAKEDGST4	CRYPTO_BA+0x510	R	SHA/HMAC SHAKE Digest Message 4	0x0000_0000
CRYPTO_HMAC_SHAKEDGST5	CRYPTO_BA+0x514	R	SHA/HMAC SHAKE Digest Message 5	0x0000_0000
CRYPTO_HMAC_SHAKEDGST6	CRYPTO_BA+0x518	R	SHA/HMAC SHAKE Digest Message 6	0x0000_0000
CRYPTO_HMAC_SHAKEDGST7	CRYPTO_BA+0x51C	R	SHA/HMAC SHAKE Digest Message 7	0x0000_0000
CRYPTO_HMAC_SHAKEDGST8	CRYPTO_BA+0x520	R	SHA/HMAC SHAKE Digest Message 8	0x0000_0000
CRYPTO_HMAC_SHAKEDGST9	CRYPTO_BA+0x524	R	SHA/HMAC SHAKE Digest Message 9	0x0000_0000
CRYPTO_HMAC_SHAKEDGST10	CRYPTO_BA+0x528	R	SHA/HMAC SHAKE Digest Message 10	0x0000_0000
CRYPTO_HMAC_SHAKEDGST11	CRYPTO_BA+0x52C	R	SHA/HMAC SHAKE Digest Message 11	0x0000_0000
CRYPTO_HMAC_SHAKEDGST12	CRYPTO_BA+0x530	R	SHA/HMAC SHAKE Digest Message 12	0x0000_0000
CRYPTO_HMAC_SHAKEDGST13	CRYPTO_BA+0x534	R	SHA/HMAC SHAKE Digest Message 13	0x0000_0000
CRYPTO_HMAC_SHAKEDGST14	CRYPTO_BA+0x538	R	SHA/HMAC SHAKE Digest Message 14	0x0000_0000
CRYPTO_HMAC_SHAKEDGST15	CRYPTO_BA+0x53C	R	SHA/HMAC SHAKE Digest Message 15	0x0000_0000
CRYPTO_HMAC_SHAKEDGST16	CRYPTO_BA+0x540	R	SHA/HMAC SHAKE Digest Message 16	0x0000_0000
CRYPTO_HMAC_SHAKEDGST17	CRYPTO_BA+0x544	R	SHA/HMAC SHAKE Digest Message 17	0x0000_0000
CRYPTO_HMAC_SHAKEDGST18	CRYPTO_BA+0x548	R	SHA/HMAC SHAKE Digest Message 18	0x0000_0000
CRYPTO_HMAC_SHAKEDGST19	CRYPTO_BA+0x54C	R	SHA/HMAC SHAKE Digest Message 19	0x0000_0000
CRYPTO_HMAC_SHAKEDGST20	CRYPTO_BA+0x550	R	SHA/HMAC SHAKE Digest Message 20	0x0000_0000
CRYPTO_HMAC_SHAKEDGST21	CRYPTO_BA+0x554	R	SHA/HMAC SHAKE Digest Message 21	0x0000_0000
CRYPTO_HMAC_SHAKEDGST22	CRYPTO_BA+0x558	R	SHA/HMAC SHAKE Digest Message 22	0x0000_0000
CRYPTO_HMAC_SHAKEDGST23	CRYPTO_BA+0x55C	R	SHA/HMAC SHAKE Digest Message 23	0x0000_0000
CRYPTO_HMAC_SHAKEDGST24	CRYPTO_BA+0x560	R	SHA/HMAC SHAKE Digest Message 24	0x0000_0000
CRYPTO_HMAC_SHAKEDGST25	CRYPTO_BA+0x564	R	SHA/HMAC SHAKE Digest Message 25	0x0000_0000
CRYPTO_HMAC_SHAKEDGST26	CRYPTO_BA+0x568	R	SHA/HMAC SHAKE Digest Message 26	0x0000_0000
CRYPTO_HMAC_SHAKEDGST27	CRYPTO_BA+0x56C	R	SHA/HMAC SHAKE Digest Message 27	0x0000_0000
CRYPTO_HMAC_SHAKEDGST28	CRYPTO_BA+0x570	R	SHA/HMAC SHAKE Digest Message 28	0x0000_0000
CRYPTO_HMAC_SHAKEDGST29	CRYPTO_BA+0x574	R	SHA/HMAC SHAKE Digest Message 29	0x0000_0000
CRYPTO_HMAC_SHAKEDGST30	CRYPTO_BA+0x578	R	SHA/HMAC SHAKE Digest Message 30	0x0000_0000
CRYPTO_HMAC_SHAKEDGST31	CRYPTO_BA+0x57C	R	SHA/HMAC SHAKE Digest Message 31	0x0000_0000
CRYPTO_HMAC_SHAKEDGST32	CRYPTO_BA+0x580	R	SHA/HMAC SHAKE Digest Message 32	0x0000_0000
CRYPTO_HMAC_SHAKEDGST33	CRYPTO_BA+0x584	R	SHA/HMAC SHAKE Digest Message 33	0x0000_0000
CRYPTO_HMAC_SHAKEDGST34	CRYPTO_BA+0x588	R	SHA/HMAC SHAKE Digest Message 34	0x0000_0000
CRYPTO_HMAC_SHAKEDGST35	CRYPTO_BA+0x58C	R	SHA/HMAC SHAKE Digest Message 35	0x0000_0000
CRYPTO_HMAC_SHAKEDGST36	CRYPTO_BA+0x590	R	SHA/HMAC SHAKE Digest Message 36	0x0000_0000
CRYPTO_HMAC_SHAKEDGST37	CRYPTO_BA+0x594	R	SHA/HMAC SHAKE Digest Message 37	0x0000_0000

CRYPTO_HMAC_SHAKEDGST38	CRYPTO_BA+0x598	R	SHA/HMAC SHAKE Digest Message 38	0x0000_0000
CRYPTO_HMAC_SHAKEDGST39	CRYPTO_BA+0x59C	R	SHA/HMAC SHAKE Digest Message 39	0x0000_0000
CRYPTO_HMAC_SHAKEDGST40	CRYPTO_BA+0x5A0	R	SHA/HMAC SHAKE Digest Message 40	0x0000_0000
CRYPTO_HMAC_SHAKEDGST41	CRYPTO_BA+0x5A4	R	SHA/HMAC SHAKE Digest Message 41	0x0000_0000
CRYPTO_ECC_CTL	CRYPTO_BA+0x800	R/W	ECC Control Register	0x0000_0000
CRYPTO_ECC_STS	CRYPTO_BA+0x804	R	ECC Status Register	0x0000_0000
CRYPTO_ECC_X1_00	CRYPTO_BA+0x808	R/W	ECC the X-coordinate Word0 of the First Point	0x0000_0000
CRYPTO_ECC_X1_01	CRYPTO_BA+0x80C	R/W	ECC the X-coordinate Word1 of the First Point	0x0000_0000
CRYPTO_ECC_X1_02	CRYPTO_BA+0x810	R/W	ECC the X-coordinate Word2 of the First Point	0x0000_0000
CRYPTO_ECC_X1_03	CRYPTO_BA+0x814	R/W	ECC the X-coordinate Word3 of the First Point	0x0000_0000
CRYPTO_ECC_X1_04	CRYPTO_BA+0x818	R/W	ECC the X-coordinate Word4 of the First Point	0x0000_0000
CRYPTO_ECC_X1_05	CRYPTO_BA+0x81C	R/W	ECC the X-coordinate Word5 of the First Point	0x0000_0000
CRYPTO_ECC_X1_06	CRYPTO_BA+0x820	R/W	ECC the X-coordinate Word6 of the First Point	0x0000_0000
CRYPTO_ECC_X1_07	CRYPTO_BA+0x824	R/W	ECC the X-coordinate Word7 of the First Point	0x0000_0000
CRYPTO_ECC_X1_08	CRYPTO_BA+0x828	R/W	ECC the X-coordinate Word8 of the First Point	0x0000_0000
CRYPTO_ECC_X1_09	CRYPTO_BA+0x82C	R/W	ECC the X-coordinate Word9 of the First Point	0x0000_0000
CRYPTO_ECC_X1_10	CRYPTO_BA+0x830	R/W	ECC the X-coordinate Word10 of the First Point	0x0000_0000
CRYPTO_ECC_X1_11	CRYPTO_BA+0x834	R/W	ECC the X-coordinate Word11 of the First Point	0x0000_0000
CRYPTO_ECC_X1_12	CRYPTO_BA+0x838	R/W	ECC the X-coordinate Word12 of the First Point	0x0000_0000
CRYPTO_ECC_X1_13	CRYPTO_BA+0x83C	R/W	ECC the X-coordinate Word13 of the First Point	0x0000_0000
CRYPTO_ECC_X1_14	CRYPTO_BA+0x840	R/W	ECC the X-coordinate Word14 of the First Point	0x0000_0000
CRYPTO_ECC_X1_15	CRYPTO_BA+0x844	R/W	ECC the X-coordinate Word15 of the First Point	0x0000_0000
CRYPTO_ECC_X1_16	CRYPTO_BA+0x848	R/W	ECC the X-coordinate Word16 of the First Point	0x0000_0000
CRYPTO_ECC_X1_17	CRYPTO_BA+0x84C	R/W	ECC the X-coordinate Word17 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_00	CRYPTO_BA+0x850	R/W	ECC the Y-coordinate Word0 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_01	CRYPTO_BA+0x854	R/W	ECC the Y-coordinate Word1 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_02	CRYPTO_BA+0x858	R/W	ECC the Y-coordinate Word2 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_03	CRYPTO_BA+0x85C	R/W	ECC the Y-coordinate Word3 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_04	CRYPTO_BA+0x860	R/W	ECC the Y-coordinate Word4 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_05	CRYPTO_BA+0x864	R/W	ECC the Y-coordinate Word5 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_06	CRYPTO_BA+0x868	R/W	ECC the Y-coordinate Word6 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_07	CRYPTO_BA+0x86C	R/W	ECC the Y-coordinate Word7 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_08	CRYPTO_BA+0x870	R/W	ECC the Y-coordinate Word8 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_09	CRYPTO_BA+0x874	R/W	ECC the Y-coordinate Word9 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_10	CRYPTO_BA+0x878	R/W	ECC the Y-coordinate Word10 of the First Point	0x0000_0000

CRYPTO_ECC_Y1_11	CRYPTO_BA+0x87C	R/W	ECC the Y-coordinate Word11 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_12	CRYPTO_BA+0x880	R/W	ECC the Y-coordinate Word12 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_13	CRYPTO_BA+0x884	R/W	ECC the Y-coordinate Word13 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_14	CRYPTO_BA+0x888	R/W	ECC the Y-coordinate Word14 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_15	CRYPTO_BA+0x88C	R/W	ECC the Y-coordinate Word15 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_16	CRYPTO_BA+0x890	R/W	ECC the Y-coordinate Word16 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_17	CRYPTO_BA+0x894	R/W	ECC the Y-coordinate Word17 of the First Point	0x0000_0000
CRYPTO_ECC_X2_00	CRYPTO_BA+0x898	R/W	ECC the X-coordinate Word0 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_01	CRYPTO_BA+0x89C	R/W	ECC the X-coordinate Word1 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_02	CRYPTO_BA+0x8A0	R/W	ECC the X-coordinate Word2 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_03	CRYPTO_BA+0x8A4	R/W	ECC the X-coordinate Word3 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_04	CRYPTO_BA+0x8A8	R/W	ECC the X-coordinate Word4 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_05	CRYPTO_BA+0x8AC	R/W	ECC the X-coordinate Word5 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_06	CRYPTO_BA+0x8B0	R/W	ECC the X-coordinate Word6 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_07	CRYPTO_BA+0x8B4	R/W	ECC the X-coordinate Word7 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_08	CRYPTO_BA+0x8B8	R/W	ECC the X-coordinate Word8 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_09	CRYPTO_BA+0x8BC	R/W	ECC the X-coordinate Word9 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_10	CRYPTO_BA+0x8C0	R/W	ECC the X-coordinate Word10 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_11	CRYPTO_BA+0x8C4	R/W	ECC the X-coordinate Word11 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_12	CRYPTO_BA+0x8C8	R/W	ECC the X-coordinate Word12 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_13	CRYPTO_BA+0x8CC	R/W	ECC the X-coordinate Word13 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_14	CRYPTO_BA+0x8D0	R/W	ECC the X-coordinate Word14 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_15	CRYPTO_BA+0x8D4	R/W	ECC the X-coordinate Word15 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_16	CRYPTO_BA+0x8D8	R/W	ECC the X-coordinate Word16 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_17	CRYPTO_BA+0x8DC	R/W	ECC the X-coordinate Word17 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_00	CRYPTO_BA+0x8E0	R/W	ECC the Y-coordinate Word0 of the Second Point	0x0000_0000

CRYPTO_ECC_Y2_01	CRYPTO_BA+0x8E4	R/W	ECC the Y-coordinate Word1 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_02	CRYPTO_BA+0x8E8	R/W	ECC the Y-coordinate Word2 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_03	CRYPTO_BA+0x8EC	R/W	ECC the Y-coordinate Word3 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_04	CRYPTO_BA+0x8F0	R/W	ECC the Y-coordinate Word4 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_05	CRYPTO_BA+0x8F4	R/W	ECC the Y-coordinate Word5 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_06	CRYPTO_BA+0x8F8	R/W	ECC the Y-coordinate Word6 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_07	CRYPTO_BA+0x8FC	R/W	ECC the Y-coordinate Word7 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_08	CRYPTO_BA+0x900	R/W	ECC the Y-coordinate Word8 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_09	CRYPTO_BA+0x904	R/W	ECC the Y-coordinate Word9 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_10	CRYPTO_BA+0x908	R/W	ECC the Y-coordinate Word10 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_11	CRYPTO_BA+0x90C	R/W	ECC the Y-coordinate Word11 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_12	CRYPTO_BA+0x910	R/W	ECC the Y-coordinate Word12 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_13	CRYPTO_BA+0x914	R/W	ECC the Y-coordinate Word13 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_14	CRYPTO_BA+0x918	R/W	ECC the Y-coordinate Word14 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_15	CRYPTO_BA+0x91C	R/W	ECC the Y-coordinate Word15 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_16	CRYPTO_BA+0x920	R/W	ECC the Y-coordinate Word16 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_17	CRYPTO_BA+0x924	R/W	ECC the Y-coordinate Word17 of the Second Point	0x0000_0000
CRYPTO_ECC_A_00	CRYPTO_BA+0x928	R/W	ECC the Parameter CURVEA Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_01	CRYPTO_BA+0x92C	R/W	ECC the Parameter CURVEA Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_02	CRYPTO_BA+0x930	R/W	ECC the Parameter CURVEA Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_03	CRYPTO_BA+0x934	R/W	ECC the Parameter CURVEA Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_04	CRYPTO_BA+0x938	R/W	ECC the Parameter CURVEA Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_05	CRYPTO_BA+0x93C	R/W	ECC the Parameter CURVEA Word5 of Elliptic Curve	0x0000_0000

CRYPTO_ECC_A_06	CRYPTO_BA+0x940	R/W	ECC the Parameter CURVEA Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_07	CRYPTO_BA+0x944	R/W	ECC the Parameter CURVEA Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_08	CRYPTO_BA+0x948	R/W	ECC the Parameter CURVEA Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_09	CRYPTO_BA+0x94C	R/W	ECC the Parameter CURVEA Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_10	CRYPTO_BA+0x950	R/W	ECC the Parameter CURVEA Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_11	CRYPTO_BA+0x954	R/W	ECC the Parameter CURVEA Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_12	CRYPTO_BA+0x958	R/W	ECC the Parameter CURVEA Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_13	CRYPTO_BA+0x95C	R/W	ECC the Parameter CURVEA Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_14	CRYPTO_BA+0x960	R/W	ECC the Parameter CURVEA Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_15	CRYPTO_BA+0x964	R/W	ECC the Parameter CURVEA Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_16	CRYPTO_BA+0x968	R/W	ECC the Parameter CURVEA Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_17	CRYPTO_BA+0x96C	R/W	ECC the Parameter CURVEA Word17 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_00	CRYPTO_BA+0x970	R/W	ECC the Parameter CURVEB Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_01	CRYPTO_BA+0x974	R/W	ECC the Parameter CURVEB Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_02	CRYPTO_BA+0x978	R/W	ECC the Parameter CURVEB Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_03	CRYPTO_BA+0x97C	R/W	ECC the Parameter CURVEB Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_04	CRYPTO_BA+0x980	R/W	ECC the Parameter CURVEB Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_05	CRYPTO_BA+0x984	R/W	ECC the Parameter CURVEB Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_06	CRYPTO_BA+0x988	R/W	ECC the Parameter CURVEB Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_07	CRYPTO_BA+0x98C	R/W	ECC the Parameter CURVEB Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_08	CRYPTO_BA+0x990	R/W	ECC the Parameter CURVEB Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_09	CRYPTO_BA+0x994	R/W	ECC the Parameter CURVEB Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_10	CRYPTO_BA+0x998	R/W	ECC the Parameter CURVEB Word10 of Elliptic Curve	0x0000_0000

CRYPTO_ECC_B_11	CRYPTO_BA+0x99C	R/W	ECC the Parameter CURVEB Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_12	CRYPTO_BA+0x9A0	R/W	ECC the Parameter CURVEB Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_13	CRYPTO_BA+0x9A4	R/W	ECC the Parameter CURVEB Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_14	CRYPTO_BA+0x9A8	R/W	ECC the Parameter CURVEB Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_15	CRYPTO_BA+0x9AC	R/W	ECC the Parameter CURVEB Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_16	CRYPTO_BA+0x9B0	R/W	ECC the Parameter CURVEB Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_17	CRYPTO_BA+0x9B4	R/W	ECC the Parameter CURVEB Word17 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_00	CRYPTO_BA+0x9B8	R/W	ECC the Parameter CURVEN Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_01	CRYPTO_BA+0x9BC	R/W	ECC the Parameter CURVEN Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_02	CRYPTO_BA+0x9C0	R/W	ECC the Parameter CURVEN Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_03	CRYPTO_BA+0x9C4	R/W	ECC the Parameter CURVEN Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_04	CRYPTO_BA+0x9C8	R/W	ECC the Parameter CURVEN Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_05	CRYPTO_BA+0x9CC	R/W	ECC the Parameter CURVEN Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_06	CRYPTO_BA+0x9D0	R/W	ECC the Parameter CURVEN Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_07	CRYPTO_BA+0x9D4	R/W	ECC the Parameter CURVEN Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_08	CRYPTO_BA+0x9D8	R/W	ECC the Parameter CURVEN Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_09	CRYPTO_BA+0x9DC	R/W	ECC the Parameter CURVEN Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_10	CRYPTO_BA+0x9E0	R/W	ECC the Parameter CURVEN Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_11	CRYPTO_BA+0x9E4	R/W	ECC the Parameter CURVEN Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_12	CRYPTO_BA+0x9E8	R/W	ECC the Parameter CURVEN Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_13	CRYPTO_BA+0x9EC	R/W	ECC the Parameter CURVEN Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_14	CRYPTO_BA+0x9F0	R/W	ECC the Parameter CURVEN Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_15	CRYPTO_BA+0x9F4	R/W	ECC the Parameter CURVEN Word15 of Elliptic Curve	0x0000_0000

CRYPTO_ECC_N_16	CRYPTO_BA+0x9F8	R/W	ECC the Parameter CURVEN Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_17	CRYPTO_BA+0x9FC	R/W	ECC the Parameter CURVEN Word17 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_K_00	CRYPTO_BA+0xA00	W	ECC the Scalar SCALARK Word0 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_01	CRYPTO_BA+0xA04	W	ECC the Scalar SCALARK Word1 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_02	CRYPTO_BA+0xA08	W	ECC the Scalar SCALARK Word2 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_03	CRYPTO_BA+0xA0C	W	ECC the Scalar SCALARK Word3 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_04	CRYPTO_BA+0xA10	W	ECC the Scalar SCALARK Word4 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_05	CRYPTO_BA+0xA14	W	ECC the Scalar SCALARK Word5 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_06	CRYPTO_BA+0xA18	W	ECC the Scalar SCALARK Word6 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_07	CRYPTO_BA+0xA1C	W	ECC the Scalar SCALARK Word7 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_08	CRYPTO_BA+0xA20	W	ECC the Scalar SCALARK Word8 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_09	CRYPTO_BA+0xA24	W	ECC the Scalar SCALARK Word9 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_10	CRYPTO_BA+0xA28	W	ECC the Scalar SCALARK Word10 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_11	CRYPTO_BA+0xA2C	W	ECC the Scalar SCALARK Word11 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_12	CRYPTO_BA+0xA30	W	ECC the Scalar SCALARK Word12 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_13	CRYPTO_BA+0xA34	W	ECC the Scalar SCALARK Word13 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_14	CRYPTO_BA+0xA38	W	ECC the Scalar SCALARK Word14 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_15	CRYPTO_BA+0xA3C	W	ECC the Scalar SCALARK Word15 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_16	CRYPTO_BA+0xA40	W	ECC the Scalar SCALARK Word16 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_17	CRYPTO_BA+0xA44	W	ECC the Scalar SCALARK Word17 of Point Multiplication	0x0000_0000
CRYPTO_ECC_SADDR	CRYPTO_BA+0xA48	R/W	ECC DMA Source Address Register	0x0000_0000
CRYPTO_ECC_DADDR	CRYPTO_BA+0xA4C	R/W	ECC DMA Destination Address Register	0x0000_0000
CRYPTO_ECC_STARTREG	CRYPTO_BA+0xA50	R/W	ECC Starting Address of Updated Registers	0x0000_0000
CRYPTO_ECC_WORDCNT	CRYPTO_BA+0xA54	R/W	ECC DMA Word Count	0x0000_0000
CRYPTO_RSA_CTL	CRYPTO_BA+0xB00	R/W	RSA Control Register	0x0000_0000

CRYPTO_RSA_STS	CRYPTO_BA+0xB04	R	RSA Status Register	0x0000_0000
CRYPTO_RSA_SADDR0	CRYPTO_BA+0xB08	R/W	RSA DMA Source Address Register0	0x0000_0000
CRYPTO_RSA_SADDR1	CRYPTO_BA+0xB0C	R/W	RSA DMA Source Address Register1	0x0000_0000
CRYPTO_RSA_SADDR2	CRYPTO_BA+0xB10	R/W	RSA DMA Source Address Register2	0x0000_0000
CRYPTO_RSA_SADDR3	CRYPTO_BA+0xB14	R/W	RSA DMA Source Address Register3	0x0000_0000
CRYPTO_RSA_SADDR4	CRYPTO_BA+0xB18	R/W	RSA DMA Source Address Register4	0x0000_0000
CRYPTO_RSA_DADDR	CRYPTO_BA+0xB1C	R/W	RSA DMA Destination Address Register	0x0000_0000
CRYPTO_RSA_MADDR0	CRYPTO_BA+0xB20	R/W	RSA DMA Middle Address Register0	0x0000_0000
CRYPTO_RSA_MADDR1	CRYPTO_BA+0xB24	R/W	RSA DMA Middle Address Register1	0x0000_0000
CRYPTO_RSA_MADDR2	CRYPTO_BA+0xB28	R/W	RSA DMA Middle Address Register2	0x0000_0000
CRYPTO_RSA_MADDR3	CRYPTO_BA+0xB2C	R/W	RSA DMA Middle Address Register3	0x0000_0000
CRYPTO_RSA_MADDR4	CRYPTO_BA+0xB30	R/W	RSA DMA Middle Address Register4	0x0000_0000
CRYPTO_RSA_MADDR5	CRYPTO_BA+0xB34	R/W	RSA DMA Middle Address Register5	0x0000_0000
CRYPTO_RSA_MADDR6	CRYPTO_BA+0xB38	R/W	RSA DMA Middle Address Register6	0x0000_0000
CRYPTO_PRNG_KSCTL	CRYPTO_BA+0xF00	W	PRNG Key Control Register	0x0000_0000
CRYPTO_PRNG_KSSTS	CRYPTO_BA+0xF04	R	PRNG Key Status Register	0x0000_0000
CRYPTO_AES_KSCTL	CRYPTO_BA+0xF10	W	AES Key Control Register	0x0000_0000
CRYPTO_HMAC_KSCTL	CRYPTO_BA+0xF30	W	HMAC Key Control Register	0x0000_0000
CRYPTO_ECC_KSCTL	CRYPTO_BA+0xF40	W	ECC Key Control Register	0x0000_0000
CRYPTO_ECC_KSSTS	CRYPTO_BA+0xF44	R	ECC Key Status Register	0x0000_0000
CRYPTO_ECC_KSXY	CRYPTO_BA+0xF48	W	ECC XY Number Register	0x0000_0000
CRYPTO_RSA_KSCTL	CRYPTO_BA+0xF50	W	RSA Key Control Register	0x0000_0000
CRYPTO_RSA_KSSTS0	CRYPTO_BA+0xF54	R/W	RSA Key Status Register 0	0x0000_0000
CRYPTO_RSA_KSSTS1	CRYPTO_BA+0xF58	R/W	RSA Key Status Register 1	0x0000_0000

6.40.7 Register Description

6.40.7.1 Crypto Register

CRYPTO Interrupt Enable Control Register (CRYPTO_INTEN)

Register	Offset	R/W	Description	Reset Value
CRYPTO_INTEN	CRYPTO_BA+0x000	R/W	Crypto Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
RSAEIEN	RSAIEN	Reserved				HMACEIEN	HMACIEN
23	22	21	20	19	18	17	16
ECCEIEN	ECCIEN	Reserved				PRNGEIEN	PRNGIEN
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						AESEIEN	AESIEN

Bits	Description	
[31]	**RSAEIEN**	**RSA Error Interrupt Enable Bit** 0 = RSA error interrupt flag Disabled. 1 = RSA error interrupt flag Enabled.
[30]	**RSAIEN**	**RSA Interrupt Enable Bit** 0 = RSA interrupt Disabled. 1 = RSA interrupt Enabled.
[29:26]	**Reserved**	Reserved.
[25]	**HMACEIEN**	**SHA/HMAC Error Interrupt Enable Bit** 0 = SHA/HMAC error interrupt flag Disabled. 1 = HMAC error interrupt flag Enabled.
[24]	**HMACIEN**	**SHA/HMAC Interrupt Enable Bit** 0 = SHA/HMAC interrupt Disabled. 1 = SHA/HMAC interrupt Enabled. **Note:** In DMA mode, an interrupt will be triggered when an amount of data set in HMAC_DMA_CNT is fed into the SHA/HMAC engine. In Non-DMA mode, an interrupt will be triggered when the SHA/HMAC engine finishes the operation.
[23]	**ECCEIEN**	**ECC Error Interrupt Enable Bit** 0 = ECC error interrupt flag Disabled. 1 = ECC error interrupt flag Enabled.
[22]	**ECCIEN**	**ECC Interrupt Enable Bit** 0 = ECC interrupt Disabled. 1 = ECC interrupt Enabled. **Note:** In DMA mode, an interrupt will be triggered when an amount of data set in ECC_DMA_CNT is fed into the ECC engine. In Non-DMA mode, an interrupt will be triggered when the ECC engine finishes the operation.

[21:18]	**Reserved**	Reserved.
[17]	**PRNGEIEN**	**PRNG Error Flag Enable Bit** 0 = PRNG error interrupt flag Disabled. 1 = PRNG error interrupt flag Enabled.
[16]	**PRNGIEN**	**PRNG Interrupt Enable Bit** 0 = PRNG interrupt Disabled. 1 = PRNG interrupt Enabled.
[15:2]	**Reserved**	Reserved.
[1]	**AESEIEN**	**AES Error Flag Enable Bit** 0 = AES error interrupt flag Disabled. 1 = AES error interrupt flag Enabled.
[0]	**AESIEN**	**AES Interrupt Enable Bit** 0 = AES interrupt Disabled. 1 = AES interrupt Enabled. **Note:** In DMA mode, an interrupt will be triggered when an amount of data set in AES_DMA_CNT is fed into the AES engine. In Non-DMA mode, an interrupt will be triggered when the AES engine finishes the operation.

CRYPTO Interrupt Flag Register (CRYPTO_INTSTS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_INTSTS	CRYPTO_BA+0x004	R/W	Crypto Interrupt Flag	0x0000_0000

31	30	29	28	27	26	25	24
RSAEIF	RSAIF	Reserved				HMACEIF	HMACIF
23	22	21	20	19	18	17	16
ECCEIF	ECCIF	Reserved				PRNGEIF	PRNGIF
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						AESEIF	AESIF

Bits	Description	
[31]	**RSAEIF**	**RSA Error Interrupt Flag** This register includes operating and setting error. The detail flag is shown in CRYPTO_RSA_STS register. 0 = No RSA error. 1 = RSA error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[30]	**RSAIF**	**RSA Finish Interrupt Flag** 0 = No RSA interrupt. 1 = RSA operation done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[29:26]	**Reserved**	Reserved.
[25]	**HMACEIF**	**SHA/HMAC Error Flag** This register includes operating and setting error. The detail flag is shown in CRYPTO_HMAC_STS register. 0 = No SHA/HMAC error. 1 = SHA/HMAC error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[24]	**HMACIF**	**SHA/HMAC Finish Interrupt Flag** 0 = No SHA/HMAC interrupt. 1 = SHA/HMAC operation done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[23]	**ECCEIF**	**ECC Error Flag** This register includes operating and setting error. The detail flag is shown in CRYPTO_ECC_STS register. 0 = No ECC error. 1 = ECC error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.

[22]	**ECCIF**	**ECC Finish Interrupt Flag** 0 = No ECC interrupt. 1 = ECC operation done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[21:18]	**Reserved**	Reserved.
[17]	**PRNGEIF**	**PRNG Error Flag** 0 = No PRNG error. 1 = PRNG key generation error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[16]	**PRNGIF**	**PRNG Finish Interrupt Flag** 0 = No PRNG interrupt. 1 = PRNG key generation done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[15:2]	**Reserved**	Reserved.
[1]	**AESEIF**	**AES Error Flag** 0 = No AES error. 1 = AES encryption/decryption error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[0]	**AESIF**	**AES Finish Interrupt Flag** 0 = No AES interrupt. 1 = AES encryption/decryption done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.

6.40.7.2 PRNG Register

PRNG Control Register (CRYPTO_PRNG_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_CTL	CRYPTO_BA+0x008	R/W	PRNG Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							SEEDSRC
15	14	13	12	11	10	9	8
Reserved							BUSY
7	6	5	4	3	2	1	0
Reserved		KEYSZ				SEEDRLD	START

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**SEEDSRC**	**Seed Source** 0 = Seed is from TRNG. 1 = Seed is from PRNG seed register. **Note:** When SEEDRLD is set to 0, this bit (SEEDSRC) is meaningless.
[15:9]	**Reserved**	Reserved.
[8]	**BUSY**	**PRNG Busy (Read Only)** 0 = PRNG engine is idle. 1 = PRNG engine is generating CRYPTO_PRNG_KEYx.
[7:6]	**Reserved**	Reserved.

[5:2]	**KEYSZ**	**PRNG Generate Key Size** 0000 = 128 bits. 0001 = 163 bits. 0010 = 192 bits. 0011 = 224 bits. 0100 = 233 bits. 0101 = 255 bits. 0110 = 256 bits. 0111 = 283 bits (only for KS). 1000 = 384 bits (only for KS). 1001 = 409 bits (only for KS). 1010 = 512 bits (only for KS). 1011 = 521 bits (only for KS). 1100 = 571 bits (only for KS). 1101 = Reserved. 1110 = Reserved. 1111 = Reserved. **Note:** 283~571 bits are only generated for key store.
[1]	**SEEDRLD**	**Reload New Seed for PRNG Engine** 0 = Generating key based on the current seed. 1 = Reload new seed.
[0]	**START**	**Start PRNG Engine** 0 = Stop PRNG engine. 1 = Generate new key and store the new key to register CRYPTO_PRNG_KEYx, which will be cleared when the new key is generated.

PRNG Seed Register (CRYPTO_PRNG_SEED)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_SEED	CRYPTO_BA+0x00C	W	Seed for PRNG	Undefined

31	30	29	28	27	26	25	24
SEED							
23	22	21	20	19	18	17	16
SEED							
15	14	13	12	11	10	9	8
SEED							
7	6	5	4	3	2	1	0
SEED							

Bits	Description	
[31:0]	SEED	**Seed for PRNG (Write Only)** The bits store the seed for PRNG engine. **Note:** In TRNG+PRNG mode, the seed is from TRNG engine, and it will not be stored in this register.

PRNG Key x Register (CRYPTO_PRNG_KEYx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KEY0	CRYPTO_BA+0x010	R	PRNG Generated Key0	Undefined
CRYPTO_PRNG_KEY1	CRYPTO_BA+0x014	R	PRNG Generated Key1	Undefined
CRYPTO_PRNG_KEY2	CRYPTO_BA+0x018	R	PRNG Generated Key2	Undefined
CRYPTO_PRNG_KEY3	CRYPTO_BA+0x01C	R	PRNG Generated Key3	Undefined
CRYPTO_PRNG_KEY4	CRYPTO_BA+0x020	R	PRNG Generated Key4	Undefined
CRYPTO_PRNG_KEY5	CRYPTO_BA+0x024	R	PRNG Generated Key5	Undefined
CRYPTO_PRNG_KEY6	CRYPTO_BA+0x028	R	PRNG Generated Key6	Undefined
CRYPTO_PRNG_KEY7	CRYPTO_BA+0x02C	R	PRNG Generated Key7	Undefined

31	30	29	28	27	26	25	24
KEY							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
KEY							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
KEY							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
KEY							

Bits	Description	
[31:0]	**KEY**	**Store PRNG Generated Key (Read Only)** The bits store the key that is generated by PRNG.

PRNG Status Register (CRYPTO_PRNG_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_STS	CRYPTO_BA+0x030	R	PRNG Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						KSERR	KCTLERR
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							BUSY

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**KSERR**	**PRNG Access Key Store Error Flag** 0 = No error. 1 = Access key store failed.
[16]	**KCTLERR**	**PRNG Key Control Register Error Flag** 0 = No error. 1 = PRNG key control error. When PRNG execute ECDSA or ECDH, but PRNG seed not from TRNG or key is not written to the SRAM of key store (WSDST, CRYPTO_PRNG_KSCTL[23:22] is not equal to '00').
[15:1]	**Reserved**	Reserved.
[0]	**BUSY**	**PRNG Busy Flag** 0 = PRNG engine is idle. 1 = PRNG engine is generating CRYPTO_PRNG_KEYx.

6.40.7.3 AES Register

AES Feedback x Register (CRYPTO_AES_FDBCKx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_FDBCK0	CRYPTO_BA+0x050	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK1	CRYPTO_BA+0x054	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK2	CRYPTO_BA+0x058	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK3	CRYPTO_BA+0x05C	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000

31	30	29	28	27	26	25	24
FDBCK							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FDBCK							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FDBCK							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FDBCK							

Bits	Description	
[31:0]	**FDBCK**	**AES Feedback Information** The feedback value is 128 bits in size. The AES engine uses the data from CRYPTO_AES_FDBCKx as the data inputted to CRYPTO_AES_IVx for the next block in DMA cascade mode. The AES engine outputs feedback information for IV in the next block's operation. Software can use this feedback information to implement more than four DMA channels. Software can store that feedback value temporarily. After switching back, fill the stored feedback value to CRYPTO_AES_IVx in the same channel operation, and then continue the operation with the original setting.

AES GCM IV Byte Count Register 0 (CRYPTO_AES_GCM_IVCNT0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_IVCNT0	CRYPTO_BA+0x080	R/W	AES GCM IV Byte Count Register 0	0x0000_0000

31	30	29	28	27	26	25	24
CNT							
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:0]	CNT	**AES GCM IV Byte Count** The bit length of IV is 64 bits for AES GCM mode. The CRYPTO_AES_GCM_IVCNT0 keeps the low weight byte count of initial vector (i.e., len(IV)[34:3]) of AES GCM mode and can be read and written.

AES GCM IV Byte Count Register 1 (CRYPTO_AES_GCM_IVCNT1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_IVCNT1	CRYPTO_BA+0x084	R/W	AES GCM IV Byte Count Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			CNT				
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:0]	**CNT**	**AES GCM IV Byte Count** The bit length of IV is 64 bits for AES GCM mode. The CRYPTO_AES_GCM_IVCNT1 keeps the high weight byte count of initial vector (i.e., len(IV)[64:35]) of AES GCM mode and can be read and written.

AES GCM A Byte Count Register 0 (CRYPTO_AES_GCM_ACNT0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_ACNT0	CRYPTO_BA+0x088	R/W	AES GCM A Byte Count Register 0	0x0000_0000

31	30	29	28	27	26	25	24
CNT							
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:0]	CNT	**AES GCM a Byte Count** The bit length of A is 64 bits for AES GCM mode. The CRYPTO_AES_GCM_ACNT0 keeps the low weight byte count of the additional authenticated data (i.e., len(A)[34:3]) of AES GCM mode and can be read and written.

AES GCM A Byte Count Register 1 (CRYPTO_AES_GCM_ACNT1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_ACNT1	CRYPTO_BA+0x08C	R/W	AES GCM A Byte Count Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			CNT				
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:29]	Reserved	Reserved.
[28:0]	CNT	**AES GCM a Byte Count** The bit length of A is 64 bits for AES GCM mode. The CRYPTO_AES_GCM_ACNT0 keeps the high weight byte count of the additional authenticated data (i.e., len(A)[63:35]) of AES GCM mode and can be read and written.

AES GCM P Byte Count Register 0 (CRYPTO_AES_GCM_PCNT0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_PCNT0	CRYPTO_BA+0x090	R/W	AES GCM P Byte Count Register 0	0x0000_0000

31	30	29	28	27	26	25	24
CNT							
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:0]	CNT	**AES GCM P Byte Count** The bit length of Por C is 39 bits for AES GCM mode. The CRYPTO_AES_GCM_PCNT0 keeps the low weight byte count of the plaintext or ciphertext (i.e., len(P)[34:3] or len(C)[34:3]) of AES GCM mode and can be read and written.

AES GCM P Byte Count Register 1 (CRYPTO_AES_GCM_PCNT1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_GCM_PCNT1	CRYPTO_BA+0x094	R/W	AES GCM P Byte Count Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			CNT				
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:0]	**CNT**	**AES GCM P Byte Count** The bit length of Por C is 39 bits for AES GCM mode. The CRYPTO_AES_GCM_PCNT1 keeps the high weight byte count of the plaintext or ciphertext (i.e., len(P)[38:35] or len(C)[38:35]) of AES GCM mode and can be read and written. The bit length of Por C is 64 bits for AES CCM mode. The CRYPTO_AES_GCM_PCNT1 keeps the high weight byte count of the plaintext or ciphertext (i.e., len(P)[63:35] or len(C)[63:35]) of AES CCM mode and can be read and written.

AES DMA Feedback Address Register (CRYPTO_AES_FBADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_FBADDR	CRYPTO_BA+0x0A0	R/W	AES DMA Feedback Address Register	0x0000_0000

31	30	29	28	27	26	25	24
FBADDR							
23	22	21	20	19	18	17	16
FBADDR							
15	14	13	12	11	10	9	8
FBADDR							
7	6	5	4	3	2	1	0
FBADDR							

Bits	Description	
[31:0]	FBADDR	**AES DMA Feedback Address** In DMA cascade mode, software can update DMA feedback address register for automatically reading and writing feedback values via DMA. The FBADDR keeps the feedback address of the feedback data for the next cascade operation. Based on the feedback address, the AES accelerator can read the feedback data of the last cascade operation from SRAM memory space and write the feedback data of the current cascade operation to SRAM memory space. The start of feedback address should be located at word boundary. In other words, bit 1 and 0 of FBADDR are ignored. FBADDR can be read and written. In DMA mode, software can update the next CRYPTO_AES_FBADDR before triggering START.

AES Control Register (CRYPTO_AES_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_CTL	CRYPTO_BA+0x100	R/W	AES Control Register	0x0000_0000

31	30	29	28	27	26	25	24
KEYPRT	KEYUNPRT					KINSWAP	KOUTSWAP
23	22	21	20	19	18	17	16
INSWAP	OUTSWAP	FBOUT	FBIN	Reserved			ENCRYPTO
15	14	13	12	11	10	9	8
OPMODE							
7	6	5	4	3	2	1	0
DMAEN	DMACSCAD	DMALAST	Reserved	KEYSZ		STOP	START

Bits	Description	
[31]	**KEYPRT**	**Protect Key** Read as a flag to reflect KEYPRT. 0 = No effect. 1 = Protect the content of the AES key from reading. The return value for reading CRYPTO_AES_KEYx is not the content of the registers CRYPTO_AES_KEYx. Once it is set, it can be cleared by asserting KEYUNPRT. The key content would be cleared as well.
[30:26]	**KEYUNPRT**	**Unprotect Key** Writing 0 to CRYPTO_AES_CTL[31] and “10110” to CRYPTO_AES_CTL[30:26] is to unprotect the AES key. The KEYUNPRT can be read and written. When it is written as the AES engine is operating, BUSY flag is 1, there would be no effect on KEYUNPRT.
[25]	**KINSWAP**	**AES Engine Input Key and Initial Vector Swap** 0 = Keep the original order. 1 = The order that CPU feeds key and initial vector to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[24]	**KOUTSWAP**	**AES Engine Output Key, Initial Vector and Feedback Swap** 0 = Keep the original order. 1 = The order that CPU reads key, initial vector and feedback from the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[23]	**INSWAP**	**AES Engine Input Data Swap** 0 = Keep the original order. 1 = The order that CPU feeds data to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[22]	**OUTSWAP**	**AES Engine Output Data Swap** 0 = Keep the original order. 1 = The order that CPU reads data from the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.

[21]	**FBOUT**	**Feedback Output From AES Via DMA Automatically** 0 = DMA automatic feedback output function Disabled. 1 = DMA automatic feedback output function Enabled when DMAEN = 1.
[20]	**FBIN**	**Feedback Input to AES Via DMA Automatically** 0 = DMA automatic feedback input function Disabled. 1 = DMA automatic feedback input function Enabled when DMAEN = 1.
[19:17]	**Reserved**	Reserved.
[16]	**ENCRYPTO**	**AES Encryption/Decryption** 0 = AES engine executes decryption operation. 1 = AES engine executes encryption operation.
[15:8]	**OPMODE**	**AES Engine Operation Modes** 0x00 = ECB (Electronic Codebook Mode) 0x01 = CBC (Cipher Block Chaining Mode). 0x02 = CFB (Cipher Feedback Mode). 0x03 = OFB (Output Feedback Mode). 0x04 = CTR (Counter Mode). 0x10 = CBC-CS1 (CBC Ciphertext-Stealing 1 Mode). 0x11 = CBC-CS2 (CBC Ciphertext-Stealing 2 Mode). 0x12 = CBC-CS3 (CBC Ciphertext-Stealing 3 Mode). 0x20 = GCM (Galois/Counter Mode). 0x21 = GHASH (Galois Hash Function). 0x22 = CCM (Counter with CBC-MAC Mode).
[7]	**DMAEN**	**AES Engine DMA Enable Bit** 0 = AES DMA engine Disabled. The AES engine operates in Non-DMA mode. The data need to be written in CRYPTO_AES_DATIN. 1 = AES_DMA engine Enabled. The AES engine operates in DMA mode, and data movement from/to the engine is done by DMA logic.
[6]	**DMACSCAD**	**AES Engine DMA with Cascade Mode** 0 = DMA cascade function Disabled. 1 = In DMA cascade mode, software can update DMA source address register, destination address register, and byte count register during a cascade operation, without finishing the accelerator operation. **Note:** The last two blocks of AES-CBC-CS1/2/3 must be in the last cascade operation.
[5]	**DMALAST**	**AES Last Block** In DMA mode, this bit must be set as beginning the last DMA cascade round. In Non-DMA mode, this bit must be set when feeding in the last block of data in ECB, CBC, CTR, OFB, and CFB mode, and feeding in the (last-1) block of data at CBC-CS1, CBC-CS2, and CBC-CS3 mode. This bit is always 0 when it is read back, and must be written again once START is triggered.
[4]	**Reserved**	Reserved.
[3:2]	**KEYSZ**	**AES Key Size** This bit defines three different key size for AES operation. 2'b00 = 128 bits key. 2'b01 = 192 bits key. 2'b10 = 256 bits key. 2'b11 = Reserved. If the AES accelerator is operating and the corresponding flag BUSY is 1, updating this register has no effect.

[1]	**STOP**	**AES Engine Stop** 0 = No effect. 1 = Stop AES engine. **Note:** This bit is always 0 when it is read back.
[0]	**START**	**AES Engine Start** 0 = No effect. 1 = Start AES engine. BUSY flag will be set. **Note:** This bit is always 0 when it is read back.

AES Status Flag Register (CRYPTO_AES_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_STS	CRYPTO_BA+0x104	R	AES Engine Flag	0x0001_0100

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		KSERR	BUSERR	Reserved	OUTBUFERR	OUTBUFFULL	OUTBUFEMPTY
15	14	13	12	11	10	9	8
Reserved			CNTERR	Reserved	INBUFERR	INBUFFULL	INBUFEMPTY
7	6	5	4	3	2	1	0
Reserved							BUSY

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**KSERR**	**AES Engine Access Key Store Error Flag** 0 = No error. 1 = Key store access error will stop AES engine.
[20]	**BUSERR**	**AES DMA Access Bus Error Flag** 0 = No error. 1 = Bus error will stop DMA operation and AES engine.
[19]	**Reserved**	Reserved.
[18]	**OUTBUFERR**	**AES Out Buffer Error Flag** 0 = No error. 1 = Error happened during getting the result from AES engine.
[17]	**OUTBUFFULL**	**AES Out Buffer Full Flag** 0 = AES output buffer is not full. 1 = AES output buffer is full, and software needs to get data from CRYPTO_AES_DATOUT. Otherwise, the AES engine will be pending since the output buffer is full.
[16]	**OUTBUFEMPTY**	**AES Out Buffer Empty** 0 = AES output buffer is not empty. There are some valid data kept in output buffer. 1 = AES output buffer is empty. Software cannot get data from CRYPTO_AES_DATOUT. Otherwise, the flag OUTBUFERR will be set to 1 since the output buffer is empty.
[15:13]	**Reserved**	Reserved.
[12]	**CNTERR**	**CRYPTO_AES_CNT Setting Error** 0 = No error in CRYPTO_AES_CNT setting. 1 = CRYPTO_AES_CNT is 0 if DMAEN (CRYPTO_AES_CTL[7]) is enabled.
[11]	**Reserved**	Reserved.

[10]	**INBUFERR**	**AES Input Buffer Error Flag** 0 = No error. 1 = Error happened during feeding data to the AES engine.
[9]	**INBUFFULL**	**AES Input Buffer Full Flag** 0 = AES input buffer is not full. Software can feed the data into the AES engine. 1 = AES input buffer is full. Software cannot feed data to the AES engine. Otherwise, the flag INBUFERR will be set to 1.
[8]	**INBUFEMPTY**	**AES Input Buffer Empty** 0 = There are some data in input buffer waiting for the AES engine to process. 1 = AES input buffer is empty. Software needs to feed data to the AES engine. Otherwise, the AES engine will be pending to wait for input data.
[7:1]	**Reserved**	Reserved.
[0]	**BUSY**	**AES Engine Busy** 0 = The AES engine is idle or finished. 1 = The AES engine is under processing.

AES Data Input Port Register (CRYPTO_AES_DATIN)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DATIN	CRYPTO_BA+0x108	R/W	AES Engine Data Input Port Register	0x0000_0000

31	30	29	28	27	26	25	24
DATIN							
23	22	21	20	19	18	17	16
DATIN							
15	14	13	12	11	10	9	8
DATIN							
7	6	5	4	3	2	1	0
DATIN							

Bits	Description	
[31:0]	DATIN	**AES Engine Input Port** CPU feeds data to AES engine through this port by checking CRYPTO_AES_STS. Feed data as INBUFFULL is 0.

AES Data Output Port Register (CRYPTO_AES_DATOUT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DATOUT	CRYPTO_BA+0x10C	R	AES Engine Data Output Port Register	0x0000_0000

31	30	29	28	27	26	25	24
DATOUT							
23	22	21	20	19	18	17	16
DATOUT							
15	14	13	12	11	10	9	8
DATOUT							
7	6	5	4	3	2	1	0
DATOUT							

Bits	Description	
[31:0]	DATOUT	**AES Engine Output Port** CPU gets results from the AES engine through this port by checking CRYPTO_AES_STS. Get data as OUTBUFEMPTY is 0.

AES Key Word x Register (CRYPTO_AES_KEYx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_KEY0	CRYPTO_BA+0x110	R/W	AES Key Word 0 Register	0x0000_0000
CRYPTO_AES_KEY1	CRYPTO_BA+0x114	R/W	AES Key Word 1 Register	0x0000_0000
CRYPTO_AES_KEY2	CRYPTO_BA+0x118	R/W	AES Key Word 2 Register	0x0000_0000
CRYPTO_AES_KEY3	CRYPTO_BA+0x11C	R/W	AES Key Word 3 Register	0x0000_0000
CRYPTO_AES_KEY4	CRYPTO_BA+0x120	R/W	AES Key Word 4 Register	0x0000_0000
CRYPTO_AES_KEY5	CRYPTO_BA+0x124	R/W	AES Key Word 5 Register	0x0000_0000
CRYPTO_AES_KEY6	CRYPTO_BA+0x128	R/W	AES Key Word 6 Register	0x0000_0000
CRYPTO_AES_KEY7	CRYPTO_BA+0x12C	R/W	AES Key Word 7 Register	0x0000_0000

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

Bits	Description	
[31:0]	**KEY**	**CRYPTO_AES_KEYx** The KEY keeps the security key for AES operation. x = 0, 1..7. The security key for AES accelerator can be 128, 192, or 256 bits and four, six, or eight 32-bit registers are to store each security key. {CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 128-bit security key for AES operation. {CRYPTO_AES_KEY5, CRYPTO_AES_KEY4, CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 192-bit security key for AES operation. {CRYPTO_AES_KEY7, CRYPTO_AES_KEY6, CRYPTO_AES_KEY5, CRYPTO_AES_KEY4, CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 256-bit security key for AES operation.

AES Initial Vector Word x Register (CRYPTO_AES_IVx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_IV0	CRYPTO_BA+0x130	R/W	AES Initial Vector Word 0 Register	0x0000_0000
CRYPTO_AES_IV1	CRYPTO_BA+0x134	R/W	AES Initial Vector Word 1 Register	0x0000_0000
CRYPTO_AES_IV2	CRYPTO_BA+0x138	R/W	AES Initial Vector Word 2 Register	0x0000_0000
CRYPTO_AES_IV3	CRYPTO_BA+0x13C	R/W	AES Initial Vector Word 3 Register	0x0000_0000

31	30	29	28	27	26	25	24
IV							
23	22	21	20	19	18	17	16
IV							
15	14	13	12	11	10	9	8
IV							
7	6	5	4	3	2	1	0
IV							

Bits		Description
[31:0]	**IV**	**AES Initial Vectors** x = 0, 1..3. Four initial vectors (CRYPTO_AES_IV0, CRYPTO_AES_IV1, CRYPTO_AES_IV2, and CRYPTO_AES_IV3) are for AES operating in CBC, CFB, and OFB mode. Four registers (CRYPTO_AES_IV0, CRYPTO_AES_IV1, CRYPTO_AES_IV2, and CRYPTO_AES_IV3) act as Nonce counter when the AES engine is operating in CTR mode.

AES DMA Source Address Register (CRYPTO_AES_SADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_SADDR	CRYPTO_BA+0x140	R/W	AES DMA Source Address Register	0x0000_0000

31	30	29	28	27	26	25	24
SADDR							
23	22	21	20	19	18	17	16
SADDR							
15	14	13	12	11	10	9	8
SADDR							
7	6	5	4	3	2	1	0
SADDR							

Bits	Description	
[31:0]	**SADDR**	**AES DMA Source Address** The AES accelerator supports DMA function to transfer the plain text between SRAM memory space and embedded FIFO. The SADDR keeps the source address of the data buffer where the source text is stored. Based on the source address, the AES accelerator can read the plain text (encryption) / cipher text (decryption) from SRAM memory space and do AES operation. The start of source address should be located at word boundary. In other words, bit 1 and 0 of SADDR are ignored. SADDR can be read and written. Writing to SADDR while the AES accelerator is operating doesn't affect the current AES operation. But the value of SADDR will be updated later on. Consequently, software can prepare the DMA source address for the next AES operation. In DMA mode, software can update the next CRYPTO_AES_SADDR before triggering START. The value of CRYPTO_AES_SADDR and CRYPTO_AES_DADDR can be the same.

AES DMA Destination Address Register (CRYPTO_AES_DADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DADDR	CRYPTO_BA+0x144	R/W	AES DMA Destination Address Register	0x0000_0000

31	30	29	28	27	26	25	24
DADDR							
23	22	21	20	19	18	17	16
DADDR							
15	14	13	12	11	10	9	8
DADDR							
7	6	5	4	3	2	1	0
DADDR							

Bits	Description	
[31:0]	DADDR	**AES DMA Destination Address** The AES accelerator supports DMA function to transfer the cipher text between SRAM memory space and embedded FIFO. The DADDR keeps the destination address of the data buffer where the engine output's text will be stored. Based on the destination address, the AES accelerator can write the cipher text (encryption) / plain text (decryption) back to SRAM memory space after the AES operation is finished. The start of destination address should be located at word boundary. In other words, bit 1 and 0 of DADDR are ignored. DADDR can be read and written. Writing to DADDR while the AES accelerator is operating doesn't affect the current AES operation. But the value of DADDR will be updated later on. Consequently, software can prepare the destination address for the next AES operation. In DMA mode, software can update the next CRYPTO_AES_DADDR before triggering START. The value of CRYPTO_AES_SADDR and CRYPTO_AES_DADDR can be the same.

AES Byte Count Register (CRYPTO_AES_CNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_CNT	CRYPTO_BA+0x148	R/W	AES Byte Count Register	0x0000_0000

31	30	29	28	27	26	25	24
CNT							
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:0]	CNT	**AES Byte Count** The CRYPTO_AES_CNT keeps the byte count of source text that is for the AES engine operating in DMA mode. The CRYPTO_AES_CNT is 32-bit and the maximum of byte count is 4G bytes. CRYPTO_AES_CNT can be read and written. Writing to CRYPTO_AES_CNT while the AES accelerator is operating doesn't affect the current AES operation. But the value of CRYPTO_AES_CNT will be updated later on. Consequently, software can prepare the byte count of data for the next AES operation. According to CBC-CS1, CBC-CS2, and CBC-CS3 standard, the count of operation data must be more than 16 bytes. Operations that are equal to or less than one block will output unexpected result. In Non-DMA ECB, CBC, CFB, OFB, CTR, CCM and GCM mode, CRYPTO_AES_CNT must be set as byte count for the last block of data before feeding in the last block of data. In Non-DMA CBC-CS1, CBC-CS2, and CBC-CS3 mode, CRYPTO_AES_CNT must be set as byte count for the last two blocks of data before feeding in the last two blocks of data. In AES GCM mode without DMA cascade function, the value of CRYPTO_AES_CNT is equal to the total value of {CRYPTO_AES_GCM_IVCNT1, CRYPTO_AES_GCM_IVCNT0}, {CRYPTO_AES_GCM_ACNT1, CRYPTO_AES_GCM_ACNT0} and {CRYPTO_AES_GCM_PCNT1, CRYPTO_AES_GCM_PCNT0}. In AES GCM mode with DMA cascade function, the value of CRYPTO_AES_CNT represents the byte count of source text in this cascade function. Thus, the value of CRYPTO_AES_CNT is less than or equal to the total value of {CRYPTO_AES_GCM_IVCNT1, CRYPTO_AES_GCM_IVCNT0}, {CRYPTO_AES_GCM_ACNT1, CRYPTO_AES_GCM_ACNT0} and {CRYPTO_AES_GCM_PCNT1, CRYPTO_AES_GCM_PCNT0} and must be block alignment. In AES CCM mode without DMA cascade function, the value of CRYPTO_AES_CNT is equal to the total value of {CRYPTO_AES_GCM_ACNT1, CRYPTO_AES_GCM_ACNT0} and {CRYPTO_AES_GCM_PCNT1, CRYPTO_AES_GCM_PCNT0}. In AES CCM mode with DMA cascade function, the value of CRYPTO_AES_CNT represents the byte count of source text in this cascade function. Thus, the value of CRYPTO_AES_CNT is less than or equal to the total value of {CRYPTO_AES_GCM_ACNT1, CRYPTO_AES_GCM_ACNT0} and {CRYPTO_AES_GCM_PCNT1, CRYPTO_AES_GCM_PCNT0} and must be block alignment, except for the last block of plaintext or ciphertext.

6.40.7.4 SHA/HMAC Register

SHA/HMAC Control Register (CRYPTO_HMAC_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_CTL	CRYPTO_BA+0x300	R/W	SHA/HMAC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						FINISHDGST	NEXTDGST
23	22	21	20	19	18	17	16
INSWAP	OUTSWAP	FBOUT	FBIN	Reserved			
15	14	13	12	11	10	9	8
Reserved			SHA3EN	HMACEN	OPMODE		
7	6	5	4	3	2	1	0
DMAEN	DMACSCAD	DMALAST	DMAFIRST	Reserved		STOP	START

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**FINISHDGST**	**SHAKE128/256 Next Digest Finish** 0 = No effect. 1 = finish generating the next digest.
[24]	**NEXTDGST**	**SHAKE128/256 Next Digest Start** 0 = No effect. 1 = Start SHAKE engine to generate the next digest only when SHAKEBUSY is 0. BUSY and SHAKEBUSY flag will be set.
[23]	**INSWAP**	**SHA/HMAC Engine Input Data Swap** 0 = Keep the original order. 1 = The order that CPU feeds data to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[22]	**OUTSWAP**	**SHA/HMAC Engine Output Data Swap** 0 = Keep the original order. 1 = The order that CPU feeds data to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[21]	**FBOUT**	**Feedback Output From SHA/HMAC Via DMA Automatically** 0 = DMA automatic feedback output function Disabled. 1 = DMA automatic feedback output function Enabled when DMAEN = 1.
[20]	**FBIN**	**Feedback Input to SHA/HMAC Via DMA Automatically** 0 = DMA automatic feedback input function Disabled. 1 = DMA automatic feedback input function Enabled when DMAEN = 1.
[19:13]	**Reserved**	Reserved.
[12]	**SHA3EN**	**SHA3 Engine Enable Bit** 0 = Execute other function. 1 = Execute SHA3 function.

[11]	**HMACEN**	**HMAC_SHA Engine Operating Mode** 0 = Execute SHA function. 1 = Execute HMAC function.
[10:8]	**OPMODE**	**SHA/HMAC Engine Operation Modes** When SHA3EN=0,. 0x0xx: SHA1-160 0x100: SHA2-256 0x101: SHA2-224 0x110: SHA2-512 0x111: SHA2-384 When SHA3EN=1,. 0x100: SHA3-256 0x101: SHA3-224 0x110: SHA3-512 0x111: SHA3-384 0x000: SHAKE128 0x001: SHAKE256 **Note:** These bits can be read and written. But writing to them wouldn't take effect as BUSY is 1.
[7]	**DMAEN**	**SHA/HMAC Engine DMA Enable Bit** 0 = SHA/HMAC DMA engine Disabled. SHA/HMAC engine operates in Non-DMA mode. The data need to be written in CRYPTO_HMAC_DATIN. 1 = SHA/HMAC DMA engine Enabled. SHA/HMAC engine operates in DMA mode, and data movement from/to the engine is done by DMA logic.
[6]	**DMACSCAD**	**SHA/HMAC Engine DMA with Cascade Mode** 0 = DMA cascade function Disabled. 1 = In DMA cascade mode, software can update DMA source address register, destination address register, and byte count register during a cascade operation, without finishing the accelerator operation.
[5]	**DMALAST**	**SHA/HMAC Last Block** This bit must be set as feeding in last byte of data.
[4]	**DMAFIRST**	**SHA/HMAC First Block in Cascade function** This bit must be set as feeding in first byte of data.
[3:2]	**Reserved**	Reserved.
[1]	**STOP**	**SHA/HMAC Engine Stop** 0 = No effect. 1 = Stop SHA/HMAC engine. **Note:** This bit is always 0 when it is read back.
[0]	**START**	**SHA/HMAC Engine Start** 0 = No effect. 1 = Start SHA/HMAC engine. BUSY flag will be set. **Note:** This bit is always 0 when it is read back.

SHA/HMAC Status Register (CRYPTO_HMAC_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_STS	CRYPTO_BA+0x304	R	SHA/HMAC Status Flag	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							DATINREQ
15	14	13	12	11	10	9	8
Reserved						KSERR	DMAERR
7	6	5	4	3	2	1	0
Reserved					SHAKEBUSY	DMABUSY	BUSY

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DATINREQ**	**SHA/HMAC Non-DMA Mode Data Input Request** 0 = No effect. 1 = Request SHA/HMAC Non-DMA mode data input.
[15:10]	**Reserved**	Reserved.
[9]	**KSERR**	**HMAC Engine Access Key Store Error Flag** 0 = No error. 1 = Access error will stop HMAC engine.
[8]	**DMAERR**	**SHA/HMAC Engine DMA Error Flag** 0 = Show the SHA/HMAC engine access normal. 1 = Show the SHA/HMAC engine access error.
[7:3]	**Reserved**	Reserved.
[2]	**SHAKEBUSY**	**SHAKE Engine Busy Flag** 0 = SHAKE engine is idle or finished. 1 = SHAKE engine is busy.
[1]	**DMABUSY**	**SHA/HMAC Engine DMA Busy Flag** 0 = SHA/HMAC DMA engine is idle or finished. 1 = SHA/HMAC DMA engine is busy.
[0]	**BUSY**	**SHA/HMAC Engine Busy** 0 = SHA/HMAC engine is idle or finished. 1 = SHA/HMAC engine is busy.

SHA/HMAC Outputs Digest Word Register (CRYPTO_HMAC_DGSTx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_DGST0	CRYPTO_BA+0x308	R	SHA/HMAC Output Feedback Data 0	0x0000_0000
CRYPTO_HMAC_DGST1	CRYPTO_BA+0x30C	R	SHA/HMAC Output Feedback Data 1	0x0000_0000
CRYPTO_HMAC_DGST2	CRYPTO_BA+0x310	R	SHA/HMAC Output Feedback Data 2	0x0000_0000
CRYPTO_HMAC_DGST3	CRYPTO_BA+0x314	R	SHA/HMAC Output Feedback Data 3	0x0000_0000
CRYPTO_HMAC_DGST4	CRYPTO_BA+0x318	R	SHA/HMAC Output Feedback Data 4	0x0000_0000
CRYPTO_HMAC_DGST5	CRYPTO_BA+0x31C	R	SHA/HMAC Output Feedback Data 5	0x0000_0000
CRYPTO_HMAC_DGST6	CRYPTO_BA+0x320	R	SHA/HMAC Output Feedback Data 6	0x0000_0000
CRYPTO_HMAC_DGST7	CRYPTO_BA+0x324	R	SHA/HMAC Output Feedback Data 7	0x0000_0000
CRYPTO_HMAC_DGST8	CRYPTO_BA+0x328	R	SHA/HMAC Output Feedback Data 8	0x0000_0000
CRYPTO_HMAC_DGST9	CRYPTO_BA+0x32C	R	SHA/HMAC Output Feedback Data 9	0x0000_0000
CRYPTO_HMAC_DGST10	CRYPTO_BA+0x330	R	SHA/HMAC Output Feedback Data 10	0x0000_0000
CRYPTO_HMAC_DGST11	CRYPTO_BA+0x334	R	SHA/HMAC Output Feedback Data 11	0x0000_0000
CRYPTO_HMAC_DGST12	CRYPTO_BA+0x338	R	SHA/HMAC Output Feedback Data 12	0x0000_0000
CRYPTO_HMAC_DGST13	CRYPTO_BA+0x33C	R	SHA/HMAC Output Feedback Data 13	0x0000_0000
CRYPTO_HMAC_DGST14	CRYPTO_BA+0x340	R	SHA/HMAC Output Feedback Data 14	0x0000_0000
CRYPTO_HMAC_DGST15	CRYPTO_BA+0x344	R	SHA/HMAC Output Feedback Data 15	0x0000_0000

31	30	29	28	27	26	25	24
DGST							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DGST							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DGST							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DGST							

Bits	Description	
[31:0]	**DGST**	**SHA/HMAC Output Feedback Data Output Register** For SHA-160, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST4. For SHA-224, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST6. For SHA-256, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST7. For SHA-384, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST11. For SHA-512, the digest is stored in CRYPTO_HMAC_DGST0 ~ CRYPTO_HMAC_DGST15.

SHA/HMAC Key Byte Count Register (CRYPTO_HMAC_KEYCNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_KEYCNT	CRYPTO_BA+0x348	R/W	SHA/HMAC Key Byte Count Register	0x0000_0000

31	30	29	28	27	26	25	24
KEYCNT							
23	22	21	20	19	18	17	16
KEYCNT							
15	14	13	12	11	10	9	8
KEYCNT							
7	6	5	4	3	2	1	0
KEYCNT							

Bits	Description	
[31:0]	KEYCNT	**SHA/HMAC Key Byte Count** The CRYPTO_HMAC_KEYCNT keeps the byte count of key that SHA/HMAC engine operates. The register is 32-bit and the key byte count is between 4 and 4G bytes. It can be read and written. Writing to the register CRYPTO_HMAC_KEYCNT as the SHA/HMAC accelerator operating doesn't affect the current SHA/HMAC operation. But the value of CRYPTO_HMAC_KEYCNT will be updated later on. Consequently, software can prepare the key count for the next SHA/HMAC operation.

SHA/HMAC DMA Source Address Register (CRYPTO_HMAC_SADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_SADDR	CRYPTO_BA+0x34C	R/W	SHA/HMAC DMA Source Address Register	0x0000_0000

31	30	29	28	27	26	25	24
SADDR							
23	22	21	20	19	18	17	16
SADDR							
15	14	13	12	11	10	9	8
SADDR							
7	6	5	4	3	2	1	0
SADDR							

Bits	Description	
[31:0]	**SADDR**	**SHA/HMAC DMA Source Address** The SHA/HMAC accelerator supports DMA function to transfer the plain text between SRAM memory space and embedded FIFO. The CRYPTO_HMAC_SADDR keeps the source address of the data buffer where the source text is stored. Based on the source address, the SHA/HMAC accelerator can read the plain text from SRAM memory space and do SHA/HMAC operation. The start of source address should be located at word boundary. In other words, bit 1 and 0 of CRYPTO_HMAC_SADDR are ignored. CRYPTO_HMAC_SADDR can be read and written. Writing to CRYPTO_HMAC_SADDR while the SHA/HMAC accelerator is operating doesn't affect the current SHA/HMAC operation. But the value of CRYPTO_HMAC_SADDR will be updated later on. Consequently, software can prepare the DMA source address for the next SHA/HMAC operation. In DMA mode, software can update the next CRYPTO_HMAC_SADDR before triggering START. CRYPTO_HMAC_SADDR and CRYPTO_HMAC_DADDR can be the same in the value.

SHA/HMAC Byte Count Register (CRYPTO_HMAC_DMACNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_DMACNT	CRYPTO_BA+0x350	R/W	SHA/HMAC Byte Count Register	0x0000_0000

31	30	29	28	27	26	25	24
DMACNT							
23	22	21	20	19	18	17	16
DMACNT							
15	14	13	12	11	10	9	8
DMACNT							
7	6	5	4	3	2	1	0
DMACNT							

Bits	Description	
[31:0]	DMACNT	**SHA/HMAC Operation Byte Count** The CRYPTO_HMAC_DMACNT keeps the byte count of source text that is for the SHA/HMAC engine operating in DMA mode. The CRYPTO_HMAC_DMACNT is 32-bit and the maximum of byte count is 4G bytes. CRYPTO_HMAC_DMACNT can be read and written. Writing to CRYPTO_HMAC_DMACNT while the SHA/HMAC accelerator is operating doesn't affect the current SHA/HMAC operation. But the value of CRYPTO_HMAC_DMACNT will be updated later on. Consequently, software can prepare the byte count of data for the next SHA/HMAC operation. In Non-DMA mode, CRYPTO_HMAC_DMACNT must be set as the byte count of the last block before feeding in the last block of data.

SHA/HMAC Data Input Port Register (CRYPTO_HMAC_DATIN)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_DATIN	CRYPTO_BA+0x354	R/W	SHA/HMAC Engine Non-DMA Mode Data Input Port Register	0x0000_0000

31	30	29	28	27	26	25	24
DATIN							
23	22	21	20	19	18	17	16
DATIN							
15	14	13	12	11	10	9	8
DATIN							
7	6	5	4	3	2	1	0
DATIN							

Bits	Description	
[31:0]	DATIN	**SHA/HMAC Engine Input Port** CPU feeds data to SHA/HMAC engine through this port by checking CRYPTO_HMAC_STS. Feed data as DATINREQ is 1.

SHA/HMAC Feedback Register (CRYPTO_HMAC_FDBCKx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_FDBCK0	CRYPTO_BA+0x358	R/W	SHA/HMAC Output Feedback Data 0 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK1	CRYPTO_BA+0x35C	R/W	SHA/HMAC Output Feedback Data 1 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK2	CRYPTO_BA+0x360	R/W	SHA/HMAC Output Feedback Data 2 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK3	CRYPTO_BA+0x364	R/W	SHA/HMAC Output Feedback Data 3 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK4	CRYPTO_BA+0x368	R/W	SHA/HMAC Output Feedback Data 4 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK5	CRYPTO_BA+0x36C	R/W	SHA/HMAC Output Feedback Data 5 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK6	CRYPTO_BA+0x370	R/W	SHA/HMAC Output Feedback Data 6 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK7	CRYPTO_BA+0x374	R/W	SHA/HMAC Output Feedback Data 7 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK8	CRYPTO_BA+0x378	R/W	SHA/HMAC Output Feedback Data 8 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK9	CRYPTO_BA+0x37C	R/W	SHA/HMAC Output Feedback Data 9 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK10	CRYPTO_BA+0x380	R/W	SHA/HMAC Output Feedback Data 10 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK11	CRYPTO_BA+0x384	R/W	SHA/HMAC Output Feedback Data 11 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK12	CRYPTO_BA+0x388	R/W	SHA/HMAC Output Feedback Data 12 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK13	CRYPTO_BA+0x38C	R/W	SHA/HMAC Output Feedback Data 13 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK14	CRYPTO_BA+0x390	R/W	SHA/HMAC Output Feedback Data 14 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK15	CRYPTO_BA+0x394	R/W	SHA/HMAC Output Feedback Data 15 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK16	CRYPTO_BA+0x398	R/W	SHA/HMAC Output Feedback Data 16 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK17	CRYPTO_BA+0x39C	R/W	SHA/HMAC Output Feedback Data 17 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK18	CRYPTO_BA+0x3A0	R/W	SHA/HMAC Output Feedback Data 18 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK19	CRYPTO_BA+0x3A4	R/W	SHA/HMAC Output Feedback Data 19 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK20	CRYPTO_BA+0x3A8	R/W	SHA/HMAC Output Feedback Data 20 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK21	CRYPTO_BA+0x3AC	R/W	SHA/HMAC Output Feedback Data 21 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK22	CRYPTO_BA+0x3B0	R/W	SHA/HMAC Output Feedback Data 22 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK23	CRYPTO_BA+0x3B4	R/W	SHA/HMAC Output Feedback Data 23 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK24	CRYPTO_BA+0x3B8	R/W	SHA/HMAC Output Feedback Data 24 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK25	CRYPTO_BA+0x3BC	R/W	SHA/HMAC Output Feedback Data 25 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK26	CRYPTO_BA+0x3C0	R/W	SHA/HMAC Output Feedback Data 26 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK27	CRYPTO_BA+0x3C4	R/W	SHA/HMAC Output Feedback Data 27 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK28	CRYPTO_BA+0x3C8	R/W	SHA/HMAC Output Feedback Data 28 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK29	CRYPTO_BA+0x3CC	R/W	SHA/HMAC Output Feedback Data 29 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK30	CRYPTO_BA+0x3D0	R/W	SHA/HMAC Output Feedback Data 30 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK31	CRYPTO_BA+0x3D4	R/W	SHA/HMAC Output Feedback Data 31 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK32	CRYPTO_BA+0x3D8	R/W	SHA/HMAC Output Feedback Data 32 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK33	CRYPTO_BA+0x3DC	R/W	SHA/HMAC Output Feedback Data 33 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK34	CRYPTO_BA+0x3E0	R/W	SHA/HMAC Output Feedback Data 34 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK35	CRYPTO_BA+0x3E4	R/W	SHA/HMAC Output Feedback Data 35 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK36	CRYPTO_BA+0x3E8	R/W	SHA/HMAC Output Feedback Data 36 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK37	CRYPTO_BA+0x3EC	R/W	SHA/HMAC Output Feedback Data 37 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK38	CRYPTO_BA+0x3F0	R/W	SHA/HMAC Output Feedback Data 38 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK39	CRYPTO_BA+0x3F4	R/W	SHA/HMAC Output Feedback Data 39 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK40	CRYPTO_BA+0x3F8	R/W	SHA/HMAC Output Feedback Data 40 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK41	CRYPTO_BA+0x3FC	R/W	SHA/HMAC Output Feedback Data 41 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK42	CRYPTO_BA+0x400	R/W	SHA/HMAC Output Feedback Data 42 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK43	CRYPTO_BA+0x404	R/W	SHA/HMAC Output Feedback Data 43 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK44	CRYPTO_BA+0x408	R/W	SHA/HMAC Output Feedback Data 44 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK45	CRYPTO_BA+0x40C	R/W	SHA/HMAC Output Feedback Data 45 After	0x0000_0000

			SHA/HMAC Operation	
CRYPTO_HMAC_FDBCK46	CRYPTO_BA+0x410	R/W	SHA/HMAC Output Feedback Data 46 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK47	CRYPTO_BA+0x414	R/W	SHA/HMAC Output Feedback Data 47 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK48	CRYPTO_BA+0x418	R/W	SHA/HMAC Output Feedback Data 48 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK49	CRYPTO_BA+0x41C	R/W	SHA/HMAC Output Feedback Data 49 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK50	CRYPTO_BA+0x420	R/W	SHA/HMAC Output Feedback Data 50 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK51	CRYPTO_BA+0x424	R/W	SHA/HMAC Output Feedback Data 51 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK52	CRYPTO_BA+0x428	R/W	SHA/HMAC Output Feedback Data 52 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK53	CRYPTO_BA+0x42C	R/W	SHA/HMAC Output Feedback Data 53 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK54	CRYPTO_BA+0x430	R/W	SHA/HMAC Output Feedback Data 54 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK55	CRYPTO_BA+0x434	R/W	SHA/HMAC Output Feedback Data 55 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK56	CRYPTO_BA+0x438	R/W	SHA/HMAC Output Feedback Data 56 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK57	CRYPTO_BA+0x43C	R/W	SHA/HMAC Output Feedback Data 57 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK58	CRYPTO_BA+0x440	R/W	SHA/HMAC Output Feedback Data 58 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK59	CRYPTO_BA+0x444	R/W	SHA/HMAC Output Feedback Data 59 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK60	CRYPTO_BA+0x448	R/W	SHA/HMAC Output Feedback Data 60 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK61	CRYPTO_BA+0x44C	R/W	SHA/HMAC Output Feedback Data 61 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK62	CRYPTO_BA+0x450	R/W	SHA/HMAC Output Feedback Data 62 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK63	CRYPTO_BA+0x454	R/W	SHA/HMAC Output Feedback Data 63 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK64	CRYPTO_BA+0x458	R/W	SHA/HMAC Output Feedback Data 64 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK65	CRYPTO_BA+0x45C	R/W	SHA/HMAC Output Feedback Data 65 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK66	CRYPTO_BA+0x460	R/W	SHA/HMAC Output Feedback Data 66 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK67	CRYPTO_BA+0x464	R/W	SHA/HMAC Output Feedback Data 67 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK68	CRYPTO_BA+0x468	R/W	SHA/HMAC Output Feedback Data 68 After SHA/HMAC Operation	0x0000_0000

CRYPTO_HMAC_FDBCK69	CRYPTO_BA+0x46C	R/W	SHA/HMAC Output Feedback Data 69 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK70	CRYPTO_BA+0x470	R/W	SHA/HMAC Output Feedback Data 70 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK71	CRYPTO_BA+0x474	R/W	SHA/HMAC Output Feedback Data 71 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK72	CRYPTO_BA+0x478	R/W	SHA/HMAC Output Feedback Data 72 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK73	CRYPTO_BA+0x47C	R/W	SHA/HMAC Output Feedback Data 73 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK74	CRYPTO_BA+0x480	R/W	SHA/HMAC Output Feedback Data 74 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK75	CRYPTO_BA+0x484	R/W	SHA/HMAC Output Feedback Data 75 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK76	CRYPTO_BA+0x488	R/W	SHA/HMAC Output Feedback Data 76 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK77	CRYPTO_BA+0x48C	R/W	SHA/HMAC Output Feedback Data 77 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK78	CRYPTO_BA+0x490	R/W	SHA/HMAC Output Feedback Data 78 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK79	CRYPTO_BA+0x494	R/W	SHA/HMAC Output Feedback Data 79 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK80	CRYPTO_BA+0x498	R/W	SHA/HMAC Output Feedback Data 80 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK81	CRYPTO_BA+0x49C	R/W	SHA/HMAC Output Feedback Data 81 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK82	CRYPTO_BA+0x4A0	R/W	SHA/HMAC Output Feedback Data 82 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK83	CRYPTO_BA+0x4A4	R/W	SHA/HMAC Output Feedback Data 83 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK84	CRYPTO_BA+0x4A8	R/W	SHA/HMAC Output Feedback Data 84 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK85	CRYPTO_BA+0x4AC	R/W	SHA/HMAC Output Feedback Data 85 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK86	CRYPTO_BA+0x4B0	R/W	SHA/HMAC Output Feedback Data 86 After SHA/HMAC Operation	0x0000_0000
CRYPTO_HMAC_FDBCK87	CRYPTO_BA+0x4B4	R/W	SHA/HMAC Output Feedback Data 87 After SHA/HMAC Operation	0x0000_0000

31	30	29	28	27	26	25	24
FDBCK							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FDBCK							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FDBCK							

7	6	5	4	3	2	1	0
FDBCK							

Bits	Description	
[31:0]	**FDBCK**	**SHA/HMAC Feedback Information** The feedback value is 1728 bits in size for SHA1/2 and 2784 bits in size for SHA3. The SHA/HMAC engine uses the data from CRYPTO_HMAC_FDBCKx as the data inputted to CRYPTO_HMAC_FDBCKx for the next block in DMA cascade mode. The SHA/HMAC engine outputs feedback information for initial setting in the next block's operation. Software can store that feedback value temporarily. After switching back, fill the stored feedback value to CRYPTO_HMAC_FDBCKx in the same operation, and then continue the operation with the original setting.

SHA/HMAC SHA512T Control Register (CRYPTO_HMAC_SHA512T)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_SHA512T	CRYPTO_BA+0x4F8	R/W	SHA/HMAC SHA512/t Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							TLEN
15	14	13	12	11	10	9	8
TLEN							
7	6	5	4	3	2	1	0
Reserved							SHA512TEN

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16:8]	**TLEN**	**SHA512/t output digest length** The TLEN is equal to value t of SHA512/t. It also means the output digest length of SHA512 /t. **Note:** TLEN < 512, and TLEN is not 384.
[7:1]	**Reserved**	Reserved.
[0]	**SHA512TEN**	**SHA512/t Engine Enable Bit** 0 = Execute other function. 1 = Execute SHA512/t function if SHA3EN=0. **Note:** When SHA512TEN=1, SHA/HMAC only execute SHA2-512.

SHA/HMAC DMA Feedback Address Register (CRYPTO_HMAC_FBADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_FBADDR	CRYPTO_BA+0x4FC	R/W	SHA/HMAC DMA Feedback Address Register	0x0000_0000

31	30	29	28	27	26	25	24
FBADDR							
23	22	21	20	19	18	17	16
FBADDR							
15	14	13	12	11	10	9	8
FBADDR							
7	6	5	4	3	2	1	0
FBADDR							

Bits	Description	
[31:0]	FBADDR	**SHA/HMAC DMA Feedback Address** In DMA cascade mode, software can update DMA feedback address register for automatically reading and writing feedback values via DMA. The FBADDR keeps the feedback address of the feedback data for the next cascade operation. Based on the feedback address, the SHA/HMAC accelerator can read the feedback data of the last cascade operation from SRAM memory space and write the feedback data of the current cascade operation to SRAM memory space. The start of feedback address should be located at word boundary. In other words, bit 1 and 0 of FBADDR are ignored. FBADDR can be read and written. In DMA mode, software can update the next CRYPTO_HMAC_FBADDR before triggering START.

SHA/HMAC SHAKE Digest Message Register (CRYPTO_HMAC_SHAKEDGST0 x)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_SHAKEDGST0	CRYPTO_BA+0x500	R	SHA/HMAC SHAKE Digest Message 0	0x0000_0000
CRYPTO_HMAC_SHAKEDGST1	CRYPTO_BA+0x504	R	SHA/HMAC SHAKE Digest Message 1	0x0000_0000
CRYPTO_HMAC_SHAKEDGST2	CRYPTO_BA+0x508	R	SHA/HMAC SHAKE Digest Message 2	0x0000_0000
CRYPTO_HMAC_SHAKEDGST3	CRYPTO_BA+0x50C	R	SHA/HMAC SHAKE Digest Message 3	0x0000_0000
CRYPTO_HMAC_SHAKEDGST4	CRYPTO_BA+0x510	R	SHA/HMAC SHAKE Digest Message 4	0x0000_0000
CRYPTO_HMAC_SHAKEDGST5	CRYPTO_BA+0x514	R	SHA/HMAC SHAKE Digest Message 5	0x0000_0000
CRYPTO_HMAC_SHAKEDGST6	CRYPTO_BA+0x518	R	SHA/HMAC SHAKE Digest Message 6	0x0000_0000
CRYPTO_HMAC_SHAKEDGST7	CRYPTO_BA+0x51C	R	SHA/HMAC SHAKE Digest Message 7	0x0000_0000
CRYPTO_HMAC_SHAKEDGST8	CRYPTO_BA+0x520	R	SHA/HMAC SHAKE Digest Message 8	0x0000_0000
CRYPTO_HMAC_SHAKEDGST9	CRYPTO_BA+0x524	R	SHA/HMAC SHAKE Digest Message 9	0x0000_0000
CRYPTO_HMAC_SHAKEDGST10	CRYPTO_BA+0x528	R	SHA/HMAC SHAKE Digest Message 10	0x0000_0000
CRYPTO_HMAC_SHAKEDGST11	CRYPTO_BA+0x52C	R	SHA/HMAC SHAKE Digest Message 11	0x0000_0000
CRYPTO_HMAC_SHAKEDGST12	CRYPTO_BA+0x530	R	SHA/HMAC SHAKE Digest Message 12	0x0000_0000
CRYPTO_HMAC_SHAKEDGST13	CRYPTO_BA+0x534	R	SHA/HMAC SHAKE Digest Message 13	0x0000_0000
CRYPTO_HMAC_SHAKEDGST14	CRYPTO_BA+0x538	R	SHA/HMAC SHAKE Digest Message 14	0x0000_0000
CRYPTO_HMAC_SHAKEDGST15	CRYPTO_BA+0x53C	R	SHA/HMAC SHAKE Digest Message 15	0x0000_0000
CRYPTO_HMAC_SHAKEDGST16	CRYPTO_BA+0x540	R	SHA/HMAC SHAKE Digest Message 16	0x0000_0000
CRYPTO_HMAC_SHAKEDGST17	CRYPTO_BA+0x544	R	SHA/HMAC SHAKE Digest Message 17	0x0000_0000
CRYPTO_HMAC_SHAKEDGST18	CRYPTO_BA+0x548	R	SHA/HMAC SHAKE Digest Message 18	0x0000_0000
CRYPTO_HMAC_SHAKEDGST19	CRYPTO_BA+0x54C	R	SHA/HMAC SHAKE Digest Message 19	0x0000_0000
CRYPTO_HMAC_SHAKEDGST20	CRYPTO_BA+0x550	R	SHA/HMAC SHAKE Digest Message 20	0x0000_0000
CRYPTO_HMAC_SHAKEDGST21	CRYPTO_BA+0x554	R	SHA/HMAC SHAKE Digest Message 21	0x0000_0000
CRYPTO_HMAC_SHAKEDGST22	CRYPTO_BA+0x558	R	SHA/HMAC SHAKE Digest Message 22	0x0000_0000
CRYPTO_HMAC_SHAKEDGST23	CRYPTO_BA+0x55C	R	SHA/HMAC SHAKE Digest Message 23	0x0000_0000
CRYPTO_HMAC_SHAKEDGST24	CRYPTO_BA+0x560	R	SHA/HMAC SHAKE Digest Message 24	0x0000_0000
CRYPTO_HMAC_SHAKEDGST25	CRYPTO_BA+0x564	R	SHA/HMAC SHAKE Digest Message 25	0x0000_0000
CRYPTO_HMAC_SHAKEDGST26	CRYPTO_BA+0x568	R	SHA/HMAC SHAKE Digest Message 26	0x0000_0000
CRYPTO_HMAC_SHAKEDGST27	CRYPTO_BA+0x56C	R	SHA/HMAC SHAKE Digest Message 27	0x0000_0000
CRYPTO_HMAC_SHAKEDGST28	CRYPTO_BA+0x570	R	SHA/HMAC SHAKE Digest Message 28	0x0000_0000
CRYPTO_HMAC_SHAKEDGST29	CRYPTO_BA+0x574	R	SHA/HMAC SHAKE Digest Message 29	0x0000_0000
CRYPTO_HMAC_SHAKEDGST30	CRYPTO_BA+0x578	R	SHA/HMAC SHAKE Digest Message 30	0x0000_0000
CRYPTO_HMAC_SHAKEDGST31	CRYPTO_BA+0x57C	R	SHA/HMAC SHAKE Digest Message 31	0x0000_0000
CRYPTO_HMAC_SHAKEDGST32	CRYPTO_BA+0x580	R	SHA/HMAC SHAKE Digest Message 32	0x0000_0000
CRYPTO_HMAC_SHAKEDGST33	CRYPTO_BA+0x584	R	SHA/HMAC SHAKE Digest Message 33	0x0000_0000

CRYPTO_HMAC_SHAKEDGST34	CRYPTO_BA+0x588	R	SHA/HMAC SHAKE Digest Message 34	0x0000_0000
CRYPTO_HMAC_SHAKEDGST35	CRYPTO_BA+0x58C	R	SHA/HMAC SHAKE Digest Message 35	0x0000_0000
CRYPTO_HMAC_SHAKEDGST36	CRYPTO_BA+0x590	R	SHA/HMAC SHAKE Digest Message 36	0x0000_0000
CRYPTO_HMAC_SHAKEDGST37	CRYPTO_BA+0x594	R	SHA/HMAC SHAKE Digest Message 37	0x0000_0000
CRYPTO_HMAC_SHAKEDGST38	CRYPTO_BA+0x598	R	SHA/HMAC SHAKE Digest Message 38	0x0000_0000
CRYPTO_HMAC_SHAKEDGST39	CRYPTO_BA+0x59C	R	SHA/HMAC SHAKE Digest Message 39	0x0000_0000
CRYPTO_HMAC_SHAKEDGST40	CRYPTO_BA+0x5A0	R	SHA/HMAC SHAKE Digest Message 40	0x0000_0000
CRYPTO_HMAC_SHAKEDGST41	CRYPTO_BA+0x5A4	R	SHA/HMAC SHAKE Digest Message 41	0x0000_0000

31	30	29	28	27	26	25	24
DGST							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DGST							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DGST							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DGST							

Bits	Description	
[31:0]	**DGST**	**SHA/HMAC SHAKE Digest Message Register** For SHAKE-128, the digest is stored in CRYPTO_HMAC_SHAKEDGST0 0 ~ CRYPTO_HMAC_ SHAKEDGST0 41. For SHAKE-256, the digest is stored in CRYPTO_HMAC_SHAKEDGST0 0 ~ CRYPTO_HMAC_ SHAKEDGST0 33.

6.40.7.5 ECC Register

ECC Control Register (CRYPTO_ECC_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_CTL	CRYPTO_BA+0x800	R/W	ECC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CURVEM							
23	22	21	20	19	18	17	16
CURVEM		LDK	LDN	LDB	LDA	LDP2	LDP1
15	14	13	12	11	10	9	8
Reserved	SCAP	CSEL	MODOP		ECCOP		FSEL
7	6	5	4	3	2	1	0
DMAEN	DFAP	ECDSAR	ECDSAS	Reserved		STOP	START

Bits	Description	
[31:22]	**CURVEM**	The key length of elliptic curve.
[21]	**LDK**	**The Control Signal of Register for SCALARK** 0 = The register for SCALARK is not modified by DMA or user. 1 = The register for SCALARK is modified by DMA or user.
[20]	**LDN**	**The Control Signal of Register for the Parameter CURVEN of Elliptic Curve** 0 = The register for CURVEN is not modified by DMA or user. 1 = The register for CURVEN is modified by DMA or user.
[19]	**LDB**	**The Control Signal of Register for the Parameter CURVEB of Elliptic Curve** 0 = The register for CURVEB is not modified by DMA or user. 1 = The register for CURVEB is modified by DMA or user.
[18]	**LDA**	**The Control Signal of Register for the Parameter CURVEA of Elliptic Curve** 0 = The register for CURVEA is not modified by DMA or user. 1 = The register for CURVEA is modified by DMA or user.
[17]	**LDP2**	**The Control Signal of Register POINTX2 and POINTY2 for the x and Y Coordinate of the Second Point** 0 = The register for POINTX2 and POINTY2 is not modified by DMA or user. 1 = The register for POINTX2 and POINTY2 is modified by DMA or user.
[16]	**LDP1**	**The Control Signal of Register POINTX1 and POINTY1 for the x and Y Coordinate of the First Point** 0 = The register for POINTX1 and POINTY1 is not modified by DMA or user. 1 = The register for POINTX1 and POINTY1 is modified by DMA or user.
[15]	**Reserved**	Reserved.
[14]	**SCAP**	**Side-channel Attack Protection** 0 = Full speed without side-channel protection. 1 = Less speed with side-channel protection.

[13]	**CSEL**	**Curve Selection** 0 = NIST suggested curve. 1 = Montgomery curve.
[12:11]	**MODOP**	**Modulus Operation for PF** 00 = Division:. POINTX1 = (POINTY1 / POINTX1) % CURVEN. 01 = Multiplication:. POINTX1 = (POINTX1 * POINTY1) % CURVEN. 10 = Addition:. POINTX1 = (POINTX1 + POINTY1) % CURVEN. 11 = Subtraction:. POINTX1 = (POINTX1 - POINTY1) % CURVEN. MODOP is active only when ECCOP = 01.
[10:9]	**ECCOP**	**Point Operation for BF and PF** 00 = Point multiplication:. (POINTX1, POINTY1) = SCALARK * (POINTX1, POINTY1). 01 = Modulus operation: choose by MODOP (CRYPTO_ECC_CTL[12:11]). 10 = Point addition:. (POINTX1, POINTY1) = (POINTX1, POINTY1) +. (POINTX2, POINTY2) 11 = Point doubling:. (POINTX1, POINTY1) = 2 * (POINTX1, POINTY1). Besides above three input data, point operations still need the parameters of elliptic curve (CURVEA, CURVEB, CURVEN and CURVEM) as shown in Figure 6.27-11.
[8]	**FSEL**	**Field Selection** 0 = Binary Field (GF(2^m)). 1 = Prime Field (GF(p)).
[7]	**DMAEN**	**ECC Accelerator DMA Enable Bit** 0 = ECC DMA engine Disabled. 1 = ECC DMA engine Enabled. Only when START and DMAEN are 1, ECC DMA engine will be active.
[6]	**DFAP**	**Differential Fault Attack Protection** 0 = Differential fault attack protection Disabled. 1 = Differential fault attack protection Enabled.
[5]	**ECDSAR**	**Generate R in ECDSA Signature Generation** 0 = No effect. 1 = Formula for generating R. (POINTX1, POINTY1) = SCALARK * (POINTX1, POINTY1).
[4]	**ECDSAS**	**Generate S in ECDSA Signature Generation** 0 = No effect. 1 = Formula for generating S. POINTX1 = ((POINTX2 * POINTY1 + POINTY2) / POINTX1) % CURVEN.
[3:2]	**Reserved**	Reserved.

[1]	**STOP**	**ECC Accelerator Stop** 0 = No effect. 1 = Abort ECC accelerator and make it into idle state. This bit is always 0 when it is read back. Remember to clear ECC interrupt flag after stopping ECC accelerator.
[0]	**START**	**ECC Accelerator Start** 0 = No effect. 1 = Start ECC accelerator. BUSY flag will be set. This bit is always 0 when it is read back. ECC accelerator will ignore this START signal when BUSY flag is 1.

ECC Status Register (CRYPTO_ECC_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_STS	CRYPTO_BA+0x804	R	ECC Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					DFAERR	KSERR	BUSERR
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						DMABUSY	BUSY

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**DFAERR**	**ECC Engine Differential Fault Attack Error Flag** 0 = No error. 1 = Differential Fault Attack happened in ECC engine. The results from ECC engine are wrong.
[17]	**KSERR**	**ECC Engine Access Key Store Error Flag** 0 = No error. 1 = Access error will stop ECC engine.
[16]	**BUSERR**	**ECC DMA Access Bus Error Flag** 0 = No error. 1 = Bus error will stop DMA operation and ECC accelerator.
[15:2]	**Reserved**	Reserved.
[1]	**DMABUSY**	**ECC DMA Busy Flag** 0 = ECC DMA is idle or finished. 1 = ECC DMA is busy.
[0]	**BUSY**	**ECC Accelerator Busy Flag** 0 = The ECC accelerator is idle or finished. 1 = The ECC accelerator is under processing and protects all registers. **Note:** Remember to clear ECC interrupt flag after ECC accelerator is finished

ECC the X-coordinate Value of the First Point Register (CRYPTO_ECC_X1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_X1_00	CRYPTO_BA+0x808	R/W	ECC the X-coordinate Word0 of the First Point	0x0000_0000
CRYPTO_ECC_X1_01	CRYPTO_BA+0x80C	R/W	ECC the X-coordinate Word1 of the First Point	0x0000_0000
CRYPTO_ECC_X1_02	CRYPTO_BA+0x810	R/W	ECC the X-coordinate Word2 of the First Point	0x0000_0000
CRYPTO_ECC_X1_03	CRYPTO_BA+0x814	R/W	ECC the X-coordinate Word3 of the First Point	0x0000_0000
CRYPTO_ECC_X1_04	CRYPTO_BA+0x818	R/W	ECC the X-coordinate Word4 of the First Point	0x0000_0000
CRYPTO_ECC_X1_05	CRYPTO_BA+0x81C	R/W	ECC the X-coordinate Word5 of the First Point	0x0000_0000
CRYPTO_ECC_X1_06	CRYPTO_BA+0x820	R/W	ECC the X-coordinate Word6 of the First Point	0x0000_0000
CRYPTO_ECC_X1_07	CRYPTO_BA+0x824	R/W	ECC the X-coordinate Word7 of the First Point	0x0000_0000
CRYPTO_ECC_X1_08	CRYPTO_BA+0x828	R/W	ECC the X-coordinate Word8 of the First Point	0x0000_0000
CRYPTO_ECC_X1_09	CRYPTO_BA+0x82C	R/W	ECC the X-coordinate Word9 of the First Point	0x0000_0000
CRYPTO_ECC_X1_10	CRYPTO_BA+0x830	R/W	ECC the X-coordinate Word10 of the First Point	0x0000_0000
CRYPTO_ECC_X1_11	CRYPTO_BA+0x834	R/W	ECC the X-coordinate Word11 of the First Point	0x0000_0000
CRYPTO_ECC_X1_12	CRYPTO_BA+0x838	R/W	ECC the X-coordinate Word12 of the First Point	0x0000_0000
CRYPTO_ECC_X1_13	CRYPTO_BA+0x83C	R/W	ECC the X-coordinate Word13 of the First Point	0x0000_0000
CRYPTO_ECC_X1_14	CRYPTO_BA+0x840	R/W	ECC the X-coordinate Word14 of the First Point	0x0000_0000
CRYPTO_ECC_X1_15	CRYPTO_BA+0x844	R/W	ECC the X-coordinate Word15 of the First Point	0x0000_0000
CRYPTO_ECC_X1_16	CRYPTO_BA+0x848	R/W	ECC the X-coordinate Word16 of the First Point	0x0000_0000
CRYPTO_ECC_X1_17	CRYPTO_BA+0x84C	R/W	ECC the X-coordinate Word17 of the First Point	0x0000_0000

31	30	29	28	27	26	25	24
POINTX1							
23	22	21	20	19	18	17	16
POINTX1							
15	14	13	12	11	10	9	8
POINTX1							
7	6	5	4	3	2	1	0
POINTX1							

Bits	Description

[31:0]	**POINTX1**	**ECC the X-coordinate Value of the First Point** For B-163 or K-163, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_05 For B-233 or K-233, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_07 For B-283 or K-283, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_08 For B-409 or K-409, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_12 For B-571 or K-571, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_17 For P-192, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_05 For P-224, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_06 For P-256 or SM2, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_07 For P-384, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_11 For P-521, POINTX1 is stored in CRYPTO_ECC_X1_00~CRYPTO_ECC_X1_16

ECC the Y-coordinate Value of the First Point Register (CRYPTO_ECC_Y1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_Y1_00	CRYPTO_BA+0x850	R/W	ECC the Y-coordinate Word0 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_01	CRYPTO_BA+0x854	R/W	ECC the Y-coordinate Word1 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_02	CRYPTO_BA+0x858	R/W	ECC the Y-coordinate Word2 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_03	CRYPTO_BA+0x85C	R/W	ECC the Y-coordinate Word3 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_04	CRYPTO_BA+0x860	R/W	ECC the Y-coordinate Word4 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_05	CRYPTO_BA+0x864	R/W	ECC the Y-coordinate Word5 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_06	CRYPTO_BA+0x868	R/W	ECC the Y-coordinate Word6 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_07	CRYPTO_BA+0x86C	R/W	ECC the Y-coordinate Word7 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_08	CRYPTO_BA+0x870	R/W	ECC the Y-coordinate Word8 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_09	CRYPTO_BA+0x874	R/W	ECC the Y-coordinate Word9 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_10	CRYPTO_BA+0x878	R/W	ECC the Y-coordinate Word10 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_11	CRYPTO_BA+0x87C	R/W	ECC the Y-coordinate Word11 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_12	CRYPTO_BA+0x880	R/W	ECC the Y-coordinate Word12 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_13	CRYPTO_BA+0x884	R/W	ECC the Y-coordinate Word13 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_14	CRYPTO_BA+0x888	R/W	ECC the Y-coordinate Word14 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_15	CRYPTO_BA+0x88C	R/W	ECC the Y-coordinate Word15 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_16	CRYPTO_BA+0x890	R/W	ECC the Y-coordinate Word16 of the First Point	0x0000_0000
CRYPTO_ECC_Y1_17	CRYPTO_BA+0x894	R/W	ECC the Y-coordinate Word17 of the First Point	0x0000_0000

31	30	29	28	27	26	25	24
POINTY1							
23	22	21	20	19	18	17	16
POINTY1							
15	14	13	12	11	10	9	8
POINTY1							
7	6	5	4	3	2	1	0
POINTY1							

Bits	Description

[31:0]	**POINTY1**	**ECC the Y-coordinate Value of the First Point** For B-163 or K-163, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_05 For B-233 or K-233, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_07 For B-283 or K-283, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_08 For B-409 or K-409, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_12 For B-571 or K-571, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_17 For P-192, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_05 For P-224, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_06 For P-256 or SM2, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_07 For P-384, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_11 For P-521, POINTY1 is stored in CRYPTO_ECC_Y1_00~CRYPTO_ECC_Y1_16

ECC the X-coordinate Value of the Second Point Register (CRYPTO_ECC_X2)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_X2_00	CRYPTO_BA+0x898	R/W	ECC the X-coordinate Word0 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_01	CRYPTO_BA+0x89C	R/W	ECC the X-coordinate Word1 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_02	CRYPTO_BA+0x8A0	R/W	ECC the X-coordinate Word2 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_03	CRYPTO_BA+0x8A4	R/W	ECC the X-coordinate Word3 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_04	CRYPTO_BA+0x8A8	R/W	ECC the X-coordinate Word4 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_05	CRYPTO_BA+0x8AC	R/W	ECC the X-coordinate Word5 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_06	CRYPTO_BA+0x8B0	R/W	ECC the X-coordinate Word6 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_07	CRYPTO_BA+0x8B4	R/W	ECC the X-coordinate Word7 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_08	CRYPTO_BA+0x8B8	R/W	ECC the X-coordinate Word8 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_09	CRYPTO_BA+0x8BC	R/W	ECC the X-coordinate Word9 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_10	CRYPTO_BA+0x8C0	R/W	ECC the X-coordinate Word10 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_11	CRYPTO_BA+0x8C4	R/W	ECC the X-coordinate Word11 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_12	CRYPTO_BA+0x8C8	R/W	ECC the X-coordinate Word12 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_13	CRYPTO_BA+0x8CC	R/W	ECC the X-coordinate Word13 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_14	CRYPTO_BA+0x8D0	R/W	ECC the X-coordinate Word14 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_15	CRYPTO_BA+0x8D4	R/W	ECC the X-coordinate Word15 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_16	CRYPTO_BA+0x8D8	R/W	ECC the X-coordinate Word16 of the Second Point	0x0000_0000
CRYPTO_ECC_X2_17	CRYPTO_BA+0x8DC	R/W	ECC the X-coordinate Word17 of the Second Point	0x0000_0000

31	30	29	28	27	26	25	24
POINTX2							
23	22	21	20	19	18	17	16
POINTX2							
15	14	13	12	11	10	9	8
POINTX2							
7	6	5	4	3	2	1	0
POINTX2							

Bits	Description

[31:0]	**POINTX2**	**ECC the X-coordinate Value of the Second Point** For B-163 or K-163, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_05 For B-233 or K-233, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_07 For B-283 or K-283, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_08 For B-409 or K-409, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_12 For B-571 or K-571, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_17 For P-192, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_05 For P-224, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_06 For P-256 or SM2, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_07 For P-384, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_11 For P-521, POINTX2 is stored in CRYPTO_ECC_X2_00~CRYPTO_ECC_X2_16

ECC the Y-coordinate Value of the Second Point Register (CRYPTO_ECC_Y2)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_Y2_00	CRYPTO_BA+0x8E0	R/W	ECC the Y-coordinate Word0 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_01	CRYPTO_BA+0x8E4	R/W	ECC the Y-coordinate Word1 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_02	CRYPTO_BA+0x8E8	R/W	ECC the Y-coordinate Word2 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_03	CRYPTO_BA+0x8EC	R/W	ECC the Y-coordinate Word3 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_04	CRYPTO_BA+0x8F0	R/W	ECC the Y-coordinate Word4 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_05	CRYPTO_BA+0x8F4	R/W	ECC the Y-coordinate Word5 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_06	CRYPTO_BA+0x8F8	R/W	ECC the Y-coordinate Word6 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_07	CRYPTO_BA+0x8FC	R/W	ECC the Y-coordinate Word7 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_08	CRYPTO_BA+0x900	R/W	ECC the Y-coordinate Word8 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_09	CRYPTO_BA+0x904	R/W	ECC the Y-coordinate Word9 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_10	CRYPTO_BA+0x908	R/W	ECC the Y-coordinate Word10 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_11	CRYPTO_BA+0x90C	R/W	ECC the Y-coordinate Word11 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_12	CRYPTO_BA+0x910	R/W	ECC the Y-coordinate Word12 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_13	CRYPTO_BA+0x914	R/W	ECC the Y-coordinate Word13 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_14	CRYPTO_BA+0x918	R/W	ECC the Y-coordinate Word14 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_15	CRYPTO_BA+0x91C	R/W	ECC the Y-coordinate Word15 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_16	CRYPTO_BA+0x920	R/W	ECC the Y-coordinate Word16 of the Second Point	0x0000_0000
CRYPTO_ECC_Y2_17	CRYPTO_BA+0x924	R/W	ECC the Y-coordinate Word17 of the Second Point	0x0000_0000

31	30	29	28	27	26	25	24
POINTY2							
23	22	21	20	19	18	17	16
POINTY2							
15	14	13	12	11	10	9	8
POINTY2							
7	6	5	4	3	2	1	0
POINTY2							

Bits	Description

[31:0]	**POINTY2**	**ECC the Y-coordinate Value of the Second Point** For B-163 or K-163, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_05 For B-233 or K-233, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_07 For B-283 or K-283, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_08 For B-409 or K-409, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_12 For B-571 or K-571, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_17 For P-192, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_05 For P-224, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_06 For P-256 or SM2, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_07 For P-384, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_11 For P-521, POINTY2 is stored in CRYPTO_ECC_Y2_00~CRYPTO_ECC_Y2_16

ECC the Parameter CURVEA Value of Elliptic Curve Register (CRYPTO_ECC_A)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_A_00	CRYPTO_BA+0x928	R/W	ECC the Parameter CURVEA Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_01	CRYPTO_BA+0x92C	R/W	ECC the Parameter CURVEA Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_02	CRYPTO_BA+0x930	R/W	ECC the Parameter CURVEA Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_03	CRYPTO_BA+0x934	R/W	ECC the Parameter CURVEA Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_04	CRYPTO_BA+0x938	R/W	ECC the Parameter CURVEA Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_05	CRYPTO_BA+0x93C	R/W	ECC the Parameter CURVEA Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_06	CRYPTO_BA+0x940	R/W	ECC the Parameter CURVEA Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_07	CRYPTO_BA+0x944	R/W	ECC the Parameter CURVEA Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_08	CRYPTO_BA+0x948	R/W	ECC the Parameter CURVEA Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_09	CRYPTO_BA+0x94C	R/W	ECC the Parameter CURVEA Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_10	CRYPTO_BA+0x950	R/W	ECC the Parameter CURVEA Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_11	CRYPTO_BA+0x954	R/W	ECC the Parameter CURVEA Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_12	CRYPTO_BA+0x958	R/W	ECC the Parameter CURVEA Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_13	CRYPTO_BA+0x95C	R/W	ECC the Parameter CURVEA Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_14	CRYPTO_BA+0x960	R/W	ECC the Parameter CURVEA Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_15	CRYPTO_BA+0x964	R/W	ECC the Parameter CURVEA Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_16	CRYPTO_BA+0x968	R/W	ECC the Parameter CURVEA Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_A_17	CRYPTO_BA+0x96C	R/W	ECC the Parameter CURVEA Word17 of Elliptic Curve	0x0000_0000

31	30	29	28	27	26	25	24
CURVEA							
23	22	21	20	19	18	17	16
CURVEA							
15	14	13	12	11	10	9	8
CURVEA							
7	6	5	4	3	2	1	0
CURVEA							

Bits	Description

[31:0]	**CURVEA**	**ECC the Parameter CURVEA Value of Elliptic Curve** The formula of elliptic curve is $y^2=x^3+CURVEA*x+CURVEB$ in GF(p) and $y^2+x*y=x^3+CURVEA*x^2+CURVEB$ in GF(2^m). For B-163 or K-163, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_05 For B-233 or K-233, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_07 For B-283 or K-283, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_08 For B-409 or K-409, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_12 For B-571 or K-571, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_17 For P-192, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_05 For P-224, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_06 For P-256 or SM2, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_07 For P-384, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_11 For P-521, CURVEA is stored in CRYPTO_ECC_A_00~CRYPTO_ECC_A_16

ECC the Parameter CURVEB Value of Elliptic Curve Register (CRYPTO_ECC_B)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_B_00	CRYPTO_BA+0x970	R/W	ECC the Parameter CURVEB Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_01	CRYPTO_BA+0x974	R/W	ECC the Parameter CURVEB Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_02	CRYPTO_BA+0x978	R/W	ECC the Parameter CURVEB Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_03	CRYPTO_BA+0x97C	R/W	ECC the Parameter CURVEB Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_04	CRYPTO_BA+0x980	R/W	ECC the Parameter CURVEB Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_05	CRYPTO_BA+0x984	R/W	ECC the Parameter CURVEB Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_06	CRYPTO_BA+0x988	R/W	ECC the Parameter CURVEB Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_07	CRYPTO_BA+0x98C	R/W	ECC the Parameter CURVEB Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_08	CRYPTO_BA+0x990	R/W	ECC the Parameter CURVEB Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_09	CRYPTO_BA+0x994	R/W	ECC the Parameter CURVEB Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_10	CRYPTO_BA+0x998	R/W	ECC the Parameter CURVEB Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_11	CRYPTO_BA+0x99C	R/W	ECC the Parameter CURVEB Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_12	CRYPTO_BA+0x9A0	R/W	ECC the Parameter CURVEB Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_13	CRYPTO_BA+0x9A4	R/W	ECC the Parameter CURVEB Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_14	CRYPTO_BA+0x9A8	R/W	ECC the Parameter CURVEB Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_15	CRYPTO_BA+0x9AC	R/W	ECC the Parameter CURVEB Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_16	CRYPTO_BA+0x9B0	R/W	ECC the Parameter CURVEB Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_B_17	CRYPTO_BA+0x9B4	R/W	ECC the Parameter CURVEB Word17 of Elliptic Curve	0x0000_0000

31	30	29	28	27	26	25	24
CURVEB							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CURVEB							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CURVEB							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURVEB							

Bits	Description

[31:0]	**CURVEB**	**ECC the Parameter CURVEB Value of Elliptic Curve** The formula of elliptic curve is $y^2=x^3+CURVEA*x+CURVEB$ in GF(p) and $y^2+x*y=x^3+CURVEA*x^2+CURVEB$ in GF(2^m). For B-163 or K-163, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_05 For B-233 or K-233, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_07 For B-283 or K-283, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_08 For B-409 or K-409, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_12 For B-521 or K-521, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_17 For P-192, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_05 For P-224, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_06 For P-256 or SM2, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_07 For P-384, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_11 For P-521, CURVEB is stored in CRYPTO_ECC_B_00~CRYPTO_ECC_B_16

ECC the Parameter CURVEN Value of Elliptic Curve Register (CRYPTO_ECC_N)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_N_00	CRYPTO_BA+0x9B8	R/W	ECC the Parameter CURVEN Word0 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_01	CRYPTO_BA+0x9BC	R/W	ECC the Parameter CURVEN Word1 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_02	CRYPTO_BA+0x9C0	R/W	ECC the Parameter CURVEN Word2 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_03	CRYPTO_BA+0x9C4	R/W	ECC the Parameter CURVEN Word3 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_04	CRYPTO_BA+0x9C8	R/W	ECC the Parameter CURVEN Word4 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_05	CRYPTO_BA+0x9CC	R/W	ECC the Parameter CURVEN Word5 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_06	CRYPTO_BA+0x9D0	R/W	ECC the Parameter CURVEN Word6 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_07	CRYPTO_BA+0x9D4	R/W	ECC the Parameter CURVEN Word7 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_08	CRYPTO_BA+0x9D8	R/W	ECC the Parameter CURVEN Word8 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_09	CRYPTO_BA+0x9DC	R/W	ECC the Parameter CURVEN Word9 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_10	CRYPTO_BA+0x9E0	R/W	ECC the Parameter CURVEN Word10 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_11	CRYPTO_BA+0x9E4	R/W	ECC the Parameter CURVEN Word11 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_12	CRYPTO_BA+0x9E8	R/W	ECC the Parameter CURVEN Word12 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_13	CRYPTO_BA+0x9EC	R/W	ECC the Parameter CURVEN Word13 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_14	CRYPTO_BA+0x9F0	R/W	ECC the Parameter CURVEN Word14 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_15	CRYPTO_BA+0x9F4	R/W	ECC the Parameter CURVEN Word15 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_16	CRYPTO_BA+0x9F8	R/W	ECC the Parameter CURVEN Word16 of Elliptic Curve	0x0000_0000
CRYPTO_ECC_N_17	CRYPTO_BA+0x9FC	R/W	ECC the Parameter CURVEN Word17 of Elliptic Curve	0x0000_0000

31	30	29	28	27	26	25	24
CURVEN							
23	22	21	20	19	18	17	16
CURVEN							
15	14	13	12	11	10	9	8
CURVEN							
7	6	5	4	3	2	1	0
CURVEN							

Bits	Description

[31:0]	**CURVEN**	**ECC the Parameter CURVEN Value of Elliptic Curve** In GF(p), CURVEN is the prime p. In GF(2^m), CURVEN is the irreducible polynomial. For B-163 or K-163, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_05 For B-233 or K-233, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_07 For B-283 or K-283, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_08 For B-409 or K-409, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_12 For B-571 or K-571, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_17 For P-192, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_05 For P-224, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_06 For P-256 or SM2, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_07 For P-384, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_11 For P-521, CURVEN is stored in CRYPTO_ECC_N_00~CRYPTO_ECC_N_16

ECC the Scalar K Value of Elliptic Curve Register (CRYPTO_ECC_K)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_K_00	CRYPTO_BA+0xA00	W	ECC the Scalar SCALARK Word0 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_01	CRYPTO_BA+0xA04	W	ECC the Scalar SCALARK Word1 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_02	CRYPTO_BA+0xA08	W	ECC the Scalar SCALARK Word2 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_03	CRYPTO_BA+0xA0C	W	ECC the Scalar SCALARK Word3 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_04	CRYPTO_BA+0xA10	W	ECC the Scalar SCALARK Word4 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_05	CRYPTO_BA+0xA14	W	ECC the Scalar SCALARK Word5 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_06	CRYPTO_BA+0xA18	W	ECC the Scalar SCALARK Word6 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_07	CRYPTO_BA+0xA1C	W	ECC the Scalar SCALARK Word7 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_08	CRYPTO_BA+0xA20	W	ECC the Scalar SCALARK Word8 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_09	CRYPTO_BA+0xA24	W	ECC the Scalar SCALARK Word9 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_10	CRYPTO_BA+0xA28	W	ECC the Scalar SCALARK Word10 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_11	CRYPTO_BA+0xA2C	W	ECC the Scalar SCALARK Word11 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_12	CRYPTO_BA+0xA30	W	ECC the Scalar SCALARK Word12 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_13	CRYPTO_BA+0xA34	W	ECC the Scalar SCALARK Word13 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_14	CRYPTO_BA+0xA38	W	ECC the Scalar SCALARK Word14 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_15	CRYPTO_BA+0xA3C	W	ECC the Scalar SCALARK Word15 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_16	CRYPTO_BA+0xA40	W	ECC the Scalar SCALARK Word16 of Point Multiplication	0x0000_0000
CRYPTO_ECC_K_17	CRYPTO_BA+0xA44	W	ECC the Scalar SCALARK Word17 of Point Multiplication	0x0000_0000

31	30	29	28	27	26	25	24
SCALARK							
23	22	21	20	19	18	17	16
SCALARK							
15	14	13	12	11	10	9	8
SCALARK							
7	6	5	4	3	2	1	0
SCALARK							

Bits	Description

[31:0]	**SCALARK**	**ECC the Scalar SCALARK Value of Point Multiplication** Because the SCALARK usually stores the private key, ECC accelerator do not allow to read the register SCALARK. For B-163 or K-163, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_05 For B-233 or K-233, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_07 For B-283 or K-283, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_08 For B-409 or K-409, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_12 For B-571 or K-571, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_17 For P-192, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_05 For P-224, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_06 For P-256 or SM2, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_07 For P-384, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_11 For P-521, SCALARK is stored in CRYPTO_ECC_K_00~CRYPTO_ECC_K_16

ECC DMA Source Address Register (CRYPTO_ECC_SADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_SADDR	CRYPTO_BA+0xA48	R/W	ECC DMA Source Address Register	0x0000_0000

31	30	29	28	27	26	25	24
SADDR							
23	22	21	20	19	18	17	16
SADDR							
15	14	13	12	11	10	9	8
SADDR							
7	6	5	4	3	2	1	0
SADDR							

Bits	Description	
[31:0]	Reserved	**ECC DMA Source Address** The ECC accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and ECC accelerator. The SADDR keeps the source address of the data buffer where the source text is stored. Based on the source address, the ECC accelerator can read the DATA and PARAMETER from SRAM memory space and do ECC operation. The start of source address should be located at word boundary. That is, bit 1 and 0 of SADDR are ignored. SADDR can be read and written. In DMA mode, software must update the CRYPTO_ECC_SADDR before triggering START.

ECC DMA Destination Address Register (CRYPTO_ECC_DADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_DADDR	CRYPTO_BA+0xA4C	R/W	ECC DMA Destination Address Register	0x0000_0000

31	30	29	28	27	26	25	24
DADDR							
23	22	21	20	19	18	17	16
DADDR							
15	14	13	12	11	10	9	8
DADDR							
7	6	5	4	3	2	1	0
DADDR							

Bits	Description	
[31:0]	DADDR	**ECC DMA Destination Address** The ECC accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory and ECC accelerator. The DADDR keeps the destination address of the data buffer where output data of ECC engine will be stored. Based on the destination address, the ECC accelerator can write the result data back to SRAM memory space after the ECC operation is finished. The start of destination address should be located at word boundary. That is, bit 1 and 0 of DADDR are ignored. DADDR can be read and written. In DMA mode, software must update the CRYPTO_ECC_DADDR before triggering START.

ECC Starting Address of Updated Registers (CRYPTO_ECC_STARTREG)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_STARTREG	CRYPTO_BA+0xA50	R/W	ECC Starting Address of Updated Registers	0x0000_0000

31	30	29	28	27	26	25	24
STARTREG							
23	22	21	20	19	18	17	16
STARTREG							
15	14	13	12	11	10	9	8
STARTREG							
7	6	5	4	3	2	1	0
STARTREG							

Bits	Description	
[31:0]	STARTREG	**ECC Starting Address of Updated Registers** The address of the updated registers that DMA feeds the first data or parameter to ECC engine. When ECC engine is active, ECC accelerator does not allow users to modify STARTRE.G. For example, to update input data from register CRYPTO_ECC POINTX1. Thus, the value of STARTREG is 0x808.

ECC DMA Word Count (CRYPTO_ECC_WORDCNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_WORDCNT	CRYPTO_BA+0xA54	R/W	ECC DMA Word Count	0x0000_0000

31	30	29	28	27	26	25	24
WORDCNT							
23	22	21	20	19	18	17	16
WORDCNT							
15	14	13	12	11	10	9	8
WORDCNT							
7	6	5	4	3	2	1	0
WORDCNT							

Bits	Description	
[31:0]	WORDCNT	**ECC DMA Word Count** The CRYPTO_ECC_WORDCNT keeps the word count of source data that is for the required input data of ECC accelerator with various operations in DMA mode. Although CRYPTO_ECC_WORDCNT is 32-bit, the maximum of word count in ECC accelerator is 144 words. CRYPTO_ECC_WORDCNT can be read and written.

6.40.7.6 RSA Register

RSA Control Register (CRYPTO_RSA_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_CTL	CRYPTO_BA+0xB00	R/W	RSA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							SCAP
7	6	5	4	3	2	1	0
Reserved		KEYLENG		CRTBYP	CRT	STOP	START

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**SCAP**	**Side Channel Attack Protection Enable Control** 0 = Side Channel Attack Protection Disabled. 1 = Side Channel Attack Protection Enabled.
[7:6]	**Reserved**	Reserved.
[5:4]	**KEYLENG**	**The Key Length of RSA Operation** 00 = 1024-bits. 01 = 2048-bits. 10 = 3072-bits. 11 = 4096-bits.
[3]	**CRTBYP**	**CRT Bypass Enable Control** 0 = CRT Bypass Disabled. 1 = CRT Bypass Enabled. CRT bypass is only used in CRT decryption with the same key. **Note:** If users want to decrypt repeatedly with the same key, they can execute CRT bypass mode **after the first time CRT decryption (means the second time to the latest time)**, but they cannot set CRTBYP to 1 in non-CRT mode.
[2]	**CRT**	**CRT Enable Control** 0 = CRT Disabled. 1 = CRT Enabled. CRT is only used in decryption with key length 2048, 3072,4096 bits.
[1]	**STOP**	**RSA Accelerator Stop** 0 = No effect. 1 = Abort RSA accelerator and make it into initial state. This bit is always 0 when it is read back. Remember to clear RSA interrupt flag after stopping RSA accelerator.

[0]	**START**	**RSA Accelerator Start** 0 = No effect. 1 = Start RSA accelerator. BUSY flag will be set. This bit is always 0 when it is read back. RSA accelerator will ignore this START signal when BUSY flag is 1.

RSA Status Register (CRYPTO_RSA_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_STS	CRYPTO_BA+0xB04	R	RSA Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					KSERR	CTLERR	BUSERR
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						DMABUSY	BUSY

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**KSERR**	**RSA Engine Access Key Store Error Flag** 0 = No error. 1 = Access error will stop RSA engine.
[17]	**CTLERR**	**RSA Control Register Error Flag** 0 = No error. 1 = RSA control error. RSA will not start in the unsupported situation. **Note:** If users use the error combination of control, even though they don't set START(CRYPTO_RSA_CTL[0]) to 1, CTLERR will still be set to 1.
[16]	**BUSERR**	**RSA DMA Access Bus Error Flag** 0 = No error. 1 = Bus error will stop DMA operation and RSA accelerator.
[15:2]	**Reserved**	Reserved.
[1]	**DMABUSY**	**RSA DMA Busy Flag** 0 = RSA DMA is idle or finished. 1 = RSA DMA is busy.
[0]	**BUSY**	**RSA Accelerator Busy Flag** 0 = The RSA accelerator is idle or finished. 1 = The RSA accelerator is under processing and protects all registers. Remember to clear RSA interrupt flag after RSA accelerator finished.

RSA DMA Source Address Register0 (CRYPTO_RSA_SADDR0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR0	CRYPTO_BA+0xB08	R/W	RSA DMA Source Address Register0	0x0000_0000

31	30	29	28	27	26	25	24
SADDR0							
23	22	21	20	19	18	17	16
SADDR0							
15	14	13	12	11	10	9	8
SADDR0							
7	6	5	4	3	2	1	0
SADDR0							

Bits	Description	
[31:0]	SADDR0	**RSA DMA Source Address Register0** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored the address of RSA the Base of Exponentiation (M).

RSA DMA Source Address Register1 (CRYPTO_RSA_SADDR1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR1	CRYPTO_BA+0xB0C	R/W	RSA DMA Source Address Register1	0x0000_0000

31	30	29	28	27	26	25	24
SADDR1							
23	22	21	20	19	18	17	16
SADDR1							
15	14	13	12	11	10	9	8
SADDR1							
7	6	5	4	3	2	1	0
SADDR1							

Bits	Description	
[31:0]	SADDR1	**RSA DMA Source Address Register1** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored the address of RSA the Base of Modulus Operation (N).

RSA DMA Source Address Register2 (CRYPTO_RSA_SADDR2)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR2	CRYPTO_BA+0xB10	R/W	RSA DMA Source Address Register2	0x0000_0000

31	30	29	28	27	26	25	24
SADDR2							
23	22	21	20	19	18	17	16
SADDR2							
15	14	13	12	11	10	9	8
SADDR2							
7	6	5	4	3	2	1	0
SADDR2							

Bits	Description	
[31:0]	SADDR2	**RSA DMA Source Address Register2** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored the address of RSA the Exponent of Exponentiation (E).

RSA DMA Source Address Register3 (CRYPTO_RSA_SADDR3)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR3	CRYPTO_BA+0xB14	R/W	RSA DMA Source Address Register3	0x0000_0000

31	30	29	28	27	26	25	24
SADDR3							
23	22	21	20	19	18	17	16
SADDR3							
15	14	13	12	11	10	9	8
SADDR3							
7	6	5	4	3	2	1	0
SADDR3							

Bits	Description	
[31:0]	SADDR3	**RSA DMA Source Address Register3** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored the address of RSA the Factor of Modulus Operation (p).

RSA DMA Source Address Register4 (CRYPTO_RSA_SADDR4)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_SADDR4	CRYPTO_BA+0xB18	R/W	RSA DMA Source Address Register4	0x0000_0000

31	30	29	28	27	26	25	24
SADDR4							
23	22	21	20	19	18	17	16
SADDR4							
15	14	13	12	11	10	9	8
SADDR4							
7	6	5	4	3	2	1	0
SADDR4							

Bits	Description	
[31:0]	SADDR4	**RSA DMA Source Address Register4** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored the address of RSA the Factor of Modulus Operation (q).

RSA DMA Destination Address Register (CRYPTO_RSA_DADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_DADDR	CRYPTO_BA+0xB1C	R/W	RSA DMA Destination Address Register	0x0000_0000

31	30	29	28	27	26	25	24
DADDR							
23	22	21	20	19	18	17	16
DADDR							
15	14	13	12	11	10	9	8
DADDR							
7	6	5	4	3	2	1	0
DADDR							

Bits	Description	
[31:0]	DADDR	**RSA DMA Destination Address Register** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator. This register is stored the address of RSA DMA Destination Address Register (Ans).

RSA DMA Middle Address Register0 (CRYPTO_RSA_MADDR0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR0	CRYPTO_BA+0xB20	R/W	RSA DMA Middle Address Register0	0x0000_0000

31	30	29	28	27	26	25	24
MADDR0							
23	22	21	20	19	18	17	16
MADDR0							
15	14	13	12	11	10	9	8
MADDR0							
7	6	5	4	3	2	1	0
MADDR0							

Bits	Description	
[31:0]	MADDR0	**RSA DMA Middle Address Register0** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register1 (CRYPTO_RSA_MADDR1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR1	CRYPTO_BA+0xB24	R/W	RSA DMA Middle Address Register1	0x0000_0000

31	30	29	28	27	26	25	24
MADDR1							
23	22	21	20	19	18	17	16
MADDR1							
15	14	13	12	11	10	9	8
MADDR1							
7	6	5	4	3	2	1	0
MADDR1							

Bits	Description	
[31:0]	MADDR1	**RSA DMA Middle Address Register1** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register2 (CRYPTO_RSA_MADDR2)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR2	CRYPTO_BA+0xB28	R/W	RSA DMA Middle Address Register2	0x0000_0000

31	30	29	28	27	26	25	24
MADDR2							
23	22	21	20	19	18	17	16
MADDR2							
15	14	13	12	11	10	9	8
MADDR2							
7	6	5	4	3	2	1	0
MADDR2							

Bits	Description	
[31:0]	MADDR2	**RSA DMA Middle Address Register2** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register3 (CRYPTO_RSA_MADDR3)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR3	CRYPTO_BA+0xB2C	R/W	RSA DMA Middle Address Register3	0x0000_0000

31	30	29	28	27	26	25	24
MADDR3							
23	22	21	20	19	18	17	16
MADDR3							
15	14	13	12	11	10	9	8
MADDR3							
7	6	5	4	3	2	1	0
MADDR3							

Bits	Description	
[31:0]	MADDR3	**RSA DMA Middle Address Register3** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register4 (CRYPTO_RSA_MADDR4)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR4	CRYPTO_BA+0xB30	R/W	RSA DMA Middle Address Register4	0x0000_0000

31	30	29	28	27	26	25	24
MADDR4							
23	22	21	20	19	18	17	16
MADDR4							
15	14	13	12	11	10	9	8
MADDR4							
7	6	5	4	3	2	1	0
MADDR4							

Bits	Description	
[31:0]	MADDR4	**RSA DMA Middle Address Register4** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register5 (CRYPTO_RSA_MADDR5)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR5	CRYPTO_BA+0xB34	R/W	RSA DMA Middle Address Register5	0x0000_0000

31	30	29	28	27	26	25	24
MADDR5							
23	22	21	20	19	18	17	16
MADDR5							
15	14	13	12	11	10	9	8
MADDR5							
7	6	5	4	3	2	1	0
MADDR5							

Bits	Description	
[31:0]	MADDR5	**RSA DMA Middle Address Register5** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

RSA DMA Middle Address Register6 (CRYPTO_RSA_MADDR6)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_MADDR6	CRYPTO_BA+0xB38	R/W	RSA DMA Middle Address Register6	0x0000_0000

31	30	29	28	27	26	25	24
MADDR6							
23	22	21	20	19	18	17	16
MADDR6							
15	14	13	12	11	10	9	8
MADDR6							
7	6	5	4	3	2	1	0
MADDR6							

Bits	Description	
[31:0]	MADDR6	**RSA DMA Middle Address Register6** The RSA accelerator supports DMA function to transfer the DATA and PARAMETER between SRAM memory space and RSA accelerator.

6.40.7.7 Key Control Register

PRNG Key Control Register (CRYPTO_PRNG_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KSCTL	CRYPTO_BA+0xF00	W	PRNG Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					OWNER		
23	22	21	20	19	18	17	16
WSDST		WDST	ECDSA	ECDH	Reserved		TRUST
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			NUM				

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:24]	**OWNER**	**Write Key Owner Selection Bits** 000 = Only for AES use. 001 = Only for HMAC engine use. 100 = Only for ECC engine use. 101 = Only for CPU engine use. Others = reserved.
[23:22]	**WSDST**	**Write Key Store Destination** 00 = Key is written to the SRAM of key store. 01 = Key is written to the Flash of key store. 10 = key is written to the OTP of key store. Others = reserved.
[21]	**WDST**	**Write Key Destination** 0 = Key is written to registers CRYPTO_PRNG_KEYx. 1 = Key is written to key store.
[20]	**ECDSA**	**ECDSA Control Bit** 0 = Reserved. 1 = Key is written to key store and used in ECDSA. **Note:** When ECDSA was set to '1', **1.** PRNG seed must from TRNG and key is must written to the SRAM of key store (WSDST, CRYPTO_PRNG_KSCTL[23:22] must set to '00'). Otherwise, KCTLERR will become '1'(CRYPTO_PRNG_KSSTS[16]). **2.** Key must in the interval [1, n-1] (the parameter n is from ECC). The value of n cannot be 0 or 1, otherwise, PRNG will always keep busy.

[19]	**ECDH**	**ECDH Control Bit** 0 = Reserved. 1 = Key is written to key store and used in ECDH. **Note:** When ECDH was set to '1', **1.** PRNG seed must from TRNG and key is must written to the SRAM of key store (WSDST, CRYPTO_PRNG_KSCTL[23:22] must set to '00'). Otherwise, KCTLERR will become '1'(CRYPTO_PRNG_KSSTS[16]). **2.** Key must in the interval [1, n-1] (the parameter n is from ECC). The value of n cannot be 0 or 1, otherwise, PRNG will always keep busy.
[18:17]	**Reserved**	Reserved.
[16]	**TRUST**	**Write Key Trust Selection Bit** 0 = Set written key as the non-secure key. 1 = Set written key as the secure key.
[15:5]	**Reserved**	Reserved.
[4:0]	**NUM**	**Write Key Number** The key number is sent to key store **Note:** Only for destination Is OTP of key store.

PRNG Key Status Register (CRYPTO_PRNG_KSSTS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KSSTS	CRYPTO_BA+0xF04	R	PRNG Key Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			NUM				

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**NUM**	**Key Number** The key number is generated by key store

AES Key Control Register (CRYPTO_AES_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_KSCTL	CRYPTO_BA+0xF10	W	AES Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
RSSRC		RSRC	NUM				

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**RSSRC**	**Read Key Store Source** 00 = Key is read from the SRAM of key store. 01 = Key is read from the Flash of key store. 10 = Key is read from the OTP of key store. Others = reserved.
[5]	**RSRC**	**Read Key Source** 0 = Key is read from registers CRYPTO_AESx_KEYx. 1 = Key is read from key store.
[4:0]	**NUM**	**Read Key Number** The key number is sent to key store

HMAC Key Control Register (CRYPTO_HMAC_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_HMAC_KSCTL	CRYPTO_BA+0xF30	W	HMAC Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
RSSRC		RSRC	NUM				

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**RSSRC**	**Read Key Store Source** 00 = Key is read from the SRAM of key store. 01 = Key is read from the Flash of key store. Others = reserved.
[5]	**RSRC**	**Read Key Source** 0 = Key is read from HMAC registers. 1 = Key is read from key store.
[4:0]	**NUM**	**Read Key Number** The key number is sent to key store

ECC Key Control Register (CRYPTO_ECC_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_KSCTL	CRYPTO_BA+0xF40	W	ECC Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					OWNER		
23	22	21	20	19	18	17	16
WSDST		WDST	XY	Reserved			TRUST
15	14	13	12	11	10	9	8
Reserved	ECDH	Reserved					
7	6	5	4	3	2	1	0
RSSRCK		RSRCK	NUMK				

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:24]	**OWNER**	**Write Key Owner Selection Bits** 000 = The ECDH written key is only for AES used. 001 = The ECDH written key is only for HMAC engine used. 100 = The ECDH written key is only for ECC engine used. 101 = The ECDH written key is only for CPU engine use. Others = reserved.
[23:22]	**WSDST**	**Write Key Store Destination** 00 = The ECDH written key is written to the SRAM of key store. 01 = The ECDH written key is written to the Flash of key store. 10 = The ECDH written key is written to the OTP of key store. Others = reserved.
[21]	**WDST**	**Write Key Destination** 0 = The ECDH written key is in registers CRYPTO_ECC_X1 and CRYPTO_ECC_Y. 1 = The ECDH written key is written to key store.
[20]	**XY**	**ECDH Output Select Bit** 0 = The ECDH written key is from X-coordinate Value. 1 = The ECDH written key is from Y-coordinate Value.
[19:17]	**Reserved**	Reserved.
[16]	**TRUST**	**Write Key Trust Selection Bit** 0 = Set ECDH written key as the non-secure key. 1 = Set ECDH written key as the secure key.
[15]	**Reserved**	Reserved.
[14]	**ECDH**	**ECDH Control Bit** 0 = Reserved. 1 = Set ECC operation is in ECDH. When this bit and RSRCK are equal to 0x1, ECC will read ECDH private key to CRYPTO_ECC_K from key store.

[13:8]	**Reserved**	Reserved.
[7:6]	**RSSRCK**	**Read Key Store Source for Key Number K** 00 = Key is read from the SRAM of key store. 01 = Key is read from the Flash of key store. 10 = Key is read from the OTP of key store. Others = reserved.
[5]	**RSRCK**	**Read Key Source for Key Number K** 0 = Key is read from ECC registers. 1 = Key is read from key store.
[4:0]	**NUMK**	**Read Key Number K** The key number of CRYPTO_ECC_K is sent to key store when RSRCK =1.

ECC Key Status Register (CRYPTO_ECC_KSSTS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_KSSTS	CRYPTO_BA+0xF44	R	ECC Key Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			NUM				

Bits	Description	
[31:5]	Reserved	Reserved.
[4:0]	NUM	**Key Number** The key number is generated by key store after ECDH.

ECC XY Number Register (CRYPTO_ECC_KSXY)

Register	Offset	R/W	Description	Reset Value
CRYPTO_ECC_KSXY	CRYPTO_BA+0xF48	W	ECC XY Number Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
RSSRCY		Reserved	NUMY				
7	6	5	4	3	2	1	0
RSSRCX		RSRCXY	NUMX				

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:14]	**RSSRCY**	**Read Key Store Source for Key Number Y** 00 = Key is read from the SRAM of key store. 01 = Key is read from the Flash of key store. 10 = Key is read from the OTP of key store. Others = reserved.
[13]	**Reserved**	Reserved.
[12:8]	**NUMY**	**Read Key Number Y** The key number of CRYPTO_ECC_Y1 is sent to key store when RSRCXY =1.
[7:6]	**RSSRCX**	**Read Key Store Source for Key Number X** 00 = Key is read from the SRAM of key store. 01 = Key is read from the Flash of key store. 10 = Key is read from the OTP of key store. Others = reserved.
[5]	**RSRCXY**	**Read Key Source for Key Number x and Y** 0 = Key is read from ECC registers. 1 = Key is read from key store.
[4:0]	**NUMX**	**Read Key Number X** The key number of CRYPTO_ECC_X1 is sent to key store when RSRCXY =1.

RSA Key Control Register (CRYPTO_RSA_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_KSCTL	CRYPTO_BA+0xF50	W	RSA Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			BKNUM				
7	6	5	4	3	2	1	0
RSSRC		RSRC	NUM				

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12:8]	**BKNUM**	**Read Exponent Blind Key Number** The key number is sent to key store, and its destination always be the SRAM of key store. CPU cannot read the exponent blind key. **Note:** Use this key number, only when executing SCA protection but no-CRT mode. When allocate space of key store, key owner selection bits(KS_METADATA[18:16]) should be '010'.
[7:6]	**RSSRC**	**Read Key Store Source** 00 = Key is read from the SRAM of key store. 01 = Key is read from the Flash of key store. Others = Reserved.
[5]	**RSRC**	**Read Key Source** 0 = Key is read from RSA engine. 1 = Key is read from key store.
[4:0]	**NUM**	**Read Key Number** The key number is sent to key store

RSA Key Status 0 Register (CRYPTO_RSA_KSSTS0)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_KSSTS0	CRYPTO_BA+0xF54	R/W	RSA Key Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			NUM3				
23	22	21	20	19	18	17	16
Reserved			NUM2				
15	14	13	12	11	10	9	8
Reserved			NUM1				
7	6	5	4	3	2	1	0
Reserved			NUM0				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**NUM3**	**Key Number3** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in the key store while operating. **Note:** The size of this key as key length.
[23:21]	**Reserved**	Reserved.
[20:16]	**NUM2**	**Key Number2** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in the key store while operating. **Note:** The size of this key as key length.
[15:13]	**Reserved**	Reserved.
[12:8]	**NUM1**	**Key Number1** The key number is generated by key store, RSA can get complete q by key number in Key Store while operating. **Note:** The size of this key as half key length.
[7:5]	**Reserved**	Reserved.
[4:0]	**NUM0**	**Key Number0** The key number is generated by key store, RSA can get complete p by key number in key store while operating. **Note:** The size of this key as half key length.

RSA Key Status 1 Register (CRYPTO_RSA_KSSTS1)

Register	Offset	R/W	Description	Reset Value
CRYPTO_RSA_KSSTS1	CRYPTO_BA+0xF58	R/W	RSA Key Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			NUM7				
23	22	21	20	19	18	17	16
Reserved			NUM6				
15	14	13	12	11	10	9	8
Reserved			NUM5				
7	6	5	4	3	2	1	0
Reserved			NUM4				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**NUM7**	**Key Number7** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in key store while operating. **Note:** The size of this key as key length.
[23:21]	**Reserved**	Reserved.
[20:16]	**NUM6**	**Key Number6** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in key store while operating. **Note:** The size of this key as key length.
[15:13]	**Reserved**	Reserved.
[12:8]	**NUM5**	**Key Number5** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in key store while operating. **Note:** The size of this key as half key length.
[7:5]	**Reserved**	Reserved.
[4:0]	**NUM4**	**Key Number4** The key number is generated by key store, RSA can get or store the intermediate temporary value by key number in key store while operating. **Note:** The size of this key as half key length.

6.41 Cryptographic Accelerator for M463 Series (CRYPTO_M463)

6.41.1 Overview

The Crypto (Cryptographic Accelerator) includes a secure pseudo random number generator (PRNG) core and supports AES algorithm.

The PRNG core supports 128, 192 and 256 bits random number generation.

The AES accelerator is an implementation fully compliant with the AES (Advance Encryption Standard) encryption and decryption algorithm. The AES accelerator supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2 and CBC-CS3 mode.

The Crypto can get key from key store and/or put the key to key store determined by the function of each accelerator.

6.41.2 Features

- PRNG
  - Supports 128, 192, and 256 bits random number generation.
  - Can take the true random number seed from TRNG
- AES
  - Supports FIPS NIST 197
  - Supports SP800-38A and addendum
  - Supports 128, 192, and 256 bits key
  - Supports both encryption and decryption
  - Supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2 and CBC-CS3 modes
  - Supports key expander for key expansion in FIPS NIST 197

6.41.3 Block Diagram

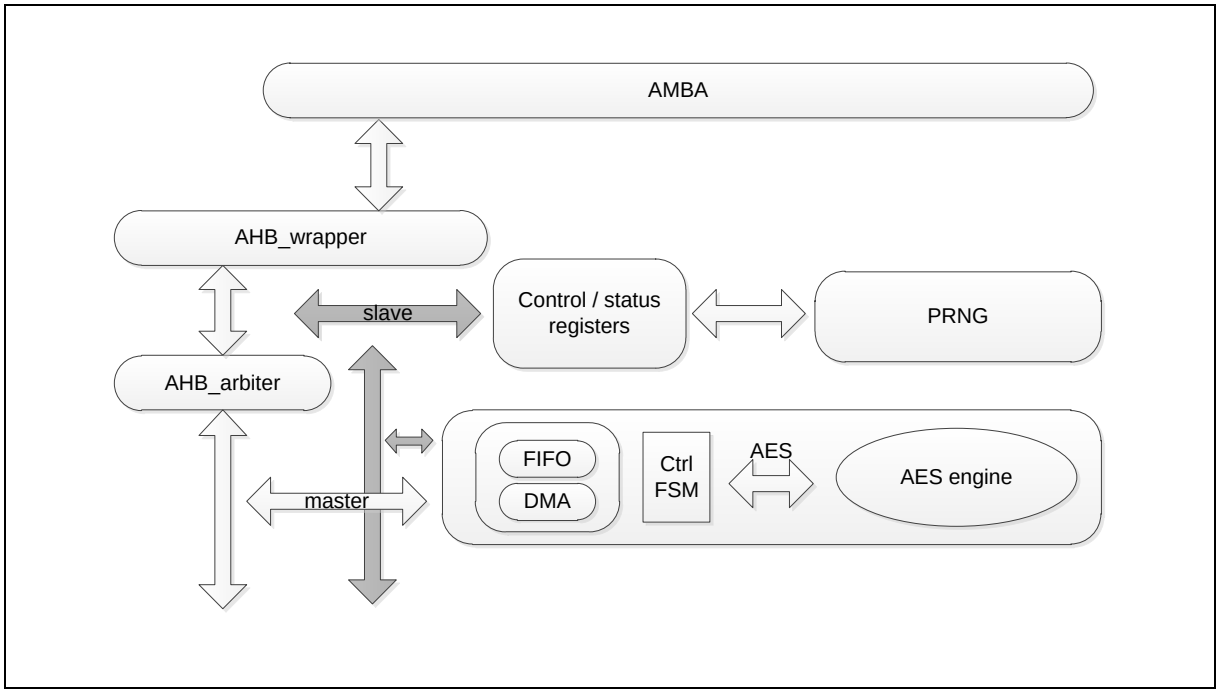


Figure 6.41-1 Cryptographic Accelerator Block Diagram

6.41.4 Basic Configuration

- Clock Source Configuration
  - Enable CRYPTO peripheral clock in CRPTCKEN(CLK_AHBCLK[12])
- Reset Configuration
  - Reset CRYPTO controller in CRPTRST(SYS_IPRST0[12])

6.41.5 Functional Description

The cryptographic accelerator includes a secure pseudo random number generator (PRNG) core and supports AES algorithm. The accelerator can be used in different data security applications, such as secure communications that need cryptographic protection and integrity.

1. The PRNG core supports 128, 192, and 256 bits random number generation configured by KEYSZ.
2. The AES accelerator is a fully compliant implementation of the AES (Advance Encryption Standard) encryption and decryption algorithm. The AES accelerator supports ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2 and CBC-CS3 mode. The AES accelerator provides the DMA function to reduce the CPU intervention, and supports three burst lengths, sixteen-words, eight-words, and four-words.

Software can control the data flow by enabling the CRYPTO_INTEN, and monitor the accelerator status by checking the CRYPTO_INTSTS status register. When any engine happened operation error or buffer error, the corresponding error flag will be set to 1 and inform to CPU if error interrupt enable bit is set to 1. If users want to detail error condition, software can check status flag register of each engine.

Engine	Error Interrupt Enable Bit	Error Interrupt Flag	Error Conditions
PRNG	PRNGEIEN (CRYPTO_INTEN[17])	PRNGEIF (CRYPTO_INTSTS[17])	KSERR/KCTLERR
AES	AESEIEN (CRYPTO_INTEN[1])	AESEIF (CRYPTO_INTSTS[1])	KSERR/INBUFERR/OUTBUFERR/BUSERR

Table 6.41-1 Each Engine Error Conditions and Error Flag

The cryptographic accelerator supports the following features to enhance the performance.

DMA Mode

Once DMA source address register, destination address register, and byte count register are configured by CPU, moving data from and to accelerator is done by DMA logic totally. This mode can off-load the loading from the CPU. The cryptographic accelerator embeds one hardware DMA channel for AES engine.

Engine	DMA Enable Bit
AES	DMAEN (CRYPTO_AES_CTL[7])

Table 6.41-2 DMA Enable Bit Table

DMA Cascade Mode

In the case that the data SRAM resource is tight, or another peripheral is scheduled to switch, the data source or sink needs an update, while the setting for the accelerator operation is planned to be kept. In this mode, software can update DMA source address register, destination address register, and byte count register during a cascade operation, without finishing the accelerator operation. AES provide FBOUT and FBIN bits in their control register to trigger DMA engine to automatically output and input all required information according to their DMA feedback address register (CRYPTO_AES_FBADDR). Please note that the last two blocks of AES-CBC-CS1/2/3 must be in the last cascade operation.

Engine	DMA Cascade Bit
AES	DMACSCAD (CRYPTO_AES_CTL[6])

Table 6.41-3 DMA Cascade Bit Table

Non-DMA Mode

In the case that the input data is small in size, DMA mode is not preferred. This mode can reduce the processing time for the accelerator, since no DMA related register needs a configuration, and no latency in DMA logic is introduced. Input data was feeding to cryptographic engine via writing to data input register.

Channel Expansion Mode

In this mode, several virtual channels are feasible in AES mode. The intermediate data (or called feedback information) from feedback registers (CRYPTO_AES_FDBCKx) should be stored temporarily in data SRAM and switched to another configuration setting of accelerator operation that includes operational mode, encryption/decryption, key, key size, IV, and other parameters. Once switching back, the intermediate data from feedback registers should be written to initial vectors (CRYPTO_AES_IVx) for the accelerator to continue the operation with the original configuration setting. Finally, user must configure AES control register (CRYPTO_AES_CTL) to complete accelerator operation. Note that, in ECB mode, there is no need to move the intermediate data from feedback registers to IV.

6.41.5.2 PRNG (Pseudo Random Number Generator)

The PRNG block diagram is depicted in Figure 6.41-2. The core supports 128, 192 and 256 bits random number generation configured by KEYSZ.

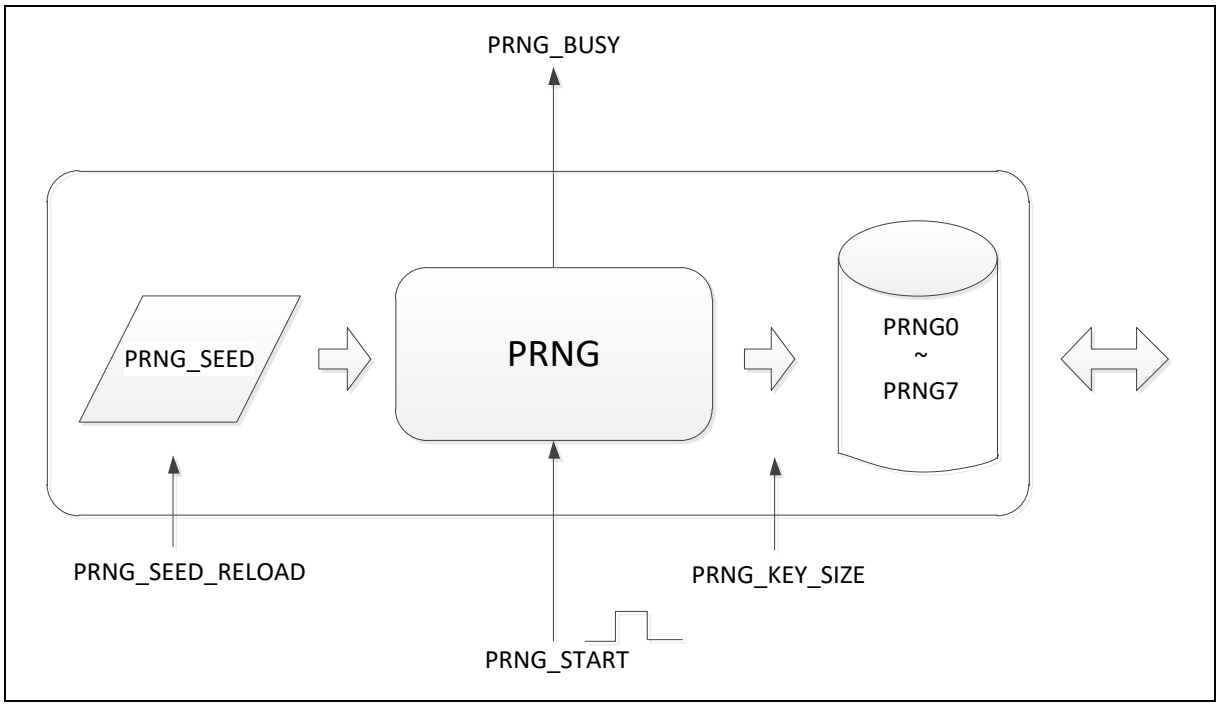


Figure 6.41-2 PRNG Function Diagram

Programming steps to get the **pseudo** random number are depicted below.

1. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
2. Write a random seed to CRYPTO_PRNG_SEED. Note that CRYPTO_PRNG_SEED should

be initialized since it is not initialized as the chip is powered up.

3. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
4. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then software can read the output random numbers from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7.

Programming steps to get the pseudo random number to key store are depicted below.

1. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
2. Write a random seed to CRYPTO_PRNG_SEED. Note that CRYPTO_PRNG_SEED should be initialized since it is not initialized as the chip is powered up.
3. Configure PRNG key control register CRYPTO_PRNG_KSCTL for key owner(CRYPTO_PRNG_KSCTL[26:24]), write key store destination(CRYPTO_PRNG_KSCTL[23:22]), write key destination(CRYPTO_PRNG_KSCTL[21]) and trusted key(CRYPTO_PRNG_KSCTL[16]) in key store.
4. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
5. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then software can read key NUMBER(CRYPTO_PRNG_KSSTS[4:0]).

Programming steps to get the random number with the true random number seed from TRNG are depicted below.

1. Select TRNG peripheral clock frequency in CLKPSC (TRNG_CTL[5:2]).
2. Enable ACT (TRNG_ACT[7]) bit.
3. Wait READY (TRNG_CTL[7]) bit to be set to 1.
4. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
5. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
6. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then software can read the output random numbers from CRYPTO_PRNG_KEY0 ~ CRYPTO_PRNG_KEY7.

Programming steps to get the random number with the true random number seed from TRNG to key store are depicted below.

1. Select TRNG peripheral clock frequency in CLKPSC (TRNG_CTL[5:2]).
2. Enable ACT (TRNG_ACT[7]) bit.
3. Wait READY (TRNG_CTL[7]) bit to be set to 1.
4. Check the BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0.
5. Configure PRNG key control register CRYPTO_PRNG_KSCTL for key owner(CRYPTO_PRNG_KSCTL[26:24]), write key store destination(CRYPTO_PRNG_KSCTL[23:22]), write key

destination(CRYPTO_PRNG_KSCTL[21]) and trusted key(CRYPTO_PRNG_KSCTL[16]) in key store.

6. Configure PRNG control register CRYPTO_PRNG_CTL for seed source (CRYPTO_PRNG_CTL[16]), key size(CRYPTO_PRNG_CTL[5:2]), seed reload(CRYPTO_PRNG_CTL[1]), and PRNG start(CRYPTO_PRNG_CTL[0]).
7. Software checks BUSY(CRYPTO_PRNG_CTL[8]) until it comes to 0, or waits for the PRNGIF(CRYPTO_INTSTS[16]) (must enable PRNGIEN(CRYPTO_INTEN[16])). Then software can read key NUMBER(CRYPTO_PRNG_KSSTS[4:0]).

Some Notices of PRNG:

1. If users want to generate key for key store, must to execute key store initialization (please refer to the Key Store (KS) section).
2.

6.41.5.3 AES (Advanced Encryption Standard)

The AES accelerator supports AES algorithm from U.S NIST. This accelerator provides 8 block cipher modes includes ECB, CBC, CFB, OFB, CTR, CBC-CS1, CBC-CS2 and CBC-CS3 mode. The detail of above block cipher modes are described in the following:

Electronic Codebook Mode

The Electronic Codebook (ECB) mode is a confidentiality mode that features the assignment of a fixed ciphertext block to each plaintext block, for a given key. It's analogous to the assignment of code words in a codebook.

In ECB encryption, each block of the plaintext is applied to the forward cipher function $CIPH_k$ directly and independently. The resulting sequence of output blocks is the ciphertext. In ECB decryption, each block of the ciphertext is applied to the inverse cipher function $CIPH^{-1}_k$ directly and independently. The resulting sequence of output blocks is the plaintext.

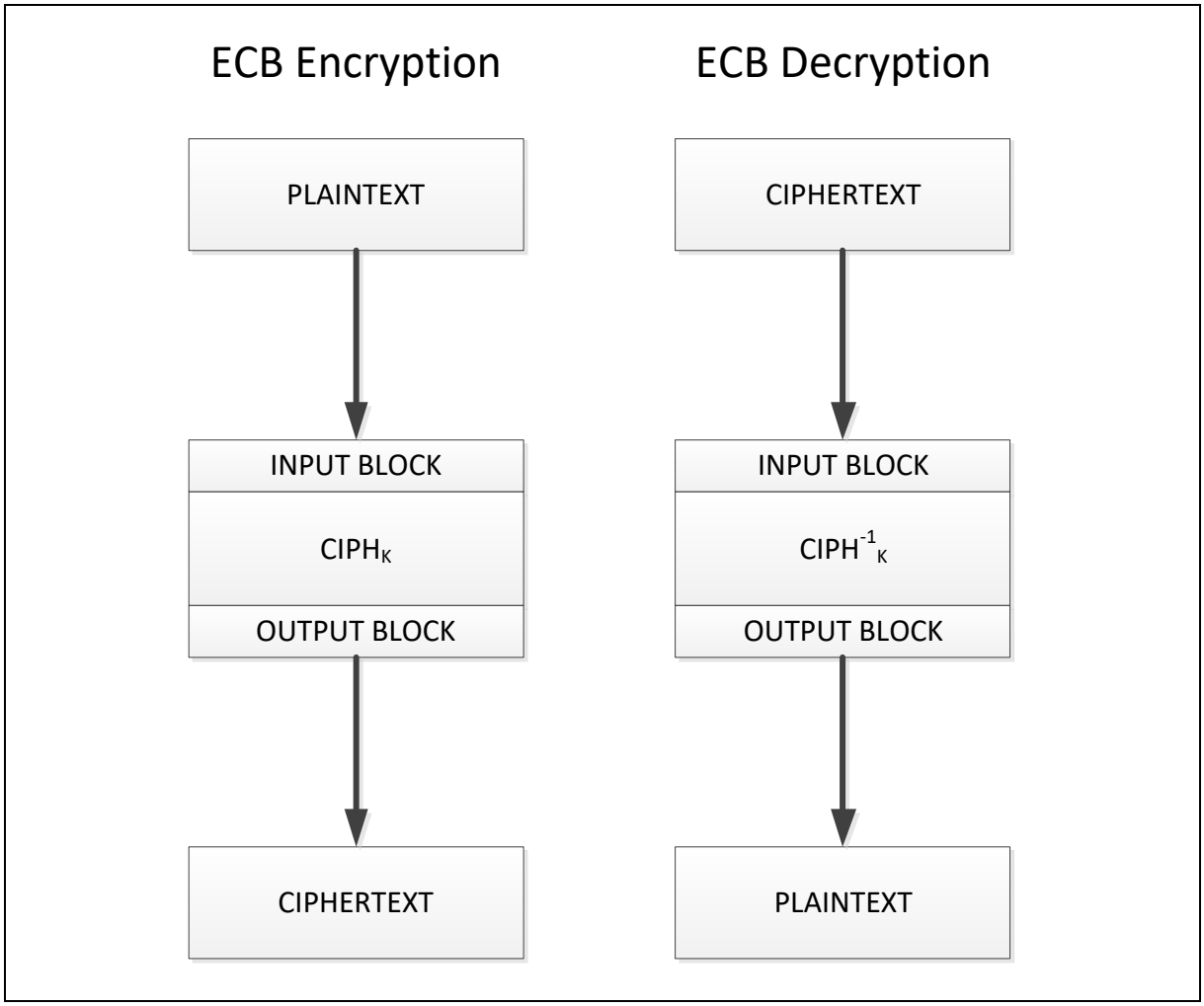


Figure 6.41-3 Electronic Codebook Mode

In ECB mode, any given plaintext block always gets encrypted to the same ciphertext block under a given key. If this property is undesirable in a particular application, the ECB mode should not be used.

Cipher Block Chaining Mode

The Cipher Block Chaining (CBC) mode is a confidentiality mode whose encryption process features the combining chaining of the plaintext blocks with the previous ciphertext blocks. The CBC mode requires an initialization vector (IV) to combine with the first plaintext block. The IV does not need to be secret, but it must be unpredictable.

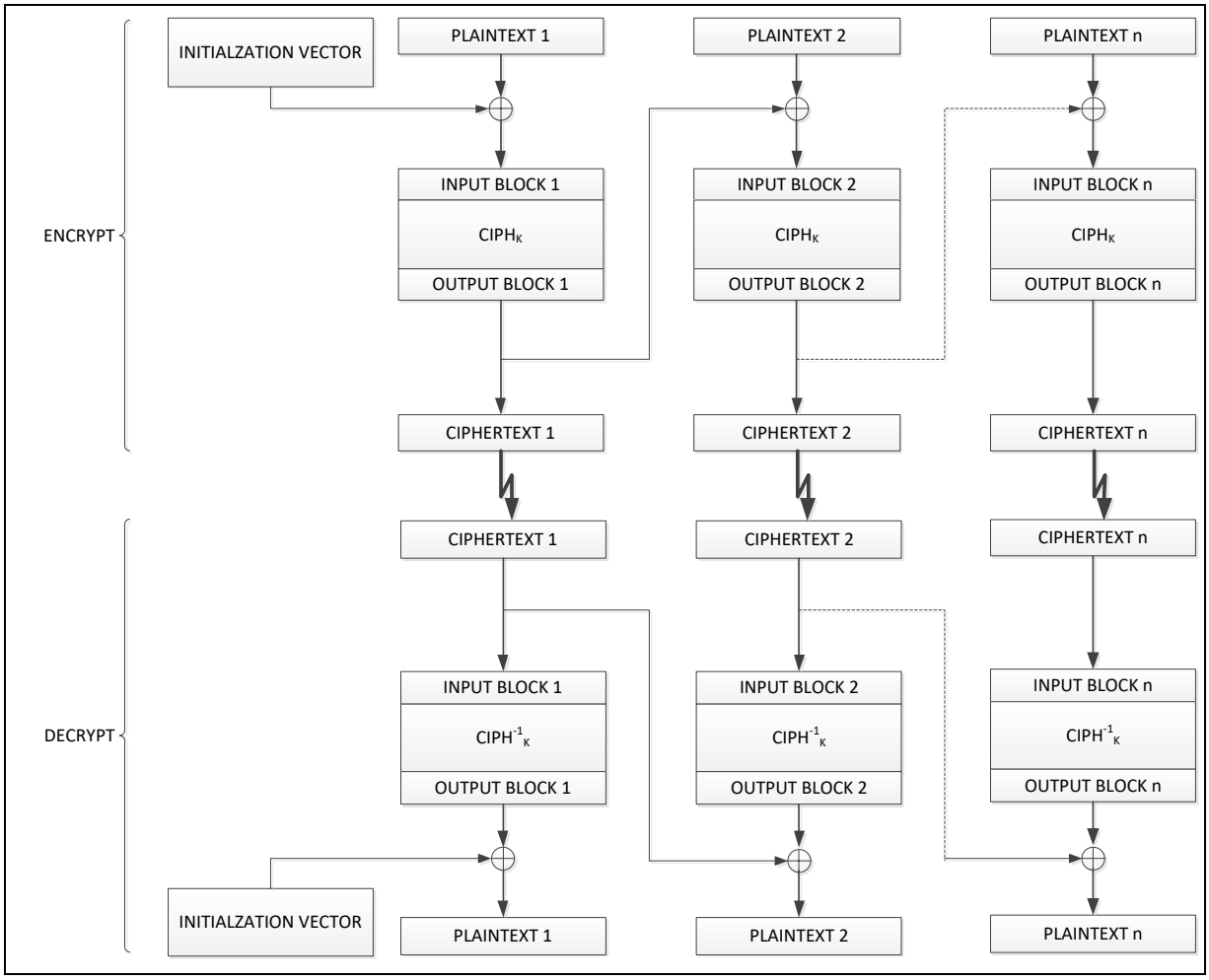


Figure 6.41-4 Cipher Block Chaining Mode

Cipher Feedback Mode (CFB)

The Cipher Feedback (CFB) mode is a confidentiality mode that features the feedback of successive ciphertext segments into the input blocks of the forward cipher to generate output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The CFB mode requires an IV as the initial input block. The IV need not be secret, but it must be unpredictable. The AES only supports 128-bit segment length CFB mode.

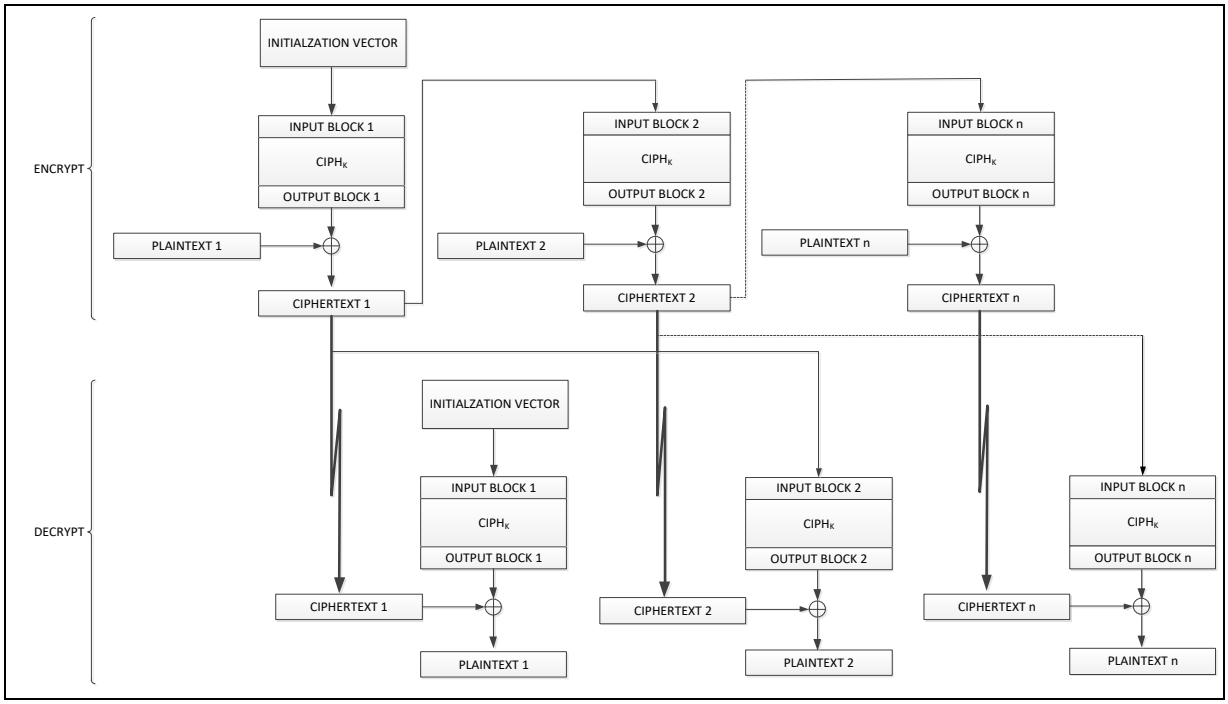


Figure 6.41-5 Cipher Feedback Mode

Output Feedback Mode

The Output Feedback (OFB) mode is a confidentiality mode that features the iteration of the forward cipher on an IV to generate a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The OFB mode requires that the IV is a nonce, i.e., the IV must be unique for each execution of the mode under the given key.

The OFB mode requires a unique IV for every message that is ever encrypted under the given key. If, contrary to this requirement, the same IV is used for the encryption of more than one message, then the confidentiality of those messages may be compromised. Confidentiality may be similarly be compromised if any of the input blocks to the forward cipher function for the encryption of a message is designated as the IV for the encryption of another message under the given key.

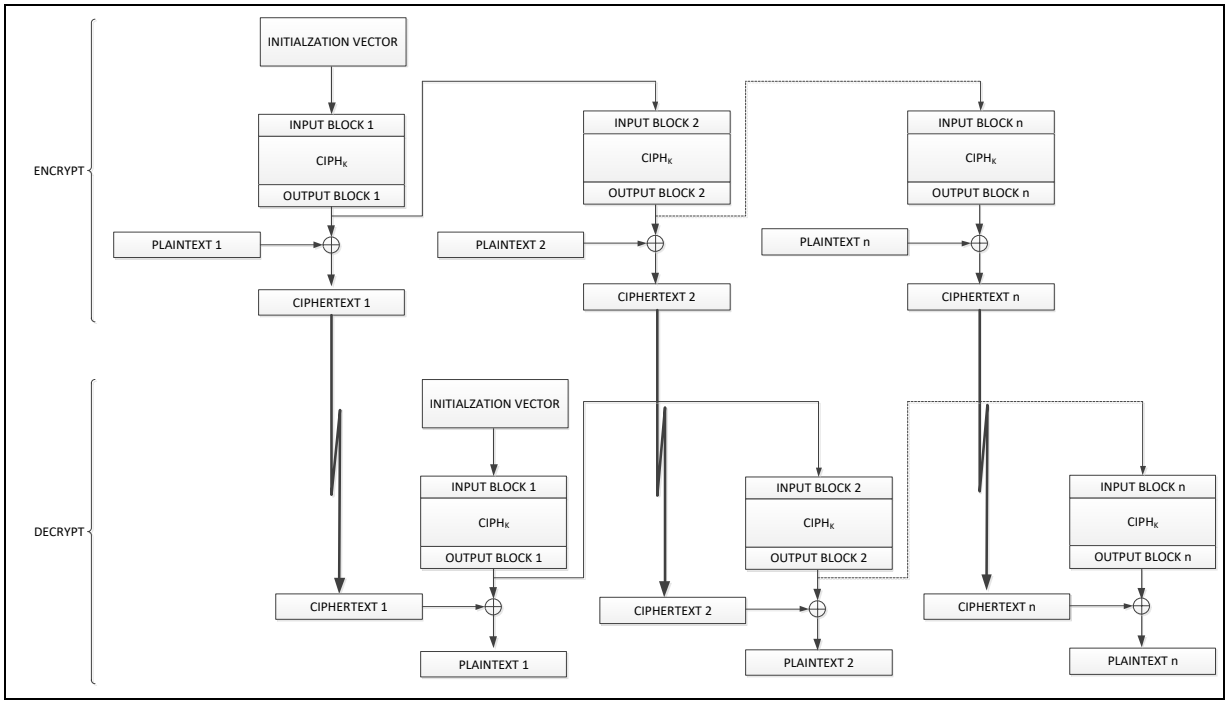


Figure 6.41-6 Output Feedback Mode

Counter Mode (CTR)

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. The sequence of counters must have the property that each block in the sequence is different from every other block. This condition is not restricted to a single message: across all of the messages that are encrypted under the given key, all of the counters must be distinct.

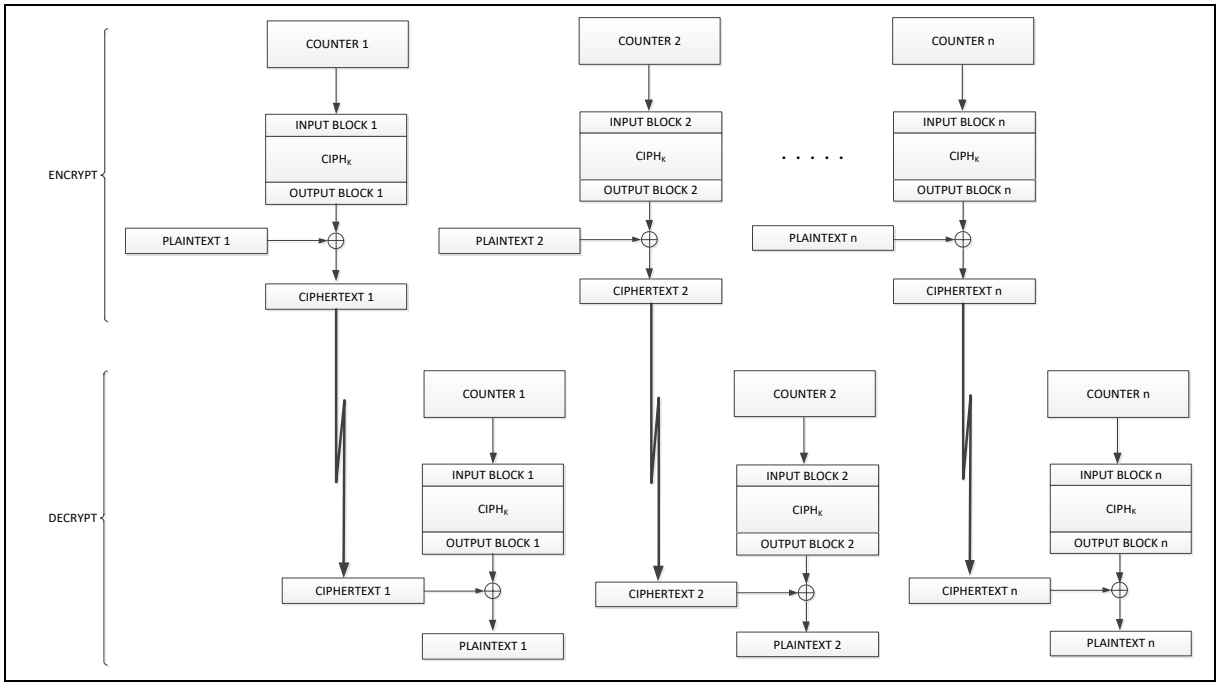


Figure 6.41-7 Counter Mode

CBC Ciphertext-Stealing 1 Mode (CBC-CS1)

Figure 6.41-8 illustrates the CBC-CS1-Encrypt algorithm for the case that P_n^* is a partial block. The cryptographic accelerator would append P_n^* with '0' to form a complete block P_n.

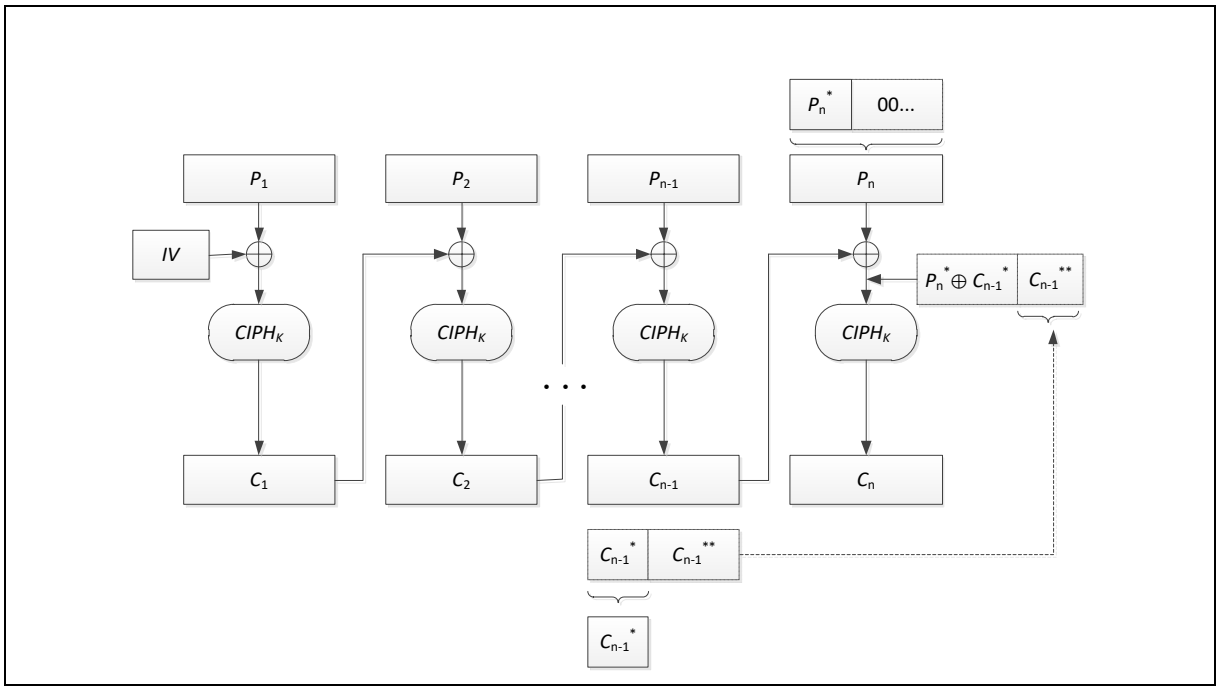


Figure 6.41-8 CBC-CS1 Encryption

Figure 6.41-9 illustrates the CBC-CS1-Decrypt algorithm for the case that C_{n-1}^* is a partial block.

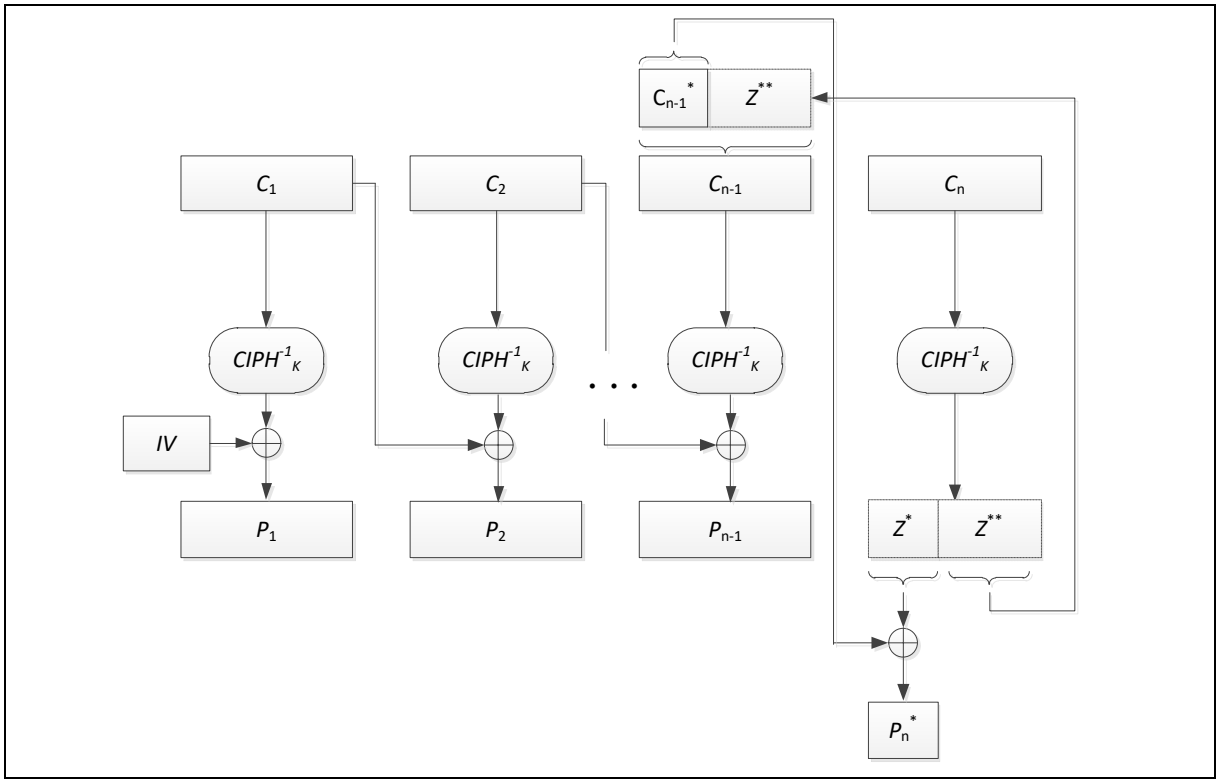


Figure 6.41-9 CBC-CS1 Decryption

CBC Ciphertext-Stealing 2 Mode (CBC-CS2)

When P_n^* is a partial block, then CBC-CS2-Encrypt and CBC-CS1-Encrypt differ only in the ordering of C_{n-1}^* and C_n.

CBC Ciphertext-Stealing 3 Mode (CBC-CS3)

C_{n-1}^* and C_n are unconditionally swapped, i.e., even when C_{n-1}^* is a complete block; therefore, CBC-CS3 is not strictly an extension of CBC mode. In the other case, i.e., when C_{n-1}^* is a nonempty partial block, CBC-CS3-Encrypt is equivalent to CBC-CS2-Encrypt.

Refer to the following programming steps for how to program the AES related registers.

AES DMA Mode Programming Flow

1. Write 1 to AESIEN (CRYPTO_INTEN[0]) to enable AES interrupt if needed.
2. Program AES key information to register CRYPTO_AES_KSCTL when key is from the key store.
3. Program AES key to registers CRYPTO_AES_KEY0 ~ CRYPTO_AES_KEY7 when AES key is not from the key store.
4. Program initial vectors to registers CRYPTO_AES_IV0 ~ CRYPTO_AES_IV3.
5. Program DMA source address to register CRYPTO_AES_SADDR.
6. Program DMA destination address to register CRYPTO_AES_DADDR.
7. Program DMA byte count to register CRYPTO_AES_CNT.
8. Program DMA feedback address to register CRYPTO_AES_FBADDR (18 words memory space must be reserved for feedback information) if DMA automatically feedback function is needed.
9. Program AES control register CRYPTO_AES_CTL for FBOUT and FBIN to trigger AES DMA

engine to write and read feedback information according to CRYPTO_AES_FBADDR if needed.

10. Configure AES control register CRYPTO_AES_CTL for key protection(KEYPRT), DMA input/output swap(INSWAP/OUTSWAP), encryption/decryption(ENCRYPTO), operational mode(OPMODE), DMA mode enable (DMAEN), DMA cascade control(DMACSCAD) and key size(KEYSZ). Write 1 to DMALAST (CRYPTO_AES_CTL[5]) if enabled DMACSCAD and current operation is last cascade operation.
11. Write input data to DMA source address with selected DMA byte count.
12. Write 1 to START(CRYPTO_AES_CTL[0]) to start AES encryption/decryption.
13. Wait for the AES interrupt flag AESIF (CRYPTO_INTSTS[0]) to be set or AES busy flag (CRYPTO_AES_STS[0]) to be set to zero.
14. Read output data from DMA destination address with selected DMA byte count.
15. Repeat step 5 to step 11 until all data processed if enabled DMACSCAD (CRYPTO_AES_CTL[6]).

AES Non-DMA Mode Programming Flow

1. Write 1 to AESIEN (CRYPTO_INTEN[0]) to enable AES interrupt if needed.
2. Program AES key information to register CRYPTO_AES_KSCTL when key is from the key store.
3. Program AES key to register CRYPTO_AES_KEY0 ~ CRYPTO_AES_KEY7 when AES key is not from the key store.
4. Program initial vectors to register CRYPTO_AES_IV0 ~ CRYPTO_AES_IV3.
5. Configure AES control register CRYPTO_AES_CTL for key protection (KEYPRT), input/output swap (INSWAP/OUTSWAP), encryption/decryption (ENCRYPTO), operational mode (OPMODE), DMA cascade control and key size(KEYSZ).
6. Write 1 to START(CRYPTO_AES_CTL[0]) to start AES encryption/decryption.
7. Polling INBUFFULL(CRYPTO_AES_STS[9]) and OUTBUFEMPTY(CRYPTO_AES_STS[16]). If INBUFFULL(CRYPTO_AES_STS[9]) is 0, write 32 bits input data to CRYPTO_AES_DATIN. If OUTBUFEMPTY(CRYPTO_AES_STS[16]) is 0, read 32 bits data from CRYPTO_AES_DATOUT.
8. Repeat step 7 until 128 bits data (16 bytes) are written to and read from AES engine.
9. Write 1 to DMALAST(CRYPTO_AES_CTL[5]) if current operation is last operation
10. Write data byte count of last operation to register CRYPTO_AES_CNT if current operation is last operation.
11. Repeat Steps 6 to 10 until all data processed.

6.41.6 Register Map

R: read only, **W:** write only, **R/W:** both read and write

Register	Offset	R/W	Description	Reset Value
CRYPTO Base Address: **CRYPTO_BA = 0x4003_2000**				
CRYPTO_INTEN	CRYPTO_BA+0x000	R/W	Crypto Interrupt Enable Control Register	0x0000_0000
CRYPTO_INTSTS	CRYPTO_BA+0x004	R/W	Crypto Interrupt Flag	0x0000_0000
CRYPTO_PRNG_CTL	CRYPTO_BA+0x008	R/W	PRNG Control Register	0x0000_0000
CRYPTO_PRNG_SEED	CRYPTO_BA+0x00C	W	Seed for PRNG	Undefined
CRYPTO_PRNG_KEY0	CRYPTO_BA+0x010	R	PRNG Generated Key0	Undefined
CRYPTO_PRNG_KEY1	CRYPTO_BA+0x014	R	PRNG Generated Key1	Undefined
CRYPTO_PRNG_KEY2	CRYPTO_BA+0x018	R	PRNG Generated Key2	Undefined
CRYPTO_PRNG_KEY3	CRYPTO_BA+0x01C	R	PRNG Generated Key3	Undefined
CRYPTO_PRNG_KEY4	CRYPTO_BA+0x020	R	PRNG Generated Key4	Undefined
CRYPTO_PRNG_KEY5	CRYPTO_BA+0x024	R	PRNG Generated Key5	Undefined
CRYPTO_PRNG_KEY6	CRYPTO_BA+0x028	R	PRNG Generated Key6	Undefined
CRYPTO_PRNG_KEY7	CRYPTO_BA+0x02C	R	PRNG Generated Key7	Undefined
CRYPTO_PRNG_STS	CRYPTO_BA+0x030	R	PRNG Status Register	0x0000_0000
CRYPTO_AES_FDBCK0	CRYPTO_BA+0x050	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK1	CRYPTO_BA+0x054	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK2	CRYPTO_BA+0x058	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK3	CRYPTO_BA+0x05C	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FBADDR	CRYPTO_BA+0x0A0	R/W	AES DMA Feedback Address Register	0x0000_0000
CRYPTO_AES_CTL	CRYPTO_BA+0x100	R/W	AES Control Register	0x0000_0000
CRYPTO_AES_STS	CRYPTO_BA+0x104	R	AES Engine Flag	0x0001_0100
CRYPTO_AES_DATIN	CRYPTO_BA+0x108	R/W	AES Engine Data Input Port Register	0x0000_0000
CRYPTO_AES_DATOUT	CRYPTO_BA+0x10C	R	AES Engine Data Output Port Register	0x0000_0000
CRYPTO_AES_KEY0	CRYPTO_BA+0x110	R/W	AES Key Word 0 Register	0x0000_0000
CRYPTO_AES_KEY1	CRYPTO_BA+0x114	R/W	AES Key Word 1 Register	0x0000_0000
CRYPTO_AES_KEY2	CRYPTO_BA+0x118	R/W	AES Key Word 2 Register	0x0000_0000
CRYPTO_AES_KEY3	CRYPTO_BA+0x11C	R/W	AES Key Word 3 Register	0x0000_0000
CRYPTO_AES_KEY4	CRYPTO_BA+0x120	R/W	AES Key Word 4 Register	0x0000_0000
CRYPTO_AES_KEY5	CRYPTO_BA+0x124	R/W	AES Key Word 5 Register	0x0000_0000
CRYPTO_AES_KEY6	CRYPTO_BA+0x128	R/W	AES Key Word 6 Register	0x0000_0000

CRYPTO_AES_KEY7	CRYPTO_BA+0x12C	R/W	AES Key Word 7 Register	0x0000_0000
CRYPTO_AES_IV0	CRYPTO_BA+0x130	R/W	AES Initial Vector Word 0 Register	0x0000_0000
CRYPTO_AES_IV1	CRYPTO_BA+0x134	R/W	AES Initial Vector Word 1 Register	0x0000_0000
CRYPTO_AES_IV2	CRYPTO_BA+0x138	R/W	AES Initial Vector Word 2 Register	0x0000_0000
CRYPTO_AES_IV3	CRYPTO_BA+0x13C	R/W	AES Initial Vector Word 3 Register	0x0000_0000
CRYPTO_AES_SADDR	CRYPTO_BA+0x140	R/W	AES DMA Source Address Register	0x0000_0000
CRYPTO_AES_DADDR	CRYPTO_BA+0x144	R/W	AES DMA Destination Address Register	0x0000_0000
CRYPTO_AES_CNT	CRYPTO_BA+0x148	R/W	AES Byte Count Register	0x0000_0000
CRYPTO_PRNG_KSCTL	CRYPTO_BA+0xF00	W	PRNG Key Control Register	0x0000_0000
CRYPTO_PRNG_KSSTS	CRYPTO_BA+0xF04	R	PRNG Key Status Register	0x0000_0000
CRYPTO_AES_KSCTL	CRYPTO_BA+0xF10	W	AES Key Control Register	0x0000_0000

6.41.7 Register Description

6.41.7.1 Crypto Register

CRYPTO Interrupt Enable Control Register (CRYPTO_INTEN)

Register	Offset	R/W	Description	Reset Value
CRYPTO_INTEN	CRYPTO_BA+0x000	R/W	Crypto Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						PRNGEIEN	PRNGIEN
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						AESEIEN	AESIEN

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**PRNGEIEN**	**PRNG Error Flag Enable Bit** 0 = PRNG error interrupt flag Disabled. 1 = PRNG error interrupt flag Enabled.
[16]	**PRNGIEN**	**PRNG Interrupt Enable Bit** 0 = PRNG interrupt Disabled. 1 = PRNG interrupt Enabled.
[15:2]	**Reserved**	Reserved.
[1]	**AESEIEN**	**AES Error Flag Enable Bit** 0 = AES error interrupt flag Disabled. 1 = AES error interrupt flag Enabled.
[0]	**AESIEN**	**AES Interrupt Enable Bit** 0 = AES interrupt Disabled. 1 = AES interrupt Enabled. **Note:** In DMA mode, an interrupt will be triggered when an amount of data set in AES_DMA_CNT is fed into the AES engine. In Non-DMA mode, an interrupt will be triggered when the AES engine finishes the operation.

CRYPTO Interrupt Flag Register (CRYPTO_INTSTS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_INTSTS	CRYPTO_BA+0x004	R/W	Crypto Interrupt Flag	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						PRNGEIF	PRNGIF
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						AESEIF	AESIF

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**PRNGEIF**	**PRNG Error Flag** 0 = No PRNG error. 1 = PRNG key generation error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[16]	**PRNGIF**	**PRNG Finish Interrupt Flag** 0 = No PRNG interrupt. 1 = PRNG key generation done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[15:2]	**Reserved**	Reserved.
[1]	**AESEIF**	**AES Error Flag** 0 = No AES error. 1 = AES encryption/decryption error interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.
[0]	**AESIF**	**AES Finish Interrupt Flag** 0 = No AES interrupt. 1 = AES encryption/decryption done interrupt. **Note:** This bit is cleared by writing 1, and it has no effect by writing 0.

6.41.7.2 PRNG Register

PRNG Control Register (CRYPTO_PRNG_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_CTL	CRYPTO_BA+0x008	R/W	PRNG Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							SEEDSRC
15	14	13	12	11	10	9	8
Reserved							BUSY
7	6	5	4	3	2	1	0
Reserved		KEYSZ				SEEDRLD	START

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**SEEDSRC**	**Seed Source** 0 = Seed is from TRNG. 1 = Seed is from PRNG seed register. **Note:** When SEEDRLD is set to 0, this bit (SEEDSRC) is meaningless.
[15:9]	**Reserved**	Reserved.
[8]	**BUSY**	**PRNG Busy (Read Only)** 0 = PRNG engine is idle. 1 = PRNG engine is generating CRYPTO_PRNG_KEYx.
[7:6]	**Reserved**	Reserved.
[5:2]	**KEYSZ**	**PRNG Generate Key Size** 0000 = 128 bits. 0010 = 192 bits. 0110 = 256 bits. Others = Reserved.
[1]	**SEEDRLD**	**Reload New Seed for PRNG Engine** 0 = Generating key based on the current seed. 1 = Reload new seed.
[0]	**START**	**Start PRNG Engine** 0 = Stop PRNG engine. 1 = Generate new key and store the new key to register CRYPTO_PRNG_KEYx, which will be cleared when the new key is generated.

PRNG Seed Register (CRYPTO_PRNG_SEED)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_SEED	CRYPTO_BA+0x00C	W	Seed for PRNG	Undefined

31	30	29	28	27	26	25	24
SEED							
23	22	21	20	19	18	17	16
SEED							
15	14	13	12	11	10	9	8
SEED							
7	6	5	4	3	2	1	0
SEED							

Bits	Description	
[31:0]	SEED	**Seed for PRNG (Write Only)** The bits store the seed for PRNG engine. **Note:** In TRNG+PRNG mode, the seed is from TRNG engine, and it will not be stored in this register.

PRNG Key x Register (CRYPTO_PRNG_KEYx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KEY0	CRYPTO_BA+0x010	R	PRNG Generated Key0	Undefined
CRYPTO_PRNG_KEY1	CRYPTO_BA+0x014	R	PRNG Generated Key1	Undefined
CRYPTO_PRNG_KEY2	CRYPTO_BA+0x018	R	PRNG Generated Key2	Undefined
CRYPTO_PRNG_KEY3	CRYPTO_BA+0x01C	R	PRNG Generated Key3	Undefined
CRYPTO_PRNG_KEY4	CRYPTO_BA+0x020	R	PRNG Generated Key4	Undefined
CRYPTO_PRNG_KEY5	CRYPTO_BA+0x024	R	PRNG Generated Key5	Undefined
CRYPTO_PRNG_KEY6	CRYPTO_BA+0x028	R	PRNG Generated Key6	Undefined
CRYPTO_PRNG_KEY7	CRYPTO_BA+0x02C	R	PRNG Generated Key7	Undefined

31	30	29	28	27	26	25	24
KEY							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
KEY							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
KEY							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
KEY							

Bits	Description	
[31:0]	**KEY**	**Store PRNG Generated Key (Read Only)** The bits store the key that is generated by PRNG.

PRNG Status Register (CRYPTO_PRNG_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_STS	CRYPTO_BA+0x030	R	PRNG Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						KSERR	Reserved
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							BUSY

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**KSERR**	**PRNG Access Key Store Error Flag** 0 = No error. 1 = Access key store failed.
[16:1]	**Reserved**	Reserved.
[0]	**BUSY**	**PRNG Busy Flag** 0 = PRNG engine is idle. 1 = PRNG engine is generating CRYPTO_PRNG_KEYx.

6.41.7.3 AES Register

AES Feedback x Register (CRYPTO_AES_FDBCKx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_FDBCK0	CRYPTO_BA+0x050	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK1	CRYPTO_BA+0x054	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK2	CRYPTO_BA+0x058	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000
CRYPTO_AES_FDBCK3	CRYPTO_BA+0x05C	R	AES Engine Output Feedback Data After Cryptographic Operation	0x0000_0000

31	30	29	28	27	26	25	24
FDBCK							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FDBCK							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FDBCK							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FDBCK							

Bits	Description	
[31:0]	**FDBCK**	**AES Feedback Information** The feedback value is 128 bits in size. The AES engine uses the data from CRYPTO_AES_FDBCKx as the data inputted to CRYPTO_AES_IVx for the next block in DMA cascade mode. The AES engine outputs feedback information for IV in the next block's operation. Software can use this feedback information to implement more than four DMA channels. Software can store that feedback value temporarily. After switching back, fill the stored feedback value to CRYPTO_AES_IVx in the same channel operation, and then continue the operation with the original setting.

AES DMA Feedback Address Register (CRYPTO_AES_FBADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_FBADDR	CRYPTO_BA+0x0A0	R/W	AES DMA Feedback Address Register	0x0000_0000

31	30	29	28	27	26	25	24
FBADDR							
23	22	21	20	19	18	17	16
FBADDR							
15	14	13	12	11	10	9	8
FBADDR							
7	6	5	4	3	2	1	0
FBADDR							

Bits	Description	
[31:0]	**FBADDR**	**AES DMA Feedback Address** In DMA cascade mode, software can update DMA feedback address register for automatically reading and writing feedback values via DMA. The FBADDR keeps the feedback address of the feedback data for the next cascade operation. Based on the feedback address, the AES accelerator can read the feedback data of the last cascade operation from SRAM memory space and write the feedback data of the current cascade operation to SRAM memory space. The start of feedback address should be located at word boundary. In other words, bit 1 and 0 of FBADDR are ignored. FBADDR can be read and written. In DMA mode, software can update the next CRYPTO_AES_FBADDR before triggering START.

AES Control Register (CRYPTO_AES_CTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_CTL	CRYPTO_BA+0x100	R/W	AES Control Register	0x0000_0000

31	30	29	28	27	26	25	24
KEYPRT	KEYUNPRT					KINSWAP	KOUTSWAP
23	22	21	20	19	18	17	16
INSWAP	OUTSWAP	FBOUT	FBIN	Reserved			ENCRYPTO
15	14	13	12	11	10	9	8
OPMODE							
7	6	5	4	3	2	1	0
DMAEN	DMACSCAD	DMALAST	Reserved	KEYSZ		STOP	START

Bits	Description	
[31]	KEYPRT	**Protect Key** Read as a flag to reflect KEYPRT. 0 = No effect. 1 = Protect the content of the AES key from reading. The return value for reading CRYPTO_AES_KEYx is not the content of the registers CRYPTO_AES_KEYx. Once it is set, it can be cleared by asserting KEYUNPRT. The key content would be cleared as well.
[30:26]	KEYUNPRT	**Unprotect Key** Writing 0 to CRYPTO_AES_CTL[31] and “10110” to CRYPTO_AES_CTL[30:26] is to unprotect the AES key. The KEYUNPRT can be read and written. When it is written as the AES engine is operating, BUSY flag is 1, there would be no effect on KEYUNPRT.
[25]	KINSWAP	**AES Engine Input Key and Initial Vector Swap** 0 = Keep the original order. 1 = The order that CPU feeds key and initial vector to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[24]	KOUTSWAP	**AES Engine Output Key, Initial Vector and Feedback Swap** 0 = Keep the original order. 1 = The order that CPU reads key, initial vector and feedback from the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[23]	INSWAP	**AES Engine Input Data Swap** 0 = Keep the original order. 1 = The order that CPU feeds data to the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.
[22]	OUTSWAP	**AES Engine Output Data Swap** 0 = Keep the original order. 1 = The order that CPU reads data from the accelerator will be changed from {byte3, byte2, byte1, byte0} to {byte0, byte1, byte2, byte3}.

[21]	**FBOUT**	**Feedback Output From AES Via DMA Automatically** 0 = DMA automatic feedback output function Disabled. 1 = DMA automatic feedback output function Enabled when DMAEN = 1.
[20]	**FBIN**	**Feedback Input to AES Via DMA Automatically** 0 = DMA automatic feedback input function Disabled. 1 = DMA automatic feedback input function Enabled when DMAEN = 1.
[19:17]	**Reserved**	Reserved.
[16]	**ENCRYPTO**	**AES Encryption/Decryption** 0 = AES engine executes decryption operation. 1 = AES engine executes encryption operation.
[15:8]	**OPMODE**	**AES Engine Operation Modes** 0x00 = ECB (Electronic Codebook Mode). 0x01 = CBC (Cipher Block Chaining Mode). 0x02 = CFB (Cipher Feedback Mode). 0x03 = OFB (Output Feedback Mode). 0x04 = CTR (Counter Mode). 0x10 = CBC-CS1 (CBC Ciphertext-Stealing 1 Mode). 0x11 = CBC-CS2 (CBC Ciphertext-Stealing 2 Mode). 0x12 = CBC-CS3 (CBC Ciphertext-Stealing 3 Mode).
[7]	**DMAEN**	**AES Engine DMA Enable Bit** 0 = AES DMA engine Disabled. The AES engine operates in Non-DMA mode. The data need to be written in CRYPTO_AES_DATIN. 1 = AES_DMA engine Enabled. The AES engine operates in DMA mode, and data movement from/to the engine is done by DMA logic.
[6]	**DMACSCAD**	**AES Engine DMA with Cascade Mode** 0 = DMA cascade function Disabled. 1 = In DMA cascade mode, software can update DMA source address register, destination address register, and byte count register during a cascade operation, without finishing the accelerator operation. **Note:** The last two blocks of AES-CBC-CS1/2/3 must be in the last cascade operation.
[5]	**DMALAST**	**AES Last Block** In DMA mode, this bit must be set as beginning the last DMA cascade round. In Non-DMA mode, this bit must be set when feeding in the last block of data in ECB, CBC, CTR, OFB, and CFB mode, and feeding in the (last-1) block of data at CBC-CS1, CBC-CS2, and CBC-CS3 mode. This bit is always 0 when it is read back, and must be written again once START is triggered.
[4]	**Reserved**	Reserved.
[3:2]	**KEYSZ**	**AES Key Size** This bit defines three different key size for AES operation. 2'b00 = 128 bits key. 2'b01 = 192 bits key. 2'b10 = 256 bits key. 2'b11 = Reserved. If the AES accelerator is operating and the corresponding flag BUSY is 1, updating this register has no effect.

[1]	**STOP**	**AES Engine Stop** 0 = No effect. 1 = Stop AES engine. **Note:** This bit is always 0 when it is read back.
[0]	**START**	**AES Engine Start** 0 = No effect. 1 = Start AES engine. BUSY flag will be set. **Note:** This bit is always 0 when it is read back.

AES Status Flag Register (CRYPTO_AES_STS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_STS	CRYPTO_BA+0x104	R	AES Engine Flag	0x0001_0100

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		KSERR	BUSERR	Reserved	OUTBUFERR	OUTBUFFULL	OUTBUFEMPTY
15	14	13	12	11	10	9	8
Reserved			CNTERR	Reserved	INBUFERR	INBUFFULL	INBUFEMPTY
7	6	5	4	3	2	1	0
Reserved							BUSY

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**KSERR**	**AES Engine Access Key Store Error Flag** 0 = No error. 1 = Key store access error will stop AES engine.
[20]	**BUSERR**	**AES DMA Access Bus Error Flag** 0 = No error. 1 = Bus error will stop DMA operation and AES engine.
[19]	**Reserved**	Reserved.
[18]	**OUTBUFERR**	**AES Out Buffer Error Flag** 0 = No error. 1 = Error happened during getting the result from AES engine.
[17]	**OUTBUFFULL**	**AES Out Buffer Full Flag** 0 = AES output buffer is not full. 1 = AES output buffer is full, and software needs to get data from CRYPTO_AES_DATOUT. Otherwise, the AES engine will be pending since the output buffer is full.
[16]	**OUTBUFEMPTY**	**AES Out Buffer Empty** 0 = AES output buffer is not empty. There are some valid data kept in output buffer. 1 = AES output buffer is empty. Software cannot get data from CRYPTO_AES_DATOUT. Otherwise, the flag OUTBUFERR will be set to 1 since the output buffer is empty.
[15:13]	**Reserved**	Reserved.
[12]	**CNTERR**	**CRYPTO_AES_CNT Setting Error** 0 = No error in CRYPTO_AES_CNT setting. 1 = CRYPTO_AES_CNT is 0 if DMAEN (CRYPTO_AES_CTL[7]) is enabled.
[11]	**Reserved**	Reserved.

[10]	**INBUFERR**	**AES Input Buffer Error Flag** 0 = No error. 1 = Error happened during feeding data to the AES engine.
[9]	**INBUFFULL**	**AES Input Buffer Full Flag** 0 = AES input buffer is not full. Software can feed the data into the AES engine. 1 = AES input buffer is full. Software cannot feed data to the AES engine. Otherwise, the flag INBUFERR will be set to 1.
[8]	**INBUFEMPTY**	**AES Input Buffer Empty** 0 = There are some data in input buffer waiting for the AES engine to process. 1 = AES input buffer is empty. Software needs to feed data to the AES engine. Otherwise, the AES engine will be pending to wait for input data.
[7:1]	**Reserved**	Reserved.
[0]	**BUSY**	**AES Engine Busy** 0 = The AES engine is idle or finished. 1 = The AES engine is under processing.

AES Data Input Port Register (CRYPTO_AES_DATIN)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DATIN	CRYPTO_BA+0x108	R/W	AES Engine Data Input Port Register	0x0000_0000

31	30	29	28	27	26	25	24
DATIN							
23	22	21	20	19	18	17	16
DATIN							
15	14	13	12	11	10	9	8
DATIN							
7	6	5	4	3	2	1	0
DATIN							

Bits	Description	
[31:0]	DATIN	**AES Engine Input Port** CPU feeds data to AES engine through this port by checking CRYPTO_AES_STS. Feed data as INBUFFULL is 0.

AES Data Output Port Register (CRYPTO_AES_DATOUT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DATOUT	CRYPTO_BA+0x10C	R	AES Engine Data Output Port Register	0x0000_0000

31	30	29	28	27	26	25	24
DATOUT							
23	22	21	20	19	18	17	16
DATOUT							
15	14	13	12	11	10	9	8
DATOUT							
7	6	5	4	3	2	1	0
DATOUT							

Bits	Description	
[31:0]	DATOUT	**AES Engine Output Port** CPU gets results from the AES engine through this port by checking CRYPTO_AES_STS. Get data as OUTBUFEMPTY is 0.

AES Key Word x Register (CRYPTO_AES_KEYx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_KEY0	CRYPTO_BA+0x110	R/W	AES Key Word 0 Register	0x0000_0000
CRYPTO_AES_KEY1	CRYPTO_BA+0x114	R/W	AES Key Word 1 Register	0x0000_0000
CRYPTO_AES_KEY2	CRYPTO_BA+0x118	R/W	AES Key Word 2 Register	0x0000_0000
CRYPTO_AES_KEY3	CRYPTO_BA+0x11C	R/W	AES Key Word 3 Register	0x0000_0000
CRYPTO_AES_KEY4	CRYPTO_BA+0x120	R/W	AES Key Word 4 Register	0x0000_0000
CRYPTO_AES_KEY5	CRYPTO_BA+0x124	R/W	AES Key Word 5 Register	0x0000_0000
CRYPTO_AES_KEY6	CRYPTO_BA+0x128	R/W	AES Key Word 6 Register	0x0000_0000
CRYPTO_AES_KEY7	CRYPTO_BA+0x12C	R/W	AES Key Word 7 Register	0x0000_0000

31	30	29	28	27	26	25	24
KEY							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
KEY							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
KEY							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
KEY							

Bits	Description	
[31:0]	**KEY**	**CRYPTO_AES_KEYx** The KEY keeps the security key for AES operation. x = 0, 1..7. The security key for AES accelerator can be 128, 192, or 256 bits and four, six, or eight 32-bit registers are to store each security key. {CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 128-bit security key for AES operation. {CRYPTO_AES_KEY5, CRYPTO_AES_KEY4, CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 192-bit security key for AES operation. {CRYPTO_AES_KEY7, CRYPTO_AES_KEY6, CRYPTO_AES_KEY5, CRYPTO_AES_KEY4, CRYPTO_AES_KEY3, CRYPTO_AES_KEY2, CRYPTO_AES_KEY1, CRYPTO_AES_KEY0} stores the 256-bit security key for AES operation.

AES Initial Vector Word x Register (CRYPTO_AES_IVx)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_IV0	CRYPTO_BA+0x130	R/W	AES Initial Vector Word 0 Register	0x0000_0000
CRYPTO_AES_IV1	CRYPTO_BA+0x134	R/W	AES Initial Vector Word 1 Register	0x0000_0000
CRYPTO_AES_IV2	CRYPTO_BA+0x138	R/W	AES Initial Vector Word 2 Register	0x0000_0000
CRYPTO_AES_IV3	CRYPTO_BA+0x13C	R/W	AES Initial Vector Word 3 Register	0x0000_0000

31	30	29	28	27	26	25	24
IV							
23	22	21	20	19	18	17	16
IV							
15	14	13	12	11	10	9	8
IV							
7	6	5	4	3	2	1	0
IV							

Bits		Description
[31:0]	**IV**	**AES Initial Vectors** x = 0, 1..3. Four initial vectors (CRYPTO_AES_IV0, CRYPTO_AES_IV1, CRYPTO_AES_IV2, and CRYPTO_AES_IV3) are for AES operating in CBC, CFB, and OFB mode. Four registers (CRYPTO_AES_IV0, CRYPTO_AES_IV1, CRYPTO_AES_IV2, and CRYPTO_AES_IV3) act as Nonce counter when the AES engine is operating in CTR mode.

AES DMA Source Address Register (CRYPTO_AES_SADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_SADDR	CRYPTO_BA+0x140	R/W	AES DMA Source Address Register	0x0000_0000

31	30	29	28	27	26	25	24
SADDR							
23	22	21	20	19	18	17	16
SADDR							
15	14	13	12	11	10	9	8
SADDR							
7	6	5	4	3	2	1	0
SADDR							

Bits	Description	
[31:0]	**SADDR**	**AES DMA Source Address** The AES accelerator supports DMA function to transfer the plain text between SRAM memory space and embedded FIFO. The SADDR keeps the source address of the data buffer where the source text is stored. Based on the source address, the AES accelerator can read the plain text (encryption) / cipher text (decryption) from SRAM memory space and do AES operation. The start of source address should be located at word boundary. In other words, bit 1 and 0 of SADDR are ignored. SADDR can be read and written. Writing to SADDR while the AES accelerator is operating doesn't affect the current AES operation. But the value of SADDR will be updated later on. Consequently, software can prepare the DMA source address for the next AES operation. In DMA mode, software can update the next CRYPTO_AES_SADDR before triggering START. The value of CRYPTO_AES_SADDR and CRYPTO_AES_DADDR can be the same.

AES DMA Destination Address Register (CRYPTO_AES_DADDR)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_DADDR	CRYPTO_BA+0x144	R/W	AES DMA Destination Address Register	0x0000_0000

31	30	29	28	27	26	25	24
DADDR							
23	22	21	20	19	18	17	16
DADDR							
15	14	13	12	11	10	9	8
DADDR							
7	6	5	4	3	2	1	0
DADDR							

Bits	Description	
[31:0]	DADDR	**AES DMA Destination Address** The AES accelerator supports DMA function to transfer the cipher text between SRAM memory space and embedded FIFO. The DADDR keeps the destination address of the data buffer where the engine output's text will be stored. Based on the destination address, the AES accelerator can write the cipher text (encryption) / plain text (decryption) back to SRAM memory space after the AES operation is finished. The start of destination address should be located at word boundary. In other words, bit 1 and 0 of DADDR are ignored. DADDR can be read and written. Writing to DADDR while the AES accelerator is operating doesn't affect the current AES operation. But the value of DADDR will be updated later on. Consequently, software can prepare the destination address for the next AES operation. In DMA mode, software can update the next CRYPTO_AES_DADDR before triggering START. The value of CRYPTO_AES_SADDR and CRYPTO_AES_DADDR can be the same.

AES Byte Count Register (CRYPTO_AES_CNT)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_CNT	CRYPTO_BA+0x148	R/W	AES Byte Count Register	0x0000_0000

31	30	29	28	27	26	25	24
CNT							
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:0]	CNT	**AES Byte Count** The CRYPTO_AES_CNT keeps the byte count of source text that is for the AES engine operating in DMA mode. The CRYPTO_AES_CNT is 32-bit and the maximum of byte count is 4G bytes. CRYPTO_AES_CNT can be read and written. Writing to CRYPTO_AES_CNT while the AES accelerator is operating doesn't affect the current AES operation. But the value of CRYPTO_AES_CNT will be updated later on. Consequently, software can prepare the byte count of data for the next AES operation. According to CBC-CS1, CBC-CS2, and CBC-CS3 standard, the count of operation data must be more than 16 bytes. Operations that are equal or less than one block will output unexpected result. In Non-DMA ECB, CBC, CFB, OFB, and CTR mode, CRYPTO_AES_CNT must be set as byte count for the last block of data before feeding in the last block of data. In Non-DMA CBC-CS1, CBC-CS2, and CBC-CS3 mode, CRYPTO_AES_CNT must be set as byte count for the last two blocks of data before feeding in the last two blocks of data.

6.41.7.4 Key Control Register

PRNG Key Control Register (CRYPTO_PRNG_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KSCTL	CRYPTO_BA+0xF00	W	PRNG Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					OWNER		
23	22	21	20	19	18	17	16
WSDST		WDST	Reserved				TRUST
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			NUM				

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:24]	**OWNER**	**Write Key Owner Selection Bits** 000 = Only for AES use. 101 = Only for CPU engine use. Others = reserved.
[23:22]	**WSDST**	**Write Key Store Destination** 10 = Key is written to the OTP of key store. Others = reserved.
[21]	**WDST**	**Write Key Destination** 0 = Key is written to registers CRYPTO_PRNG_KEYx. 1 = Key is written to key store.
[20:17]	**Reserved**	Reserved.
[16]	**TRUST**	**Write Key Trust Selection Bit** 0 = Set written key as the non-secure key. 1 = Set written key as the secure key.
[15:5]	**Reserved**	Reserved.
[4:0]	**NUM**	**Write Key Number** The key number is sent to key store **Note:** Only for destination Is OTP of key store.

PRNG Key Status Register (CRYPTO_PRNG_KSSTS)

Register	Offset	R/W	Description	Reset Value
CRYPTO_PRNG_KSSTS	CRYPTO_BA+0xF04	R	PRNG Key Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			NUM				

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**NUM**	**Key Number** The key number is generated by key store

AES Key Control Register (CRYPTO_AES_KSCTL)

Register	Offset	R/W	Description	Reset Value
CRYPTO_AES_KSCTL	CRYPTO_BA+0xF10	W	AES Key Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
RSSRC		RSRC	NUM				

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**RSSRC**	**Read Key Store Source** 10 = Key is read from the OTP of key store. Others = reserved.
[5]	**RSRC**	**Read Key Source** 0 = Key is read from registers CRYPTO_AESx_KEYx. 1 = Key is read from key store.
[4:0]	**NUM**	**Read Key Number** The key number is sent to key store

6.42 Key Store (KS)

6.42.1 Overview

The Key Store (KS) is the key management device and has a 4 Kbytes SRAM, 4 Kbytes Flash and OTP for key storage. The Key Store is capable of providing a crypto engine to access or storing the key while encryption, decryption and generation. The Key Store supports revoke key operation if the key is unused. The Key Store is able to protect the key by data remanence prevention for SRAM.

6.42.2 Features

- Supports programming interface for key management
- Supports key size required for Cryptographic engine from 128-bit to 4096-bit
- Supports 32 SRAM keys by 4 Kbytes SRAM
- Supports 32 Flash keys by 4 Kbytes Flash
- Supports 8 OTP keys, maximum key size is 256-bit
- Supports crypto engine access or store key in key store directly
- Supports to store AES/HMAC/ECC/RSA/CPU key
- Supports to store RSA middle data for CRT and SCAP mode
- Supports ECDH operation with ECC and PRNG engine
- Supports revoke operation
- Supports erase key in SRAM/Flash and revoke key in OTP while tamper event detected
- Supports data remanence prevention at SRAM
- Supports auto verify function
- Supports lock function for OTP keys

6.42.3 Block Diagram

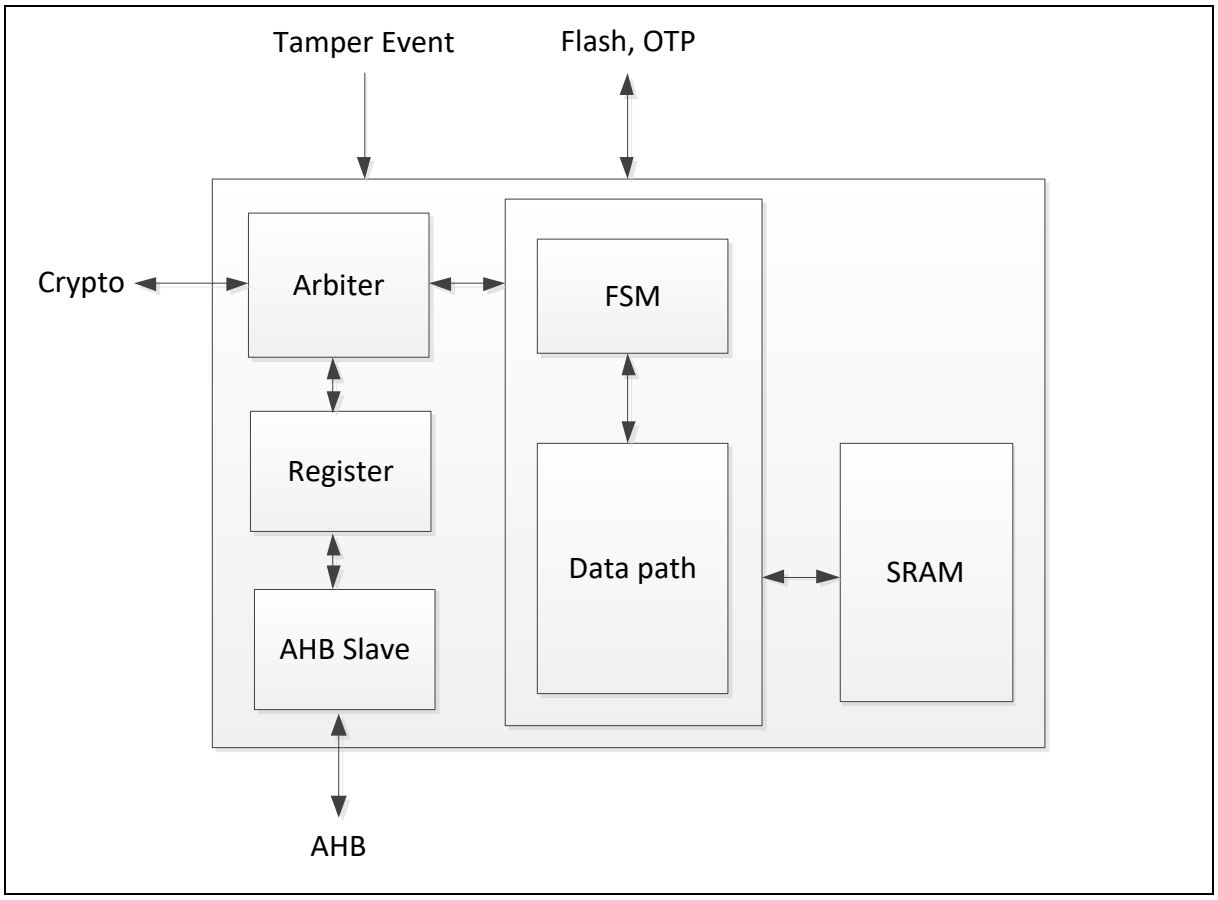


Figure 6.42-1 Key Store Block Diagram

6.42.4 Basic Configuration

- Clock Source Configuration
  - Enable KS peripheral clock in KSCKEN (CLK_AHBCLK0[13]).

6.42.5 Functional Description

6.42.5.1 Initialization

When chip is powered up, the Key Store has no information about key at Flash and OTP. The Key Store needs to read the metadata of keys from Flash and OTP, so Key Store should be initialized by setting INIT(KS_CTL[8]) and START(KS_CTL[0]). When the Key Store is initialized, INITDONE(KS_STS[7]) will be set. The Key Store will check INITDONE before executing the operation except the initialization. If INITDONE is 0, the current operation will fail and EIF(KS_STS[1]) will be set. User should clear EIF by writing it to 0 and re-initialize the Key Store.

Before initialization, KS_REMAIN and KS_OTPSTS register keeps the reset value. If initialization is finished, KS_REMAIN and KS_OTPSTS register will update current status.

The Key Store needs to be initialized when chip powered up, wakeup from SPD/DPD mode, and reset. If INITDONE is set, the Key Store does not need to do initialization again.

If KSPLOCK(CONFIG2[15:8]) in Flash Memory Controller is locked and mass erase occurs, the existing OTP keys will be revoked and SRAM keys and Flash keys will be erased after initialization. This process will only be done once. If the OTP key is created after that, this key can be normally used. If KSPLOCK is re-locked and mass erase occurs, this process will no longer be executed.

Initial flow:

1. Check BUSY(KS_STS[2]) is 0 and INITDONE(KS_STS[7]) is 0.
2. Set INIT(KS_CTL[8]) to 1 and START(KS_CTL[0]) to 1
3. Wait BUSY(KS_STS[2]) to 0
4. Check INITDONE(KS_STS[7]) is 1

6.42.5.2 Basic Operation

The Key Store has different placement mechanism at SRAM, Flash and OTP. The Key Store puts a key in SRAM and Flash by hardware placement, and puts the key in OTP by software placement. User does not need to set the key number, as long as user can read the key according key number returned by Key Store after key is placed in SRAM or Flash. If key is placed in OTP, user needs to give the key number before create key operation is executed.

There is a basic operation about create key, read key, erase key and revoke key in Key Store. User can select the location, fill in the metadata, select the operation and enable start bit, and then the Key Store will start operating. When the operation is finished, the Key Store will set the interrupt flag. If key creation is finished, the Key Store will return the key number for access this key in the future.

Each key has its metadata and metadata contains the information shown as follows.

1. Owner – OWNER(KS_METADATA[18:16])
   - Key can be set to which owner can used. Only owner can access key.
2. Size – SIZE(KS_METADATA[12:8])
   - Set key size.
3. Readable – READABLE(KS_METADATA[2])
   - Key can be read by CPU or not.
   - If OWNER is set to AES and READABLE is set to 1, CPU can read out key and AES can access.
   - If OWNER is set to CPU and READABLE is set to 0, this key is unused. CPU cannot read out and Crypto engine cannot access.
   - If OWNER is set to RSA middle data, CPU cannot read out no matter what READABLE is set to.
4. Booting State – BS(KS_METADATA[4])
   - Key can be used at booting state(BL1 state) or not. If BS is set, key can be used at booting state only.

Among the metadata mentioned above, only revoke attribute can be changed to 1 by revoke key operation and others metadata is not allowed to be changed.

User can create keys to Flash and OTP, and then Key Store programs the key and its metadata to Flash and OTP. It means the key storage at Flash is not totally 4 Kbytes space. When EIF(KS_STS[1]) is set, the EIF should be cleared by writing 1 before the next operation starts. If the create key operation is not finished at Flash, the next create key operation will write the wrong key.

If key is placed in SRAM, this key can be erased by erase one key operation and Key Store will release the storage space. If key is placed in OTP, this key can be erased by erase one key operation and Key Store will replace to spare space to add new key. Each key storage block has 3 spare space, so each OTP key has 3 times to erase.

Table 6.42-1 shows the detailed information for SRAM, Flash and OTP and Table 6.42-3 shows the information of each OTP key storage block.

	SRAM	Flash	OTP
Storage Size	4 Kbytes	4 Kbytes	1152 bytes
Store maximum key size	4096 bits	4096 bits	256 bits
Store maximum key count	32 keys	32 keys	8 keys
Store AES key	V	V	V
Store HMAC key	V	V	V
Store ECC key	V	V	V
Store RSA key	V	V	-
Store RSA middle data	V	-	-
Store CPU key	V	V	V
Store ECDH or ECDSA random key generated from PRNG	V	-	-
Read operation	V	V	V
Create operation	V	V	V
Erase one key operation	V	-	V (up to 3 times before locked)
Erase all key operation	V	V	-
Revoke key operation	V	V	V
Data Remanence	V	-	-
Lock key operation	-	-	V
Auto Verify	V	V	V

Table 6.42-1 Detailed Information for SRAM, Flash and OTP for M467 Series

	OTP
Storage Size	1152 bytes
Store maximum key size	256 bits
Store maximum key count	8 keys

Store AES key	V
Store HMAC key	-
Store ECC key	-
Store RSA key	-
Store RSA middle data	-
Store CPU key	V
Store ECDH or ECDSA random key generated from PRNG	-
Read operation	V
Create operation	V
Erase one key operation	V (up to 3 times before locked)
Erase all key operation	-
Revoke key operation	V
Data Remanence	-
Lock key operation	V
Auto Verify	V

Table 6.42-2 Detailed Information for SRAM, Flash and OTP for M463 Series

OTP Key Number	Key Naming	Key Storage Space	Erase Time
0	Secure boot ECC public key	256 bits	3
1	Secure boot ECC public key	256 bits	3
2	User defined key	256 bits	3
3	User defined key	256 bits	3
4	User defined key	256 bits	3
5	User defined key	256 bits	3
6	User defined key	256 bits	3
7	User defined key	256 bits	3

Table 6.42-3 Information of OTP Key Storage

The following shows read, create, erase and revoke operation.

Read Operation Programming Flow

Use read operation can read out which key is readable. If key is unreadable, EIF(KS_STS[1]) will be set to 1.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]), NUMBER(KS_METADATA[25:20])
3. Set OPMODE(KS_CTL[3:1]) to 000
4. Set START(KS_CTL[0]) to 1
5. Wait BUSY(KS_STS[2]) to 0.
6. Read key from KS_KEY0~KS_KEY7 registers if EIF(KS_STS[1]) is 0.
7. Repeat step 4 to step 6 and set CONT(KS_CTL[7]) to 1 when key size is larger than 256 bits

Create Operation Programming Flow at SRAM/Flash

Use create operation can create key to SRAM or Flash. After the create operation is finished, KS will return key number. If the number of create operation is greater than the number of needed, EIF(KS_STS[1]) will be set to 1.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Check FLASHFULL(KS_STS[4]) or SRAMFULL(KS_STS[3]) is 0.
3. Configure DST(KS_METADATA[31:30]), OWNER(KS_METADATA[18:16]), SIZE(KS_METADATA[12:8]),BS(KS_METADATA[4]), READABLE(KS_METADATA[2]) according key's metadata.
4. Set OPMODE(KS_CTL[3:1]) to 001
5. Write key to KS_KEY0~KEY7 registers
6. Set START(KS_CTL[0]) to 1
7. Wait BUSY(KS_STS[2]) to 0.
8. Check EIF(KS_STS[1]) is 0.
9. Repeat step 4 to step 7 and set CONT(KS_CTL[7]) to 1 when key size is larger than 256 bits
10. Read key number from NUMBER(KS_METADATA [25:20])

Note: Step 4 to Step 7 do not need to be repeated if key size is equal to or less than 256 bits, Step 4 to Step 7 need to be repeated once if the key size is larger than 256 bits and equal to or less than 512 bits, Step 4 to Step 7 need to be repeated twice if the key size is larger than 512 bits and equal to or less than 384 bits, and so on.

Create Operation Programming Flow at OTP

Use create operation can create key to OTP with specified key number. After create operation is finished, Ket Store will return key number according written the specified number. If the number of create operation is greater than the number of needed, EIF(KS_STS[1]) will be set to 1.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]), NUMBER(KS_METADATA[25:20]), OWNER(KS_METADATA[18:16]), SIZE(KS_METADATA[12:8]),BS(KS_METADATA[4]), READABLE(KS_METADATA[2]) according key's metadata.
3. Set OPMODE(KS_CTL[3:1]) to 001

4. Write key to KS_KEY0~KEY7 registers
5. Set START(KS_CTL[0]) to 1
6. Wait BUSY(KS_STS[2]) to 0.
7. Check EIF(KS_STS[1]) is 0.
8. Read key number from NUMBER(KS_METADATA [25:20]) and KS_OTPSTS register for double check.

Erase One Key Operation Programming Flow

Use erase one key operation can erase the specified key if key is unused. After the operation is finished, the space of erased key used will be released. This operation can only be used when key is stored in SRAM/OTP.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]), NUMBER(KS_METADATA[25:20])
3. Set OPMODE(KS_CTL[3:1]) to 010 and START(KS_CTL[0]) to 1
4. Wait BUSY(KS_STS[2]) to 0.
5. Check EIF(KS_STS[1]) is 0.

Erase All Keys Operation Programming Flow

Use erase all one key operation can erase whole SRAM or Flash of Key Store. After the operation is finished, the space of key used will be released. This operation can only be used to SRAM or Flash.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]).
3. Set OPMODE(KS_CTL[3:1]) to 011 and START(KS_CTL[0]) to 1
4. Wait BUSY(KS_STS[2]) to 0.
5. Check EIF(KS_STS[1]) is 0 and KS_REMAINING register to reset value

Revoke Key Operation Programming Flow

Use revoke key operation can revoke the specified key if key is unused. After the operation is finished, the key is unused but the space which key is used will not be released.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Configure DST(KS_METADATA[31:30]), NUMBER(KS_METADATA[25:20])
3. Set OPMODE(KS_CTL[3:1]) to 100 and START(KS_CTL[0]) to 1
4. Wait BUSY(KS_STS[2]) to 0.
5. Check EIF(KS_STS[1]) is 0.

Lock Key Operation Programming Flow

Use lock operation to lock key if key does not need to be updated. When the key is locked, this key only can be read out using read operation but cannot be erased or re-created using erase operation or create operation.

1. Check BUSY(KS_STS[2]) is 0 and EIF(KS_STS[1]) is 0.
2. Set DST(KS_METADATA[31:30]) to 10
3. Configure NUMBER(KS_METADATA[25:20])
4. Set OPMODE(KS_CTL[3:1]) to 111 and START(KS_CTL[0]) to 1

5. Wait BUSY(KS_STS[2]) to 0.
6. Check EIF(KS_STS[1]) is 0.

The byte counts used for key size in Table 6.42-4. KS_REMAIN register calculates the remaining space based on SIZE(KE_METADATA[12:8]) and OWNER(KS_METADATA[18:16]) for SRAM and Flash.

SIZE (KS_METADATA[12:8])	SRAM (DST = 0x0)	SRAM For RSA Blind Key (DST = 0 & OWNER = 0x2)	Flash (DST = 0x1)	Flash For RSA Blind Key (DST = 0x1 & OWNER = 0x2)
0x0 (128 bits)	16	16	20	20
0x1 (163 bits)	32	32	28	28
0x2 (192 bits)	32	32	28	28
0x3 (224 bits)	32	32	32	32
0x4 (233 bits)	32	32	36	36
0x5 (255 bits)	32	32	36	36
0x6 (256 bits)	32	32	36	36
0x7 (283 bits)	48	48	40	40
0x8 (384 bits)	48	48	52	52
0x9 (409 bits)	64	64	56	56
0xA (512 bits)	64	64	68	68
0xB (521 bits)	80	80	72	72
0xC (571 bits)	80	80	76	76
0x10 (1024 bits)	128	144	132	136
0x11 (1536 bits)	192	208	196	200
0x12 (2048 bits)	256	272	260	264
0x13 (3072 bits)	384	400	388	392
0x14 (4096 bits)	512	528	516	520

Table 6.42-4 Byte Counts for Key Size at SRAM and Flash

6.42.5.3 Access/Store keys by Crypto Engine

The key can be generated by crypto engine and located in Key Store. User can control PRNG to generate random key or operate the ECDH/ECDSA flow with PRNG and ECC, and user no need to control the programming interface of Key Store. The Key Store supports RSA to store the middle data for CRT and SCAP mode to improve decryption time.

There are policy restrictions for the crypto engine:

1. If key is created by crypto engine and owner is set to CPU, the key is readable; If key is created by crypto engine and owner is not set to CPU, the key is un-readable. User can erase key (in SRAM) or revoke key(in SRAM/Flash/OTP) when key is no longer used. Table 6.42-5 shows all situations of CPU and Crypto can access key or not.
2. The RSA key can store in SRAM and Flash, the RSA middle data can be stored in SRAM only. CPU cannot read out RSA middle data.
3. The size of HMAC key in Key Store is block size (i.e., HMAC-SHA512 and HMAC-SHA384

are 1024 bits; others are 512 bits).

4. The random key generated from PRNG for ECDH/ECDSA used only can be stored in SRAM.

	CPU Read Key	Crypto Engine Access Key
Key created by CPU	Controlled by READABLE (KS_METADATA[2]) while create key operation	Controlled by OWNER (KS_METADATA[2]) while create key operation
Key created by Crypto Engine	OWNER in Key Control Register of Crypto Engine is set to CPU while Crypto Engine is operating.	Controlled by OWNER in Key Control Register of Crypto Engine while Crypto Engine is operating.

Table 6.42-5 CPU and Crypto Engine Access Key for All Situations

6.42.5.4 Data Remanence Prevention

When data remanence prevention function is enabled, the Key Store will overwrite inverted data into same location.

Data remanence is a physical phenomenon in RAM and is the residual data that remains after delete or erase data. When RAM is power off, the data can remains a period of time. The hacker can use this property to get data. There are various countermeasure, such as clearing, encryption, overwriting and so on. The Key Store uses overwriting as the data remanence prevention method.

The data remanence prevention function is operating after OPMODE(KS_CTL[3:1]) is set to 101 and START(KS_CTL[0]) is set to 1. The Key Store will read out all key and invert key, and then write the inverted key in same location. After data remanence prevention function is finished, RAMINV(KS_STS[8]) will be inverted and IF(KS_STS[0]) will be set. User should to check that BUSY(KS_STS[2]) is set to 0 before starting the data remanence function.

If RAMINV(KS_STS[8]) is set to 1, it means key in SRAM is inverted. User can use the Key Store to operate any function normally no matter what states RAMINV is in.

6.42.5.5 Clear key while tamper detected

Key Store supports to clear all keys if RTC tamper event is detected. This function can be enabled by KSTRIGEN(RTC_SPRCTL[8]). If RTC detects the tamper event, RTC will trigger Key Store to clear key. Key Store receives the trigger signal and start to erase SRAM/Flash keys and revoke OTP keys. When all keys is cleared, INITDONE(KS_STS[7]) will be reset. Key Store must to re-initialization after clear key.

6.42.5.6 Auto Verify

While create key operation, Key Store will check whether key is written completely. When create key operation is enabled, Key Store will read back and compare with KS_KEY0~KS_KET7 registers after program. If key read back is not the same as KS_KEY0~KS_KET7 registers, key will be revoke and EIF will be set. Moreover, the space will decrease and key count will be increase.

6.42.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
KS Base Address: **KS_BA = 0x4003_5000**				
KS_CTL	KS_BA+0x000	R/W	Key Store Control Register	0x0000_0000
KS_METADATA	KS_BA+0x004	R/W	Key Store Metadata Register	0x0000_0000
KS_STS	KS_BA+0x008	R/W	Key Store Status Register	0x0000_0000
KS_REMAIN	KS_BA+0x00C	R	Key Store Remaining Space Register	0x1000_1000
KS_KEY0	KS_BA+0x020	R/W	Key Store Entry Key Word 0 Register	0x0000_0000
KS_KEY1	KS_BA+0x024	R/W	Key Store Entry Key Word 1 Register	0x0000_0000
KS_KEY2	KS_BA+0x028	R/W	Key Store Entry Key Word 2 Register	0x0000_0000
KS_KEY3	KS_BA+0x02C	R/W	Key Store Entry Key Word 3 Register	0x0000_0000
KS_KEY4	KS_BA+0x030	R/W	Key Store Entry Key Word 4 Register	0x0000_0000
KS_KEY5	KS_BA+0x034	R/W	Key Store Entry Key Word 5 Register	0x0000_0000
KS_KEY6	KS_BA+0x038	R/W	Key Store Entry Key Word 6 Register	0x0000_0000
KS_KEY7	KS_BA+0x03C	R/W	Key Store Entry Key Word 7 Register	0x0000_0000
KS_OTPSTS	KS_BA+0x040	R	Key Store OTP Keys Status Register	0x0000_0000
KS_REMKCNT	KS_BA+0x044	R	Key Store Remaining Key Count Register	0x0020_0020

6.42.7 Register Description

Key Store Control Register (KS_CTL)

Register	Offset	R/W	Description	Reset Value
KS_CTL	KS_BA+0x000	R/W	Key Store Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IEN	Reserved						INIT
7	6	5	4	3	2	1	0
CONT	Reserved			OPMODE			START

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**IEN**	**Key Store Interrupt Enable Bit** 0 = Key Store Interrupt Disabled. 1 = Key Store Interrupt Enabled.
[14:9]	**Reserved**	Reserved.
[8]	**INIT**	**Key Store Initialization** User should to check BUSY(KS_STS[2]) is 0, and then write 1 to this bit and START(KS_CTL[0[), Key Store will start to initialization. After Key Store is initialized, INIT will be cleared.
[7]	**CONT**	**Read/Write Key Continue Bit** 0 = Read/Write key operation is not continuous to previous operation. 1 = Read/Write key operation is continuous to previous operation.
[6:4]	**Reserved**	Reserved.
[3:1]	**OPMODE**	**Key Store Operation Mode** 000 = Read operation. 001 = Create operation. 010 = Erase one key operation (only for key in SRAM and OTP). 011 = Erase all keys operation (only for SRAM and Flash). 100 = Revoke key operation. 101 = Data Remanence prevention operation (only for SRAM). 111 = Lock operation (only for OTP). Others = reserved.
[0]	**START**	**Key Store Start Control Bit** 0 = No operation. 1 = Start the operation.

Key Store Metadata Register (KS_METADATA)

Register	Offset	R/W	Description	Reset Value
KS_METADATA	KS_BA+0x004	R/W	Key Store Metadata Register	0x0000_0000

31	30	29	28	27	26	25	24
DST		Reserved				NUMBER	
23	22	21	20	19	18	17	16
NUMBER				Reserved	OWNER		
15	14	13	12	11	10	9	8
Reserved			SIZE				
7	6	5	4	3	2	1	0
Reserved			BS	Reserved	READABLE	Reserved	

Bits	Description	
[31:30]	**DST**	**Key Location Selection Bits** 00 = Key is in SRAM. 01 = Key is in Flash. 10 = Key is in OTP. Others = reserved.
[29:26]	**Reserved**	Reserved.
[25:20]	**NUMBER**	**Key Number** Before read or erase one key operation is started, user should write the key number to be operated. When create operation is finished, user can read these bits to get its key number.
[19]	**Reserved**	Reserved.
[18:16]	**OWNER**	**Key Owner Selection Bits** 000 = AES. 001 = HMAC. 010 = RSA exponent blind key for SCAP(CRYPTO_RSA_CTL[8]) = 1 and CRT(CRYPTO_RSA_CTL[2]) = 0. 011 = RSA middle data, p, q and private key. 100 = ECC. 101 = CPU. Others = reserved.
[15:13]	**Reserved**	Reserved.
[12:8]	**SIZE**	**Key Size Selection Bits** 00000 = 128 bits. 00001 = 163 bits. 00010 = 192 bits. 00011 = 224 bits. 00100 = 233 bits. 00101 = 255 bits.

		00110 = 256 bits. 00111 = 283 bits. 01000 = 384 bits. 01001 = 409 bits. 01010 = 512 bits. 01011 = 521 bits. 01100 = 571 bits. 10000 = 1024 bits. 10001 = 1536 bits. 10010 = 2048 bits. 10011 = 3072 bits. 10100 = 4096 bits. Others = reserved.
[7:5]	**Reserved**	Reserved.
[4]	**BS**	**Booting State Selection Bit** 0 = Set key used at all state. 1 = Set key used at boot loader state 1 (BL1 state).
[3]	**Reserved**	Reserved.
[2]	**READABLE**	**Key Readable Control Bit** 0 = key is un-readable. 1 = key is readable.
[1:0]	**Reserved**	Reserved.

Key Store Status Register (KS_STS)

Register	Offset	R/W	Description	Reset Value
KS_STS	KS_BA+0x008	R/W	Key Store Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						KRVKF	RAMINV
7	6	5	4	3	2	1	0
INITDONE	Reserved		FLASHFULL	SRAMFULL	BUSY	EIF	IF

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**KRVKF**	**Key Store Key Revoked Flag (read only)** If KSPLOCK(COFIG2[15:8]) is locked and mass erase occurs, Key Store will erase SRAM/Flash keys and revoke OTP keys at next initialization. When initialization is finished, KRVKF will be set forever. 0 = All Keys have not been erased/revoked. 1 = All Keys have been erased/revoked.
[8]	**RAMINV**	**Key Store SRAM Invert Status (Read Only)** 0 = Key Store key in SRAM is normal. 1 = Key Store key in SRAM is inverted.
[7]	**INITDONE**	**Key Store Initialization Done Status (Read Only)** 0 = Key Store is un-initialized. 1 = Key Store is initialized.
[6:5]	**Reserved**	Reserved.
[4]	**FLASHFULL**	**Key Storage at Flash Full Status Bit (Read Only)** 0 = Key Storage at Flash is not full. 1 = Key Storage at Flash is full.
[3]	**SRAMFULL**	**Key Storage at SRAM Full Status Bit (Read Only)** 0 = Key Storage at SRAM is not full. 1 = Key Storage at SRAM is full.
[2]	**BUSY**	**Key Store Busy Flag (read only)** 0 = Key Store is idle or finished. 1 = Key Store is busy.
[1]	**EIF**	**Key Store Error Flag** This bit is cleared by writing 1 and it has no effect by writing 0. 0 = No Key Store error. 1 = Key Store error interrupt.

[0]	**IF**	**Key Store Finish Interrupt Flag** This bit is cleared by writing 1 and it has no effect by writing 0. 0 = No Key Store interrupt. 1 = Key Store operation done interrupt.

Key Store Remaining Space Register (KS_REMAIN)

Register	Offset	R/W	Description	Reset Value
KS_REMAIN	KS_BA+0x00C	R	Key Store Remaining Space Register	0x1000_1000

31	30	29	28	27	26	25	24
Reserved			**FRMNG**				
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FRMNG							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**RRMNG**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RRMNG							

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:16]	**FRMNG**	**Key Store Flash Remaining Space** The FRMNG shows the remaining byte count space for Flash.
[15:13]	**Reserved**	Reserved.
[12:0]	**RRMNG**	**Key Store SRAM Remaining Space** The RRMNG shows the remaining byte count space for SRAM.

Key Store Entry Key Register (KS_KEY)

Register	Offset	R/W	Description	Reset Value
KS_KEY0	KS_BA+0x020	R/W	Key Store Entry Key Word 0 Register	0x0000_0000
KS_KEY1	KS_BA+0x024	R/W	Key Store Entry Key Word 1 Register	0x0000_0000
KS_KEY2	KS_BA+0x028	R/W	Key Store Entry Key Word 2 Register	0x0000_0000
KS_KEY3	KS_BA+0x02C	R/W	Key Store Entry Key Word 3 Register	0x0000_0000
KS_KEY4	KS_BA+0x030	R/W	Key Store Entry Key Word 4 Register	0x0000_0000
KS_KEY5	KS_BA+0x034	R/W	Key Store Entry Key Word 5 Register	0x0000_0000
KS_KEY6	KS_BA+0x038	R/W	Key Store Entry Key Word 6 Register	0x0000_0000
KS_KEY7	KS_BA+0x03C	R/W	Key Store Entry Key Word 7 Register	0x0000_0000

31	30	29	28	27	26	25	24
KEY							
23	22	21	20	19	18	17	16
KEY							
15	14	13	12	11	10	9	8
KEY							
7	6	5	4	3	2	1	0
KEY							

Bits	Description	
[31:0]	**KEY**	**Key Data** The register will be cleared if the Key Store executes the write operation or CPU completes the reading key.

Key Store OTP Keys Status Register (KS_OTPSTS)

Register	Offset	R/W	Description	Reset Value
KS_OTPSTS	KS_BA+0x040	R	Key Store OTP Keys Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
KEY7	KEY6	KEY5	KEY4	KEY3	KEY2	KEY1	KEY0

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**KEY7**	**OTP Key 7 Used Status** 0 = OTP key 7 is unused. 1 = OTP key 7 is used.
[6]	**KEY6**	**OTP Key 6 Used Status** 0 = OTP key 6 is unused. 1 = OTP key 6 is used.
[5]	**KEY5**	**OTP Key 5 Used Status** 0 = OTP key 5 is unused. 1 = OTP key 5 is used.
[4]	**KEY4**	**OTP Key 4 Used Status** 0 = OTP key 4 is unused. 1 = OTP key 4 is used.
[3]	**KEY3**	**OTP Key 3 Used Status** 0 = OTP key 3 is unused. 1 = OTP key 3 is used.
[2]	**KEY2**	**OTP Key 2 Used Status** 0 = OTP key 2 is unused. 1 = OTP key 2 is used.
[1]	**KEY1**	**OTP Key 1 Used Status** 0 = OTP key 1 is unused. 1 = OTP key 1 is used.
[0]	**KEY0**	**OTP Key 0 Used Status** 0 = OTP key 0 is unused. 1 = OTP key 0 is used.

Key Store Remaining Key Count Register (KS_REMKCNT)

Register	Offset	R/W	Description	Reset Value
KS_REMKCNT	KS_BA+0x044	R	Key Store Remaining Key Count Register	0x0020_0020

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		FRMKCNT					
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		RRMKCNT					

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21:16]	**FRMKCNT**	**Key Store Flash Remaining Key Count** The FRMKCNT shows the remaining key count for Flash.
[15:6]	**Reserved**	Reserved.
[5:0]	**RRMKCNT**	**Key Store SRAM Remaining Key Count** The RRMKCNT shows the remaining key count for SRAM.

6.43 Keypad Interface (KPI)

6.43.1 Overview

The Keypad Interface (**KPI**) is an APB slave with configurable minimum 2-row up to 6-row scan output and minimum 1-column up to 8-column scan input. Any keys in the array pressed or released are de-bounced and generate an interrupt.

The KPI supports release multiple keys and press multiple keys scan interrupt. The interrupt is generated whenever it detects any key in the keypad pressing or releasing. User can know the interrupt source by querying KPI_STATUS register.

6.43.2 Features

- Matrix keypad interface (maximum 6x8 array, and minimum 2x1array)
- Programmable de-bounce time
- Generates interrupt and updates all the keys(maximum 48 keys, minimum 2 keys) information (press/release) every time the user pressing or releasing

6.43.3 Block Diagram

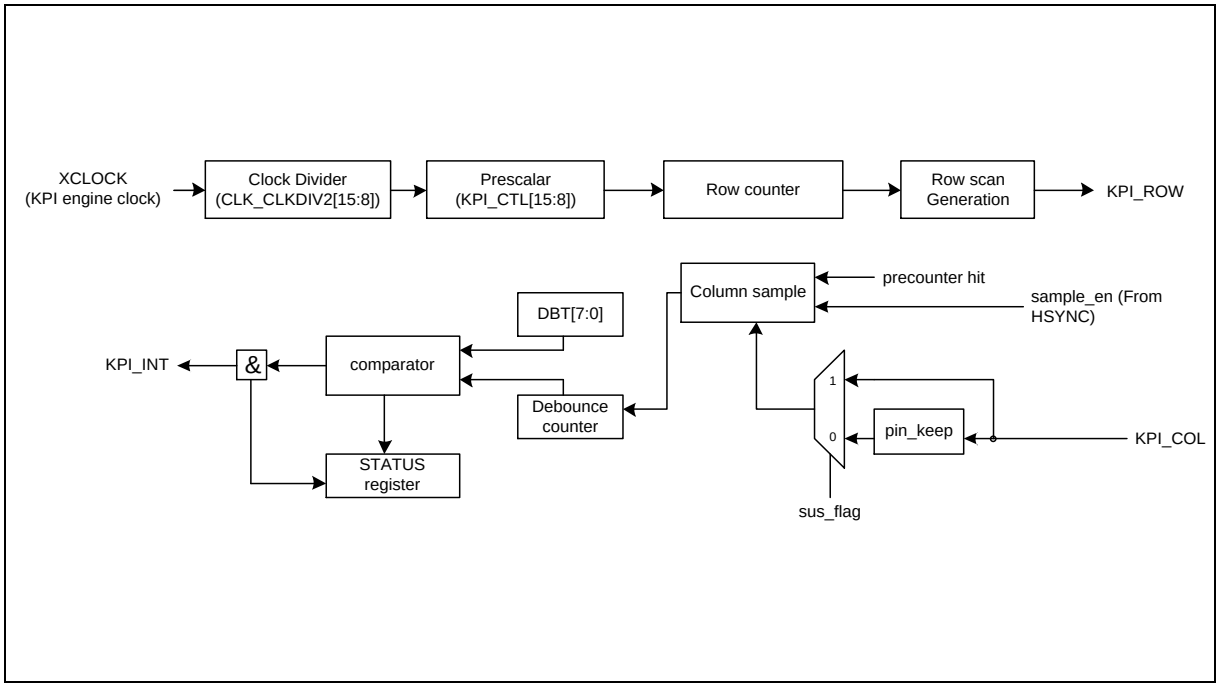


Figure 6.43-1 Keypad Controller Block Diagram

6.43.4 Basic Configuration

- Clock source Configuration
  - The KPI peripheral clock can be enabled in KPICKEN (CLK_APBCLK2[0]). The KPI engine clock source is selected by KPISEL (CLK_CLKSEL3[7:6]) and KPI engine clock divider is determined by KPIDIV (CLK_CLKDIV2[15:8]).
- Reset Configuration
  - The KPI reset can be enabled in KPIRST (SYS_IPRST3[0]).

6.43.5 Functional Description

Keypads are often used as a primary input device for embedded microcontrollers. The keypads actually consist of a number of switches, connected in a row/column arrangement as shown in Figure 6.43-2.

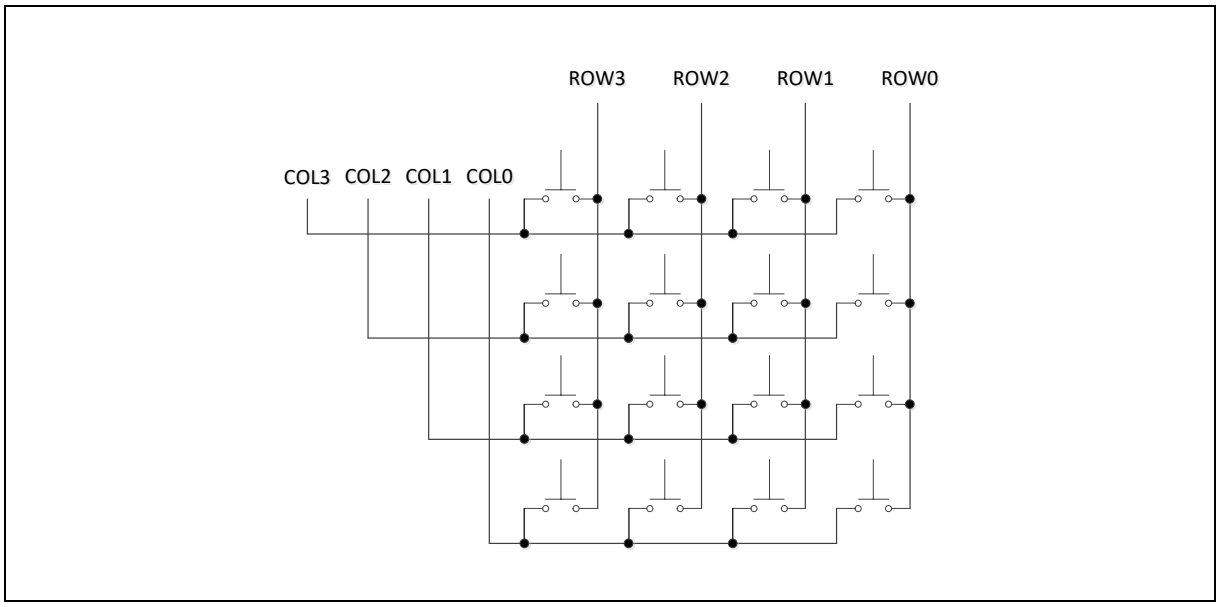


Figure 6.43-2 Keypad Connection

In order for the microcontroller to scan the keypad, it outputs a nibble to force one (only one) of the rows low and then reads the columns to see if any buttons in that column have been pressed. The columns are pulled up by the internal weak pull-up resistor.

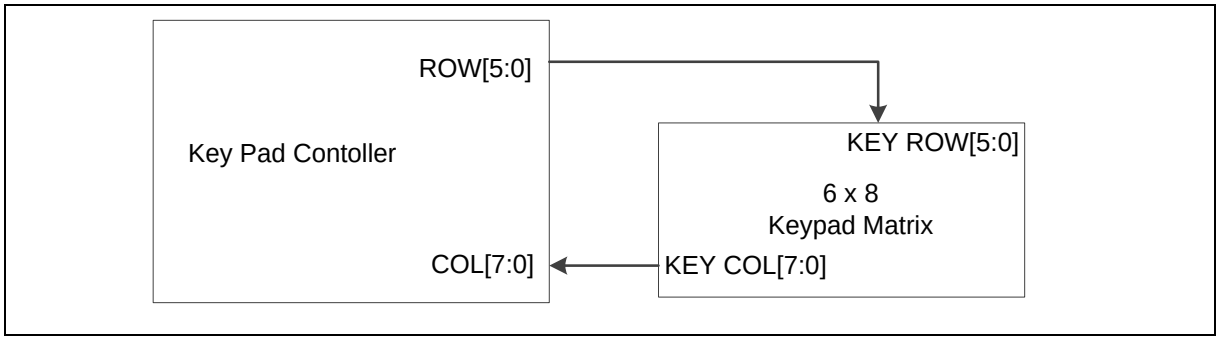


Figure 6.43-3 Keypad Interface

Consequently, as long as no buttons are pressed, the microcontroller sees a logic high on each of the pins attached to the keypad columns. The nibble driven onto the rows always contains only a single 0. The only way the microcontroller can find a 0 on any column pin is for the keypad button to be pressed that connects the row set to 0 to a row. The controller knows which column is at a 0-level and which column reads 0, allowing it to determine which key is pressed. The keypad scan timing is shown in Figure 6.43-4, and the key array scan delay time is shown in Figure 6.43-5.

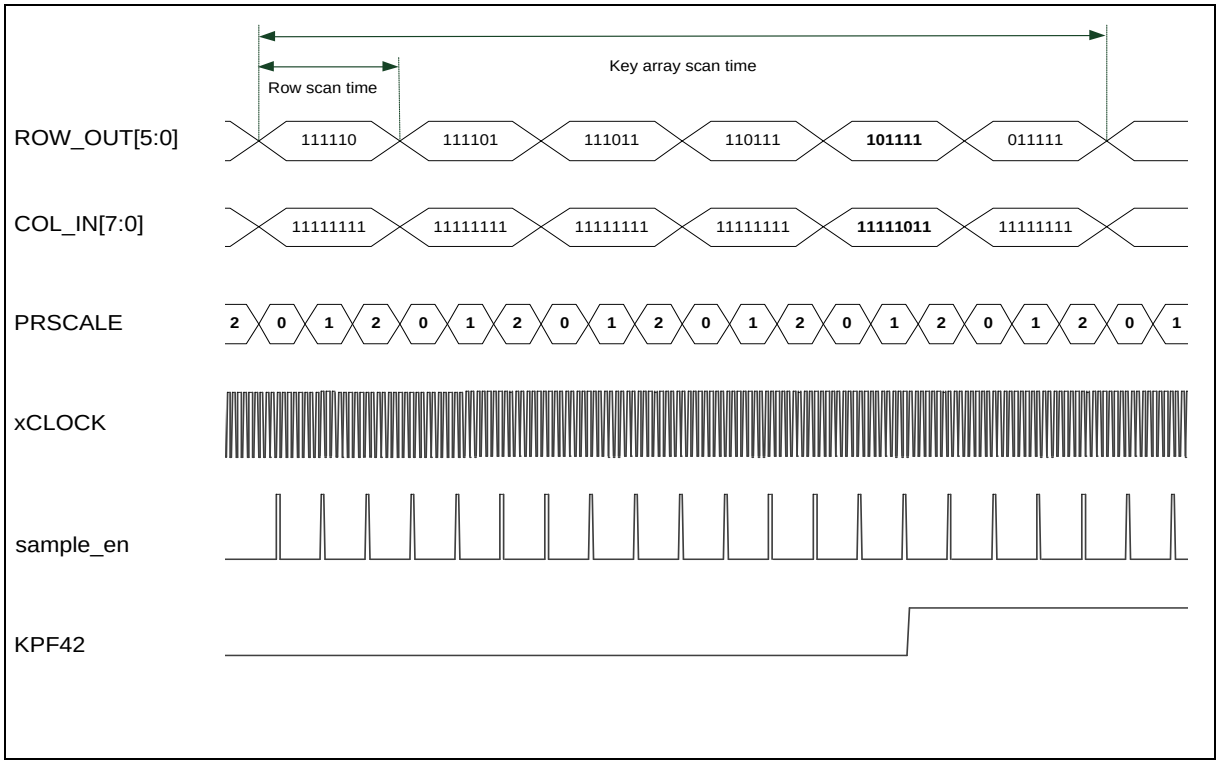


Figure 6.43-46 x8 Keypad Scan Timing Diagram

Row scan time is how long it takes to scan one row take. Key array scan time is how long it takes to scan all rows.

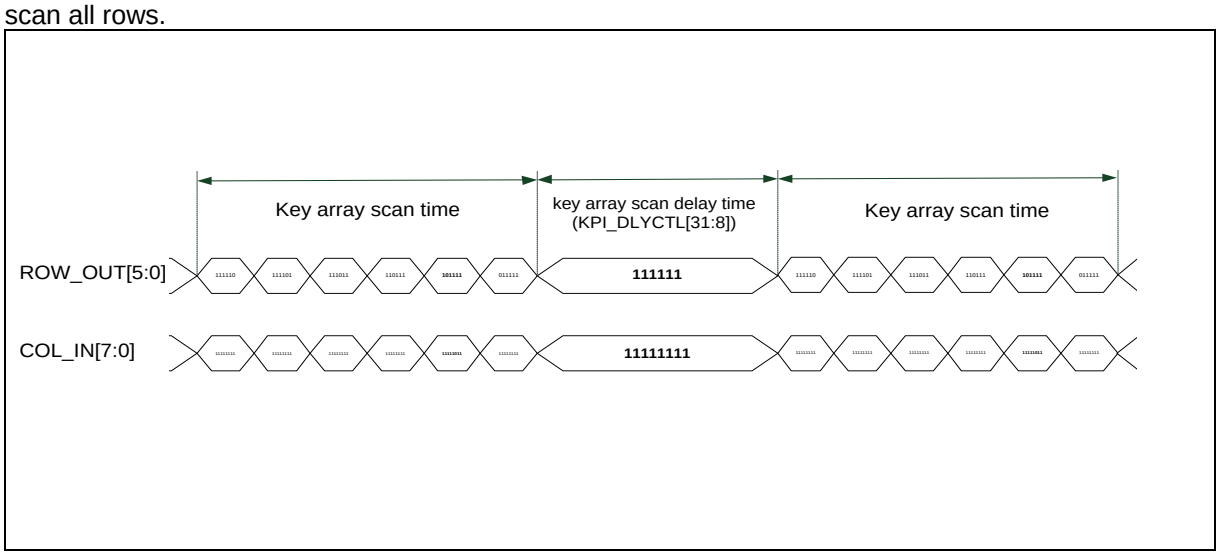


Figure 6.43-5 Key Array Scan Delay Time Diagram

6.43.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
KPI Base Address: **KPI_BA = 0x400C_2000**				
KPI_CTL	KPI_BA+0x000	R/W	Keypad Control Register	0x0003_0000
KPI_STATUS	KPI_BA+0x008	R/W	Keypad Status Register	0x0000_0000
KPI_KST0	KPI_BA+0x010	R	Keypad State Register 0	0xFFFF_FFFF
KPI_KST1	KPI_BA+0x014	R	Keypad State Register 1	0x0000_FFFF
KPI_KPF0	KPI_BA+0x018	R/W	Lower 32 Key Press Flag Register 0	0x0000_0000
KPI_KPF1	KPI_BA+0x01C	R/W	Upper 32 Key Press Flag Register 1	0x0000_0000
KPI_KRF0	KPI_BA+0x020	R/W	Lower 32 Key Release Flag Register 0	0x0000_0000
KPI_KRF1	KPI_BA+0x024	R/W	Upper 32 Key Release Flag Register 1	0x0000_0000
KPI_DLYCTL	KPI_BA+0x028	R/W	Delay Control Register	0x0000_001F

6.43.7 Register Description

Keypad Configuration Register (KPI_CTL)

Register	Offset	R/W	Description	Reset Value
KPI_CTL	KPI_BA+0x000	R/W	Keypad Control Register	0x0003_0000

31	30	29	28	27	26	25	24
Reserved	KROW			Reserved	KCOL		
23	22	21	20	19	18	17	16
ROWDLY		Reserved		DBCLKSEL			
15	14	13	12	11	10	9	8
PSC							
7	6	5	4	3	2	1	0
Reserved		DBCT		KIEN	KRIEN	KPIEN	KPEN

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**KROW**	**Keypad Matrix ROW Number** The keypad matrix is set by ROW x COL. The ROW number can be set 2 to 6. 000 = reserved. 001 = 2. 010 = 3. 011 = 4. 100 = 5. 101 = 6. 110 = Reserved. 111 = Reserved.
[27]	**Reserved**	Reserved.
[26:24]	**KCOL**	**Keypad Matrix COL Number** The keypad matrix is set by ROW x COL. The COL number can be set 1 to 8. 000 = 1. 001 = 2. 010 = 3. 011 = 4. 100 = 5. 101 = 6. 110 = 7. 111 = 8.

[23:22]	**ROWDLY**	**Scan Row Delay** Setting delay cycle when row change, for avoid KPI from detecting wrong key. 00 = 4 KPI engine clock cycle. 01 = 8 KPI engine clock cycle. 10 = 16 KPI engine clock cycle. 11 = 32 KPI engine clock cycle. **Note:** Scan row delay cycle < debounce sampling cycle. Row scan time > scan row delay cycle + (2 * debounce sampling cycle) + 1 xclock cycle (change row) + 2 xclock cycle (cross clock domain).
[21:20]	**Reserved**	Reserved.
[19:16]	**DBCLKSEL**	**Scan in De-bounce Sampling Cycle Selection** 0000 = Reserved. 0001 = Reserved. 0010 = Reserved. 0011 = Sample interrupt input once per 8 clocks. 0100 = Sample interrupt input once per 16 clocks. 0101 = Sample interrupt input once per 32 clocks. 0110 = Sample interrupt input once per 64 clocks. 0111 = Sample interrupt input once per 128 clocks. 1000 = Sample interrupt input once per 256 clocks. 1001 = Sample interrupt input once per 512 clocks. 1010 = Sample interrupt input once per 1024 clocks. 1011 = Sample interrupt input once per 2048 clocks. 1100 = Sample interrupt input once per 4096 clocks. 1101 = Sample interrupt input once per 8192 clocks. 1110 = reserved. 1111 = reserved. **Note:** Scan row delay cycle < debounce sampling cycle. Row scan time > scan row delay cycle + (2 * debounce sampling cycle) + 1 xclock cycle (change row) + 2 xclock cycle (cross clock domain). Row scan time = xCLOCK x PSC x PSCDIV. For example, if debounce sampling cycle chooses 1024 xclock, and scan row delay cycle choose 8 xclock, row scan time should choose larger than (8+2048+3) xclock, Suppose PSCDIV = 0x1F, then prescale = 65 (2059/32 = 64.35).

[15:8]	**PSC**	**Row Scan Cycle Pre-scale Value** This value is used to pre-scale row scan cycle. The pre-scale counter is clocked by the divided crystal clock, xCLOCK. The divided number is from 1 to 256. For example, if the crystal clock is 1 MHz then the xCLOCK period is 1us. If the keypad matric is 3x3 then Each row scan time = xCLOCK x PSC x PSCDIV. Key array scan time = Each row scan time x ROWS. Example scan time for PRESCALE = 0x40, and PSCDIV = 0x1F. Each row scan time = 1us x 65 x 32 = 2.08ms. Scan time = 2.08 x 3 = 6.24ms. **Note:** When PSC is determined, De-bounce sampling cycle should not exceed the half of (PSC x PSCDIV), in the above example, and if scan row delay cycle is 4 xclock The maximum DBCLKSEL should be 1024 xCLOCK, bouncing time is 1ms.
[7:6]	**Reserved**	Reserved.
[5:4]	**DBCT**	**De-bounce Cycle Time** For keypad debounce, keypad will generate an interrupt when key press or key release continued n * key array scan time. 00 = n=1. 01 = n=2. 10 = n=3. 11 = n=4. **Note:** It would need more time to indicate key press and release event when users selected more debounce cycle time.
[3]	**KIEN**	**Key Interrupt Enable** 0 = Keypad interrupt Disabled. 1 = Keypad interrupt Enabled. **Note:** The bit will be reset when KPI reset occurs.
[2]	**KRIEN**	**Key Release Interrupt Enable** The keypad controller will generate an interrupt when the controller detects keypad status changes from press to release. 0 = Keypad release interrupt Disabled. 1 = Keypad release interrupt Enabled. **Note:** The bit will be reset when KPI reset occurs.
[1]	**KPIEN**	**Key Press Interrupt Enable** The keypad controller will generate an interrupt when the controller detects any effective key press. 0 = Keypad press interrupt Disabled. 1 = Keypad press interrupt Enabled. **Note:** The bit will be reset when KPI reset occurs.
[0]	**KPEN**	**Keypad Enable** Setting this bit high enables the key scan function. 0 = Keypad scan Disabled. 1 = Keypad scan Enabled.

Keypad Status Register (KPI_STATUS)

Register	Offset	R/W	Description	Reset Value
KPI_STATUS	KPI_BA+0x008	R/W	Keypad Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			KPIF	KRIF	KIF	Reserved	

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**KPIF**	**Key Press Interrupt Flag** This bit indicates that some keys (one or multiple key) have been pressed. When READ: 0 = No key press. 1 = At least one key press. **Note:** To clear KPIF, software must clear each pressing flag that are shown on KPI_KPF0/1 registers.
[3]	**KRIF**	**Key Release Interrupt Flag** This bit indicates that some keys (one or multiple key) have been released. When READ: 0 = No key release. 1 = At least one key release. **Note:** To clear KRIF, software must clear each releasing flag that are shown on KPI_KRF0/1 registers.
[2]	**KIF**	**Key Interrupt Flag** This bit indicates the key scan interrupt is active when any key press or key release. When READ: 0 = No reset. 1 = Key press/Key release interrupt occurred. **Note:** To clear KIF, software must clear KPIF, KRIF and TKRIF.
[1:0]	**Reserved**	Reserved.

Keypad State Register 0 (KPI_KST0)

Register	Offset	R/W	Description	Reset Value
KPI_KST0	KPI_BA+0x010	R	Keypad State Register 0	0xFFFF_FFFF

31	30	29	28	27	26	25	24
KST37	KST36	KST35	KST34	KST33	KST32	KST31	KST30
23	22	21	20	19	18	17	16
KST27	KST26	KST25	KST24	KST23	KST22	KST21	KST20
15	14	13	12	11	10	9	8
KST17	KST16	KST15	KST14	KST13	KST12	KST11	KST10
7	6	5	4	3	2	1	0
KST07	KST06	KST05	KST04	KST03	KST02	KST01	KST00

Bits	Description	
[31:0]	KSTmn	**Key State** m is row number, n is column number. 0 = Key is pressing. 1 = Key is releasing.

Keypad State Register 1 (KPI_KST1)

Register	Offset	R/W	Description	Reset Value
KPI_KST1	KPI_BA+0x014	R	Keypad State Register 1	0x0000_FFFF

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
KEST57	KEST56	KEST55	KEST54	KEST53	KEST52	KEST51	KEST50
7	6	5	4	3	2	1	0
KEST47	KEST46	KEST45	KEST44	KEST43	KEST42	KEST41	KEST40

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**KESTmn**	**Key State** m is row number, n is column number. 0 = Key is pressing. 1 = Key is releasing.

Lower 32 Key Press Event Indicator (KPI_KPF0)

Register	Offset	R/W	Description	Reset Value
KPI_KPF0	KPI_BA+0x018	R/W	Lower 32 Key Press Flag Register 0	0x0000_0000

31	30	29	28	27	26	25	24
KPF37	KPF36	KPF35	KPF34	KPF33	KPF32	KPF31	KPF30
23	22	21	20	19	18	17	16
KPF27	KPF26	KPF25	KPF24	KPF23	KPF22	KPF21	KPF20
15	14	13	12	11	10	9	8
KPF17	KPF16	KPF15	KPF14	KPF13	KPF12	KPF11	KPF10
7	6	5	4	3	2	1	0
KPF07	KPF06	KPF05	KPF04	KPF03	KPF02	KPF01	KPF00

Bits	Description	
[31:0]	KPFmn	**Lower 32 Key Press Flag** m is row number, n is column number. 0 = No key event. 1 = Corresponding key has a high to low event change. **Note:** This bit will be set by hardware, and should be cleared by software writing 1. KPIF (KPI_STATUS[4]) can be cleared by writing 1.

Upper 32 Key Press Event Indicator (KPI_KPF1)

Register	Offset	R/W	Description	Reset Value
KPI_KPF1	KPI_BA+0x01C	R/W	Upper 32 Key Press Flag Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
KPF57	KPF56	KPF55	KPF54	KPF53	KPF52	KPF51	KPF50
7	6	5	4	3	2	1	0
KPF47	KPF46	KPF45	KPF44	KPF43	KPF42	KPF41	KPF40

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**KPEmn**	**Upper 32 Key Press Flag** m is row number, n is column number. 0 = No key event. 1 = Corresponding key has a high to low event change. **Note:** This bit will be set by hardware, and should be cleared by software writing 1.

Lower 32 Key Release Event Indicator (KPI_KRF0)

Register	Offset	R/W	Description	Reset Value
KPI_KRF0	KPI_BA+0x020	R/W	Lower 32 Key Release Flag Register 0	0x0000_0000

31	30	29	28	27	26	25	24
KRF37	KRF36	KRF35	KRF34	KRF33	KRF32	KRF31	KRF30
23	22	21	20	19	18	17	16
KRF27	KRF26	KRF25	KRF24	KRF23	KRF22	KRF21	KRF20
15	14	13	12	11	10	9	8
KRF17	KRF16	KRF15	KRF14	KRF13	KRF12	KRF11	KRF10
7	6	5	4	3	2	1	0
KRF07	KRF06	KRF05	KRF04	KRF03	KRF02	KRF01	KRF00

Bits	Description	
[31:0]	KRFmn	**Lower 32 Key Release Flag** m is row number, n is column number. 0 = No key event. 1 = Corresponding key has a low to high event change. **Note:** This bit will be set by hardware, and should be cleared by software writing 1.

Upper 32 Key Release Event Indicator (KPI_KRF1)

Register	Offset	R/W	Description	Reset Value
KPI_KRF1	KPI_BA+0x024	R/W	Upper 32 Key Release Flag Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
KRF57	KRF56	KRF55	KRF54	KRF53	KRF52	KRF51	KRF50
7	6	5	4	3	2	1	0
KRF47	KRF46	KRF45	KRF44	KRF43	KRF42	KRF41	KRF40

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**KRFmn**	**Upper 32 Key Release Flag** m is row number, n is column number. 0 = No key event. 1 = Corresponding key has a low to high event change. **Note:** This bit will be set by hardware, and should be cleared by software writing 1.

Delay Control Register (KPI_DLYCTL)

Register	Offset	R/W	Description	Reset Value
KPI_DLYCTL	KPI_BA+0x028	R/W	Delay Control Register	0x0000_001F

31	30	29	28	27	26	25	24
SCANDLY							
23	22	21	20	19	18	17	16
SCANDLY							
15	14	13	12	11	10	9	8
SCANDLY							
7	6	5	4	3	2	1	0
PSCDIV							

Bits	Description	
[31:8]	**SCANDLY**	**Key Array Scan Delay** This value is used to insert delay cycle between each key array scan. The key array scan delay counter is clocked by the divided crystal clock, xCLOCK. Key array scan delay time = xCLOCK x SCANDLY. The number of key array scan delay cycle is 0 and from 2 to 2^24. 0 = No delay. Others = others + 1 cycles. **Note:** If the key array scan delay is set to 0, there is no delay between each key array scan. There is no delay 1 cycle situation.
[7:0]	**PSCDIV**	**Pre-scale Divider** This value is used to divide RESCALE that is set in KPI_CTL[15:8]. The prescale divider counter is clocked by the divided crystal clock, xCLOCK. The number is from 1 to 256. For example, if the crystal clock is 1 MHz then the xCLOCK period is 1us. If the keypad matrix is 3x3. Then, Each row scan time = xCLOCK x PSC x PSCDIV. Key array scan time = each row scan time x ROWS. Example scan time for PSC = 0x40, and PSCDIV = 0x1F. Each row scan time = 1us x 65 x 32 = 2.08ms. Scan time = 2.08 x 3 = 6.24ms. **Note:** When PSC (KPI_CTL[15:8]) is determined, De-bounce sampling cycle should not exceed the half of (PSC x PSCDIV), in the above example, and if scan row delay cycle is 4 xclock. The maximum DBCLKSEL(KPI_CTL[19:16]) should be 1024 x clock, bouncing time is 1ms.

6.44 Enhanced 12-bit Analog-to-Digital Converter (EADC)

6.44.1 Overview

The chip contains three 12-bit successive approximation analog-to-digital converters (SAR EADC converter) with 16 external input channels and 3 internal channels. The EADC converter can be started by software trigger, EPWM0/1 triggers, BPWM0/1 triggers, timer0~3 overflow pulse triggers, ADINT0 interrupt EOC (End of conversion) and ADINT1 interrupt EOC pulse trigger and external pin (EADC0_ST, EADC1_ST, EADC2_ST for EADC0~2 respectively) input signal.

6.44.2 Features

- Analog input voltage range: 0~ V_{REF} (Max to 3.6V)
- Reference voltage from V_{REF} pin
- 12-bit resolution and 10-bit accuracy is guaranteed
- Up to 16 single-end analog external input channels or 8 pair differential analog input channels
- Up to 3 internal channels: band-gap voltage (V_{BG}), temperature sensor (V_{TEMP}), and Battery power divided by 4 ($V_{BAT}/4$)
- Four EADC interrupts (ADINT0~3) with individual interrupt vector addresses for EADC
- Maximum EADC clock frequency is 95 MHz for EADC
- Up to 5 MSPS conversion rate for EADC
- Supports calibration function and calibration interrupt
- Supports internal reference voltage V_{REF}: 1.6V, 2V, 2.5V and 3V.
- Supports power-down mode
- Up to 19 sample modules
  - Sample modules0~15 is configurable for EADC converter channel (EADC_CH0~15) and trigger source for each EADC
  - Sample module 16~18 is fixed for channel 16, 17, 18 input sources as band-gap voltage, temperature sensor, and Battery power divided by 4 ($V_{BAT}/4$)
  - Double buffer for sample control logic module 0~3
  - Configurable sampling time for each sample module
  - Conversion results are held in 19 data registers with valid and overrun indicators.
- Averaging (2^n times, n=0~8) to support up to 12-bit result and over-sampling, or called Accumulation, (2^n times, n=0~8) to support up to 16-bit result
- Supports conversion results left-alignment
- Any EADC conversion of each EADC can be started by:
  - Write 1 to SWTRGn (EADC_SWTRG[n], n = 0~18)
  - External pin EADC0_ST, EADC1_ST, EADC2_ST for EADC0~2 respectively
  - Timer0~3 overflow pulse triggers
  - ADINT0 and ADINT1 interrupt EOC (End of conversion) pulse triggers
  - EPWM/BPWM triggers
  - ADINT0 and ADINT1 interrupt SOC (Start of conversion) pulse trigger with delay counter

- Supports PDMA transfer
- Conversion Result Monitor by Compare Mode
- Supports triple ADC simultaneous mode

6.44.3 Block Diagram

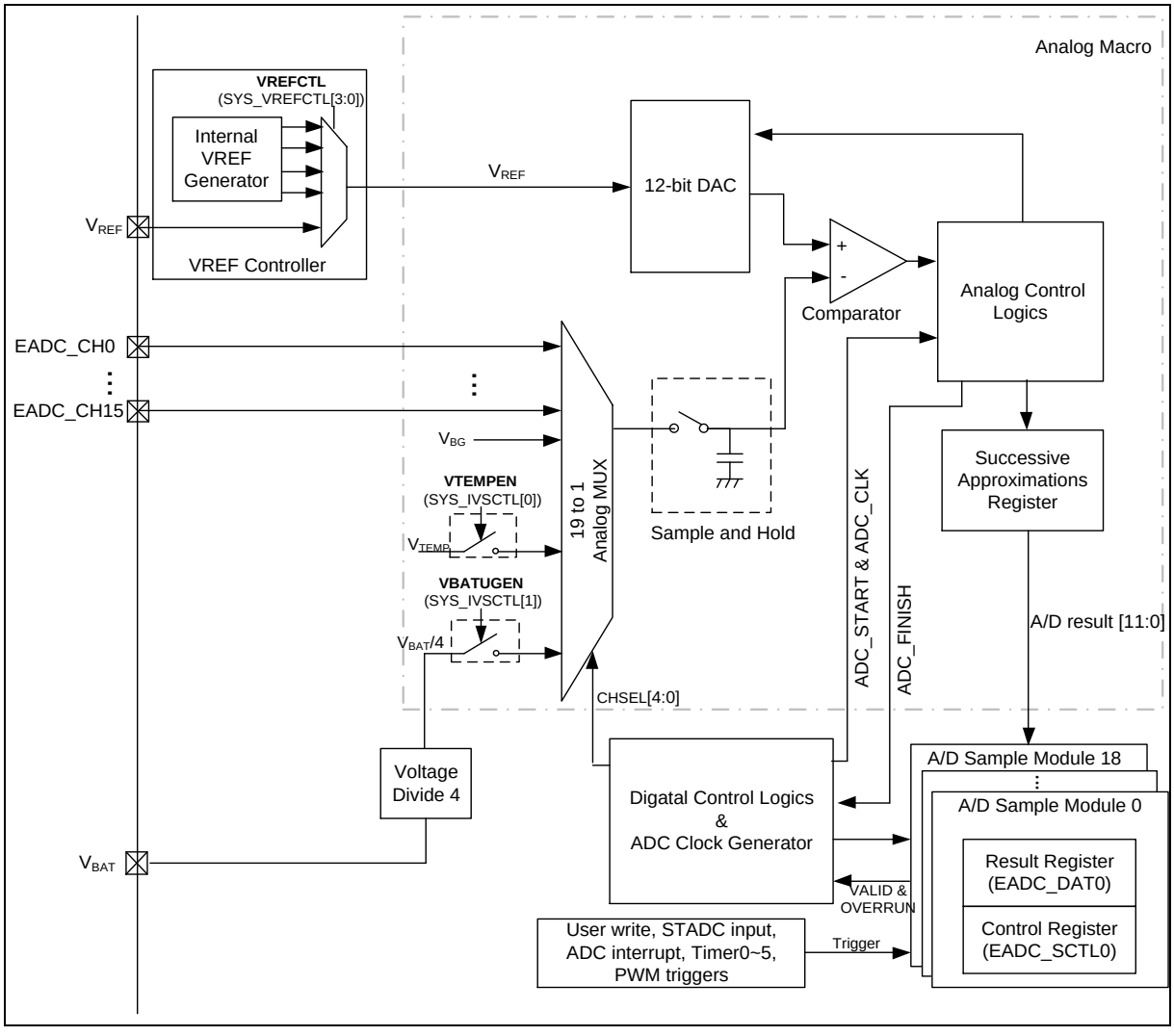


Figure 6.44-1 EADC Converter Block Diagram

6.44.4 Basic Configuration

- EADC0 Clock Source Configuration
  - Select the clock divider number on EADCDIV (CLK_CLKDIV0[23:16])
  - Enable EADC peripheral clock in EADC0CKEN (CLK_APBCLK0[28]).
- Reset Configuration
  - Reset EADC controller in EADCRST (SYS_IPRST1[28]).

- EADC1 Clock Source Configuration
  - Select the clock divider number on EADC1DIV (CLK_CLKDIV2[31:24])
  - Enable EADC1 peripheral clock in EADC1CKEN (CLK_APBCLK1[31]).
- Reset Configuration
  - Reset EADC1 controller in EADC1RST (SYS_IPRST2[31]).
- EADC2 Clock Source Configuration
  - Select the clock divider number on EADC2DIV (CLK_CLKDIV5[31:24])
  - Enable EADC2 peripheral clock in EADC2CKEN (CLK_APBCLK2[6]).
- Reset Configuration
  - Reset EADC2 controller in EADC2RST (SYS_IPRST3[6]).
- EADC2 Pin configuration

6.44.5 Functional Description

The EADC controller consists of a 19 channel analog switch, 19 sample modules and a 12-bit successive approximation analog-to-digital converter. The EADC operation is based on sample module 0~19, and sample module 0~15 has its configuration to decide which trigger source to start the conversion, which channel to convert. Sample module 0~15 can be configured to EADC0_CH0~15 channel, and different trigger source. It provides user a flexible means to get the over-sampling results. The sample module 0~3 and sample module 4~15 are shows as follows. Note that sample 16~18 is fixed for internal channel of band-gap voltage (V_{BG}), temperature sensor, and Battery power divided by 4 ($V_{BAT}/4$).

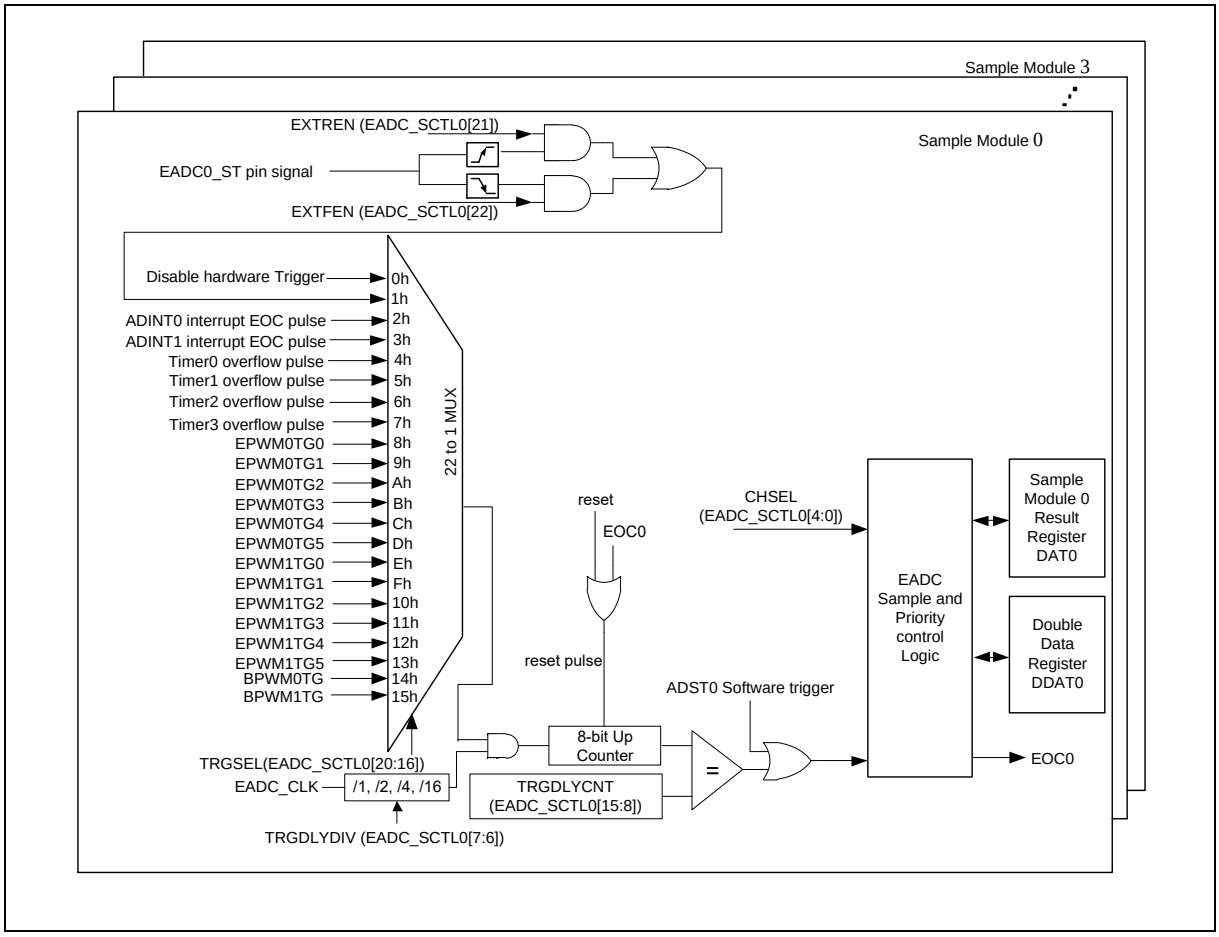


Figure 6.44-2 Sample Module 0~3 Block Diagram

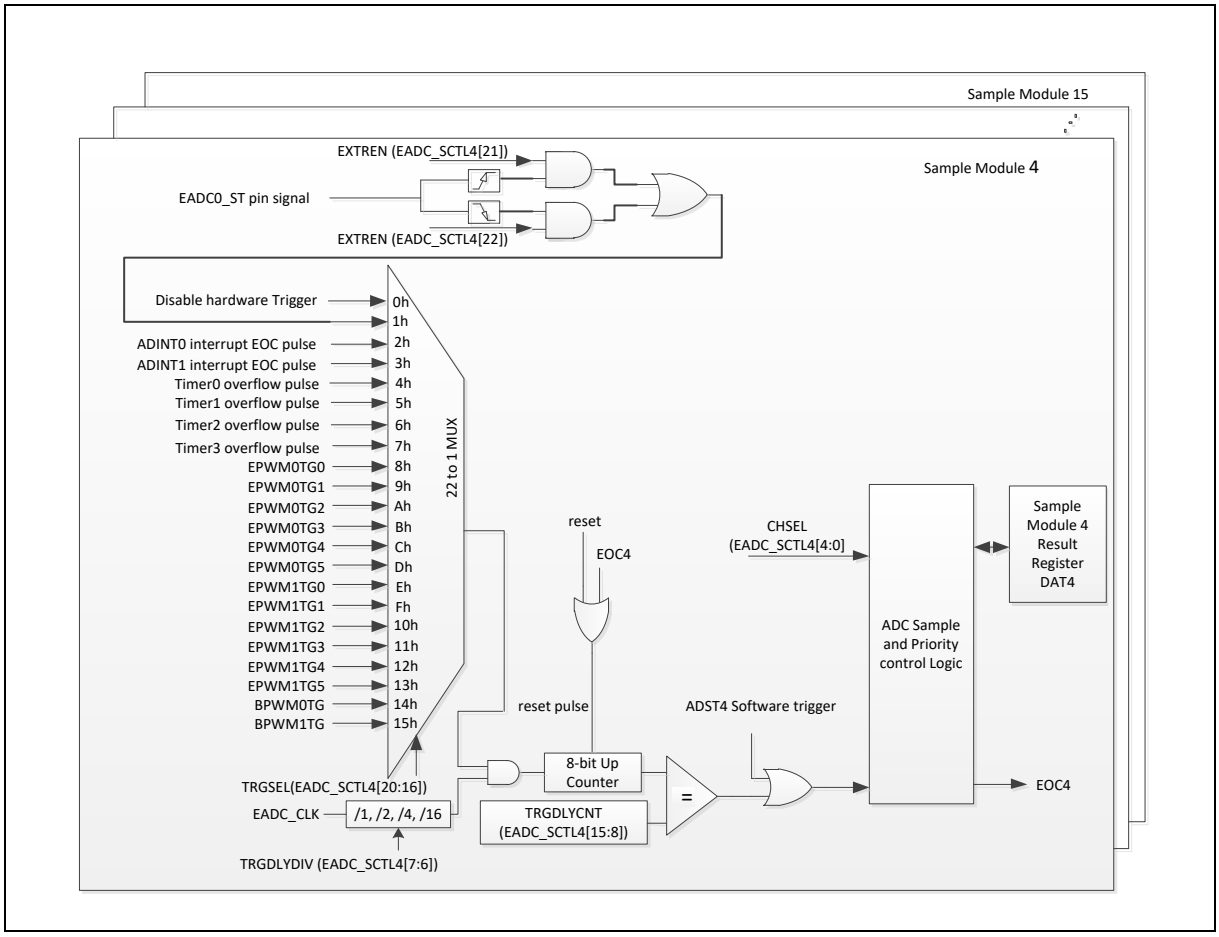


Figure 6.44-3 Sample Module 4~15 Block Diagram

Sample module 16~18 can convert internal channel (V_{BG}, V_{TEMP}, $V_{BAT}/4$) and can be triggered by user write SWTRGn (EADC_SWTRG[n], n = 16~18). Figure 6.44-4 shows the sample module 16~18.

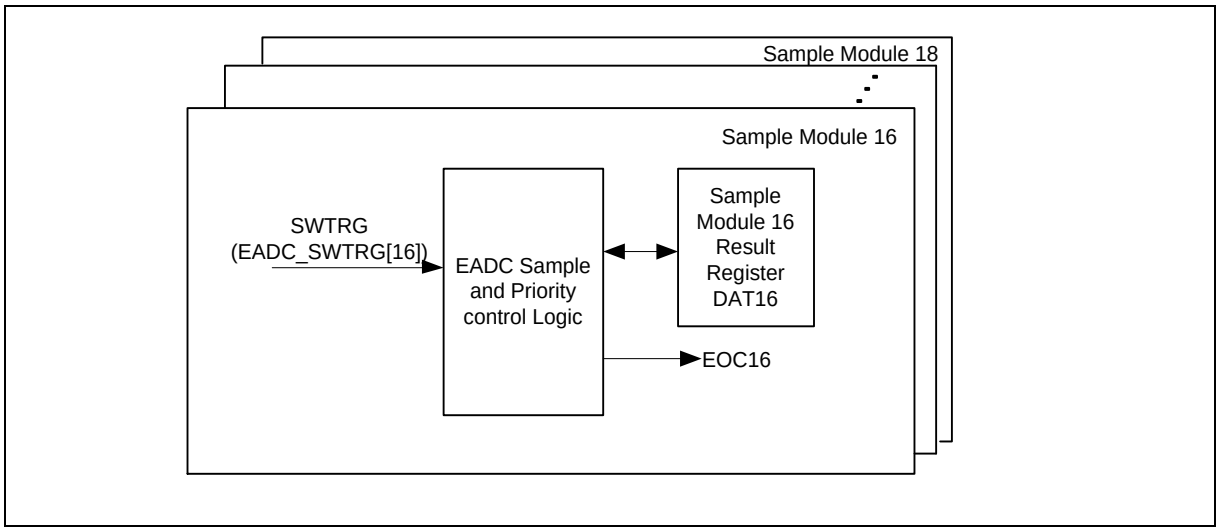


Figure 6.44-4 Sample Module 16~18 Block Diagram

The EADC conversion trigger sources in sample module 0~15 are listed below:

- Write 1 to SWTRGn (EADC_SWTRG[n], n = 0~15)
- External pin EADC0_ST, EADC1_ST, EADC2_ST for EADC0~2 respectively
- Timer0~3 overflow pulse triggers
- ADINT0, ADINT1 EADC interrupt EOC (End of conversion) pulse triggers
- EPWM/BPWM triggers

The ADINT0 or ADINT1 interrupt pulses are generated whenever the specific sample module EADC EOC (End of conversion) pulse is generated. ADINT0 or ADINT1 interrupt pulse triggers can be fed back to trigger another EADC conversion, and is useful if a continuous scan conversion is needed.

6.44.5.1 EADC Clock Generator

The maximum EADC clock frequency is up to 95 MHz and the maximum sampling rate is up to 5 MSPS.

The clock control of EADC is shown as Figure 6.44-5. The EADC peripheral clock source is from PCLK1 clock, the EADC clock frequency is divided by an 8-bit pre-scalar with the following formula:

EADC0 clock frequency = (PCLK1) / (EADCDIV (CKL_CLKDIV0[23:16])+1)

EADC1 clock frequency = (PCLK1) / (EADC1DIV (CKL_CLKDIV2[31:24])+1)

EADC2 clock frequency = (PCLK1) / (EADC2DIV (CKL_CLKDIV5[31:24])+1)

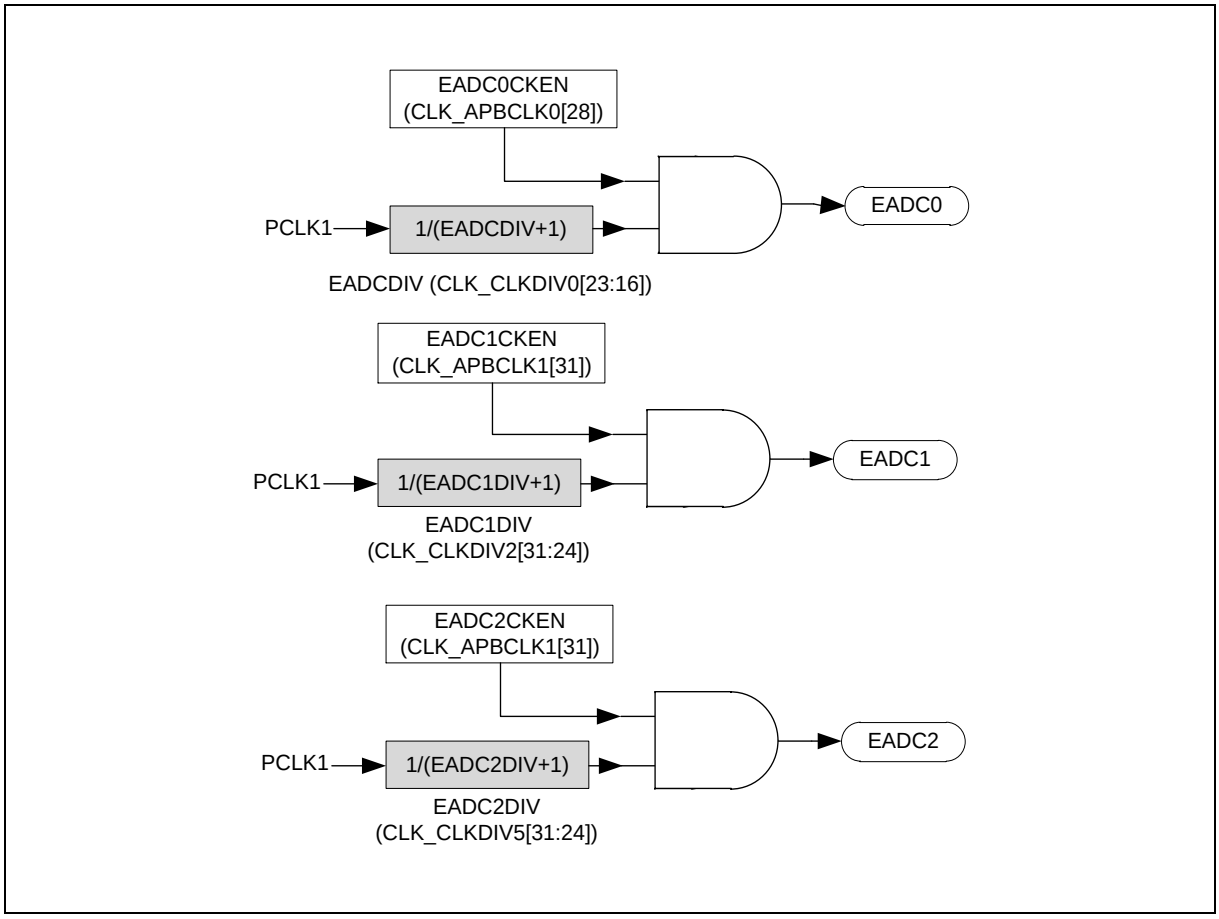


Figure 6.44-5 EADC Clock Control

6.44.5.2 EADC Software Trigger Mode

When a EADC conversion is performed on the sample module specified single channel, the operations are as follows:

1. EADC conversion is started when the SWTRGn (EADC_SWTRG[n], n=0~18) is set to 1 by user or other trigger inputs.
2. When EADC conversion is finished, the 12-bit result is stored in the EADC data register EADC_DATn (n=0~18) corresponding to the sample module.
3. On completion of conversion, the ADIFn (EADC_STATUS2[3:0], n=0~3) is set to 1 and EADC interrupt (ADINTn, n=0~3) is requested if the ADCIENn (EADC_CTL[5:2], n=0~3) bit is set to 1.
4. When EADC conversion ends, the SWTRGn (n=0~18) bit is automatically cleared to 0 and the EADC converter will do another pending conversion.
5. Take EADC0 for example. The timing diagram of a conversion cycle is shown in Figure 6.44-6.

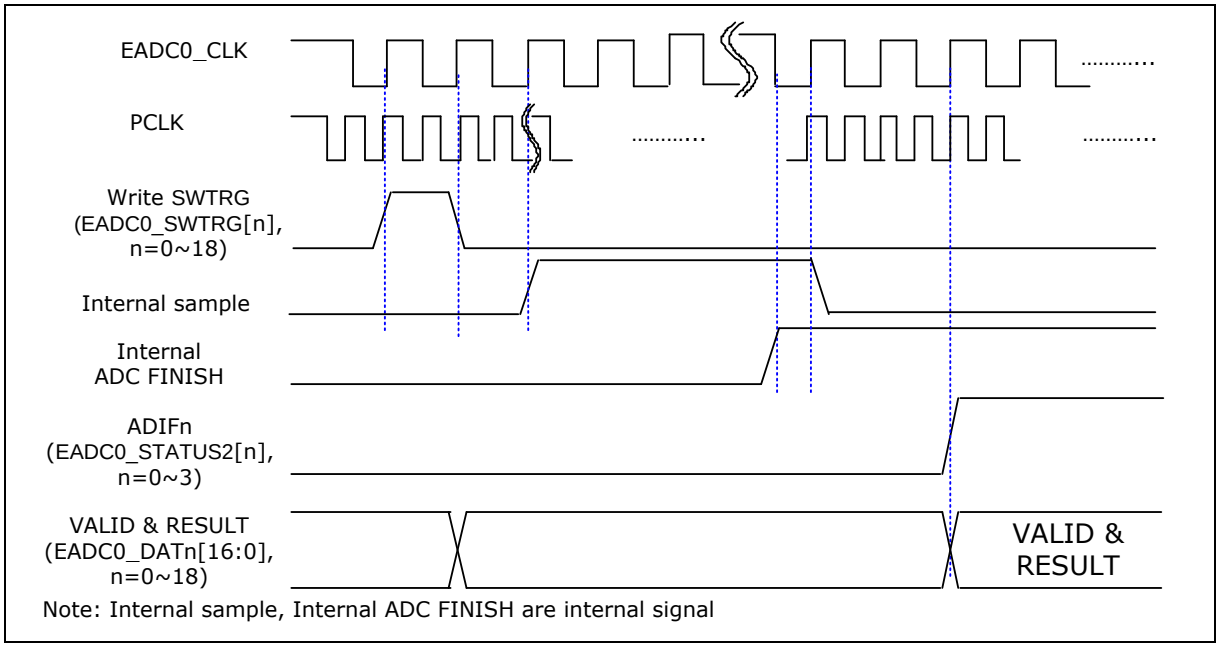


Figure 6.44-6 Example EADC Conversion Timing Diagram, n=0~18

If more than one sample module is enabled to convert analog signal, the sample module specified channel with highest priority is firstly converted and other enabled sample module will be pended. The lower number sample module has higher priority. The sample module 0 is highest priority and the sample module 18 is lowest priority.

6.44.5.3 EADC Conversion Priority

There is a priority group converter for determining the conversion order when multiple sample module trigger flags are set at the same time. Sample module with lower number has higher priority than the higher number sample module. The priority of sample module is shown as Figure 6.44-7. When more than one Sample Module are triggered at the same time, the Sample Module with lower number will start to convert first. The other Sample Module will be in the queue and the corresponding pending flag STPF(EADC_PENDSTS[n], n=0~18) are set to 1 by HW. After the Sample Module finish the conversion, STPF(EADC_PENDSTS[n], n=0~18) will be set to 0 automatically. If the Sample Module which is in the queue is triggered once more, the corresponding Overrun Flag SPOVF(EADC_OVSTS[n], n=0~18) will

be set to 1 by HW.

For example, the Sample Module 0, 2, 3, 5 are triggered simultaneously. The input channel of Sample Module 0 will be converted first. Sample Module 2, 3, 5 will be suspended and STPF (EADC_PENDSTS[2], EADC_PENDSTS [3], EADC_PENDSTS [5]) will be set to 1. If Sample Module 5 is trigger once more in the same time, SPOVF(EADC_OVSTS[5]) will be set to 1.

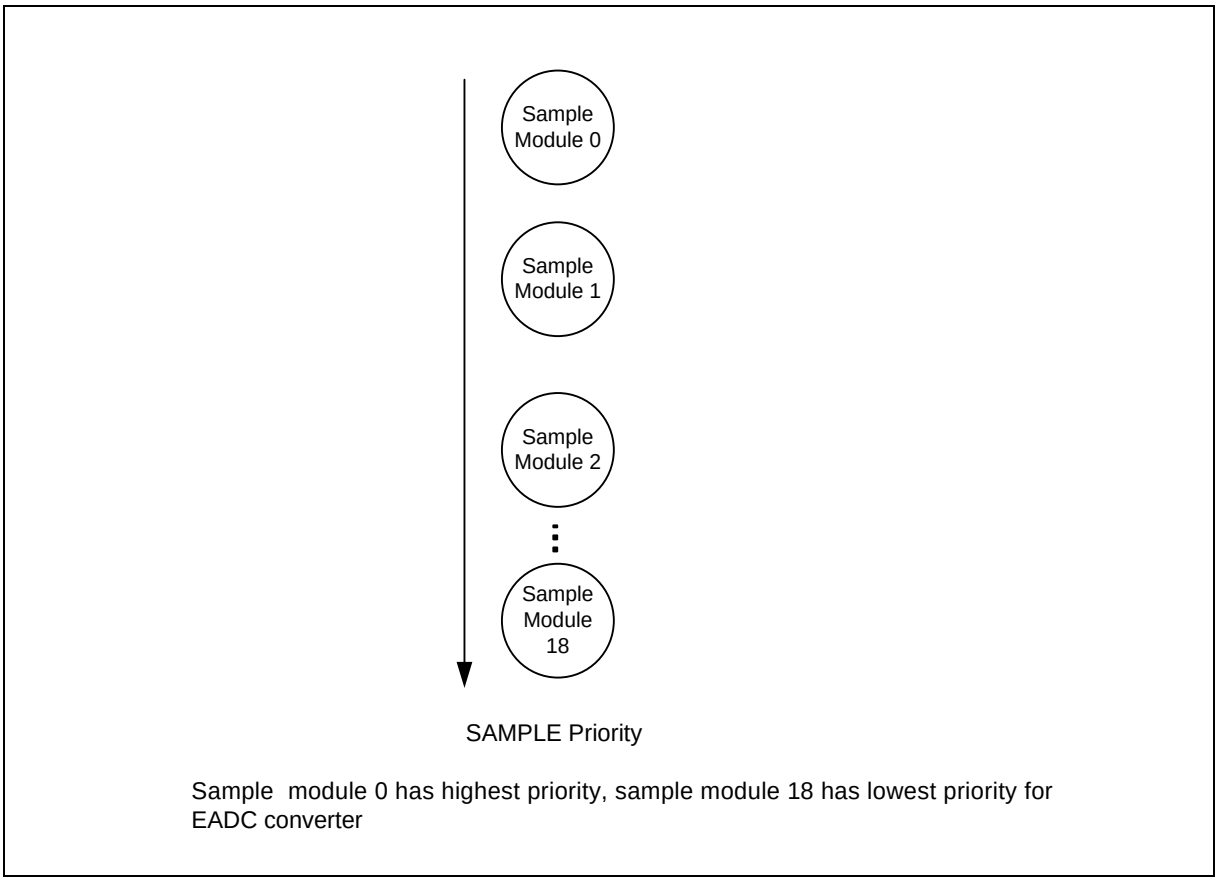


Figure 6.44-7 Sample Module Conversion Priority Arbitrator Diagram

6.44.5.4 Maximum Sampling Frequency Conversion by Software Trigger

If user needs to get maximum sampling frequency, the conversion needs to be executed by the condition as: multiple sample modules, triggered by software, and triggered repeatedly during the last conversion. An example of continuous scan is as follows:

1. Using Module 0~15 to carry out successive conversion. Set CHSEL (EADC_SCTL0~15[4:0]) as one of fast channel (EADC0_CH10~ EADC0_CH15). Set EXTSMPT (EADC_SCTL0~15[31:24]) and TRGDLYCNT (EADC_SCTL0~15[15:8]) as 0x00 to minimize the sampling time.
2. Set SWTRG (EADC_SWTRG[26:0]) as 0xffff to trigger Module 0~15.
3. Wait CURSPL (EADC_STATUS3[4:0]) changes to 0xf which means Module 0~14 have been executed and Module 15 is in the process. Set SWTRG (EADC_SWTRG[26:0]) as 0x7fff to trigger Module 0~14 again for next round.
4. Wait CURSPL (EADC_STATUS3[4:0]) changes to 0x1, set SWTRG (EADC_SWTRG[26:0]) as 0x8000 to trigger Module 15.
5. Repeat Step 3~4 to continue the conversion.

6.44.5.5 EADC Sample Module End of Conversion Interrupt Operation

There are 4 EADC interrupts ADINT0~3, and each of these interrupts has its own interrupt vector address and can be configured to set multiple sample module EOC pulse (sample module 0~26 End of conversion pulses) as its interrupt trigger source. Figure 6.44-8 shows the control logic of interrupts. Take ADINT0 as an example, when ADCIEN0 (EADC_CTL[2]) = 1 and SPLIEn (EADC_INTSRC0[n]) = 1 (n=0~18), the specific module EOC (End of conversion) pulses will set flag ADIF0 (EADC_STATUS2[0]) as 1 and interrupt (ADINT0) will be asserted either.

The interrupt pulses (ADINT0/1) are generated whenever the specific sample module EADC EOC pulse is generated. It also can be the sample module conversion trigger sources, and user can use it to do the EADC continuous scan conversion.

The example of continuous scan triggered by interrupt is as follows:

1. If EADC sample module 2 EOC2 pulse is selected as ADINT0 interrupt trigger SPLIE2 (EADC_INTSRC0[2]) = 1 and ADINT0 is selected as sample module 0, 1, 2 hardware conversion trigger.
2. Set software trigger SWTRG2 (EADC_SWTRG[2]) to 1 to start a sample module 2 EADC conversion, after the conversion completes, it generates an EOC2 pulse signal and ADINT0 interrupt pulse at end of sample module 2 EADC conversion, ADINT0 interrupt pulse will trigger the sample module 0, 1, 2 to start the EADC conversions.
3. ADINT0 interrupt pulse repeats to trigger sample module 0, 1, 2 EADC conversions automatically.
4. Clear TRGSEL (EADC_SCTL2[20:16]) to 0 to disable sample module 2 ADINT0 interrupt pulse hardware trigger, if needs to stop the continuous scan.

Note: The average conversion cycles of continuous scan triggered by interrupt is 19 EADC_CLK. (EADC0~ EADC2 are the same)

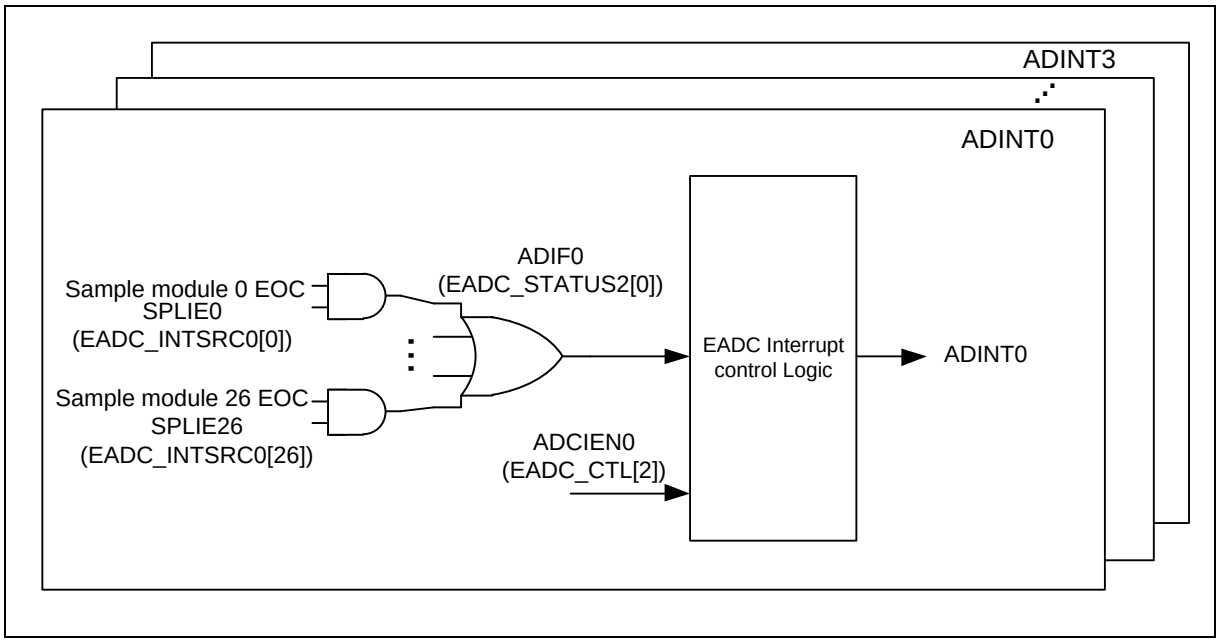


Figure 6.44-8 Specific Sample Module EADC EOC Signal for ADINT0~3 Interrupt

6.44.5.6 EADC Trigger by Timer Trigger and External Pin

There are 4 Timer trigger source and external pin which can configure sample module 0~18 to trigger EADC start when Timer overflow occurs.

6.44.5.7 EADC Start Synchronous with EPWM/BPWM Trigger

Besides user start, ADINT0/1 interrupt pulse, external pin EADC0_ST, EADC1_ST, EADC2_ST and Timer0~3 overflow pulse to start EADC conversion, this device has new feature to allow EPWM/BPWM channels to trigger the EADC start. User may configure EPWM/BPWM trigger types: rising, falling EPWM/BPWM edge or center point of EPWM/BPWM (center-aligned mode only) to trigger EADC start. The device also allows user to configure the amount of delay period to EADC start after hardware detected the external trigger. User can configure the trigger delay time by setting TRGDLYCNT (EADC_SCTLn[15:8], n=0~15) and TRGDLYDIV (EADC_SCTLn[7:6], n=0~15). Figure 6.44-9 shows the programmable delay time for EPWM/BPWM-triggered EADC start conversion.

Figure 6.44-10 shows the programmable delay time for other trigger source.

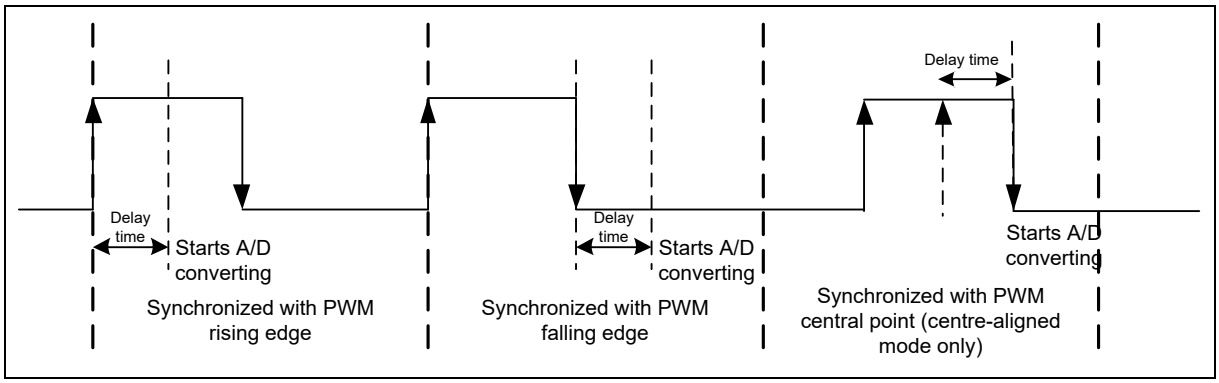


Figure 6.44-9 EPWM-triggered EADC Start Conversion

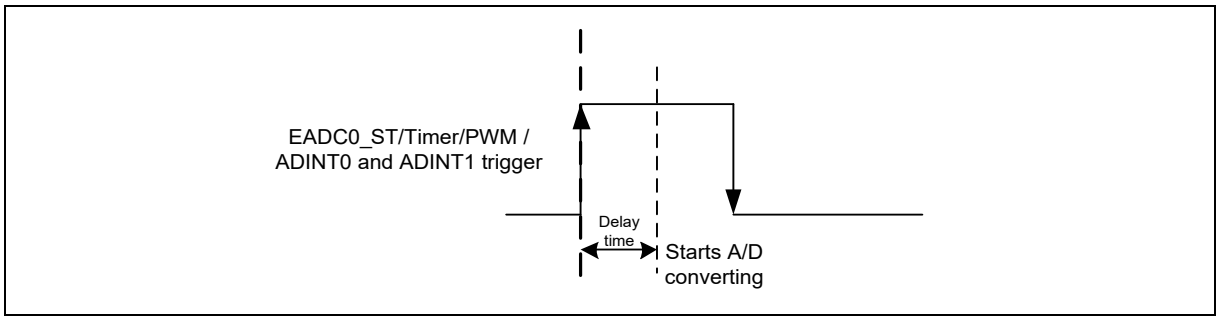


Figure 6.44-10 External triggered EADC Start Conversion

6.44.5.8 EADC Conversion Time and External Trigger

The EADC converter sample the analog input when EADC conversion start delay time (Td) has passed after SWTRGn (EADC_SWTRG[n], n=0~18) is set to 1, then start conversion. Due to EADC clock is generated by PCLK divided by (EADCDIV (CLKDIV0[23:16])+1), the maximum delay time from user write SWTRGn to EADC start sampling analog input time is four EADC clock cycles that operate for synchronization and control pre-processing. The start delay time is shown in Figure 6.44-11.

For example, EADC conversion can be triggered by external pin EADC0_ST request for EADC0. Setting the TRGSEL (EADC_SCTLn[20:16], n=0~15) to 0x01 is to select external trigger input from the EADC0_ST pin. User can set EXTFEN (EADC_SCTLn[22], n=0~15) and EXTREN (EADC_SCTLn[21], n=0~15) to enable pin EADC0_ST trigger condition is falling or rising edge. There is a de-bounce circuit to detect falling or rising edge. If rising edge trigger condition is selected, the low state must be kept at least 2 PCLK cycles and the following high state must be kept at least 3 PCLK cycles. If falling edge trigger condition is selected, the high state must be kept at least 2 PCLK cycles and the following low state must be kept at least 3 PCLK cycles. Pulse that is shorter than this specification will be ignored. The external trigger timing is shown in Figure 6.44-12.

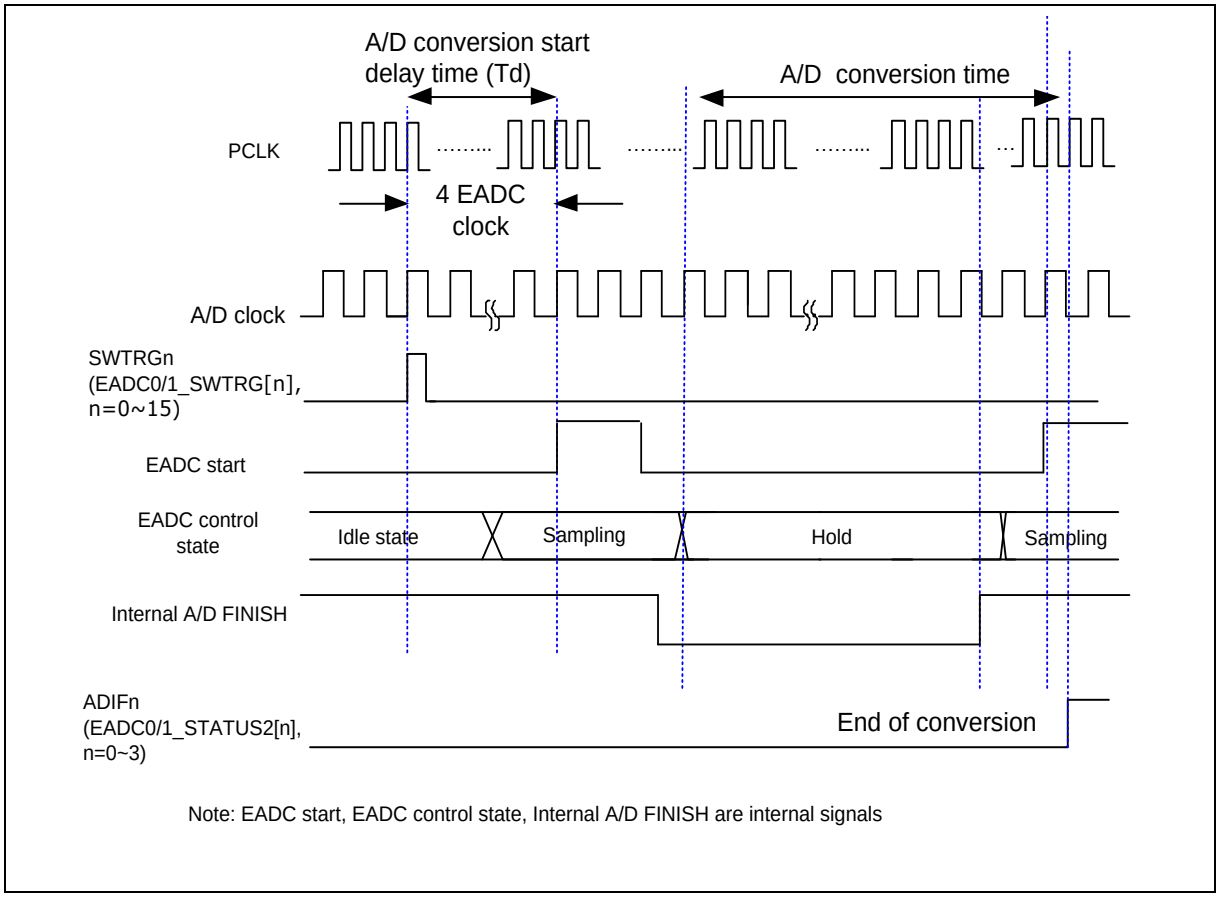


Figure 6.44-11 Conversion Start Delay Timing Diagram

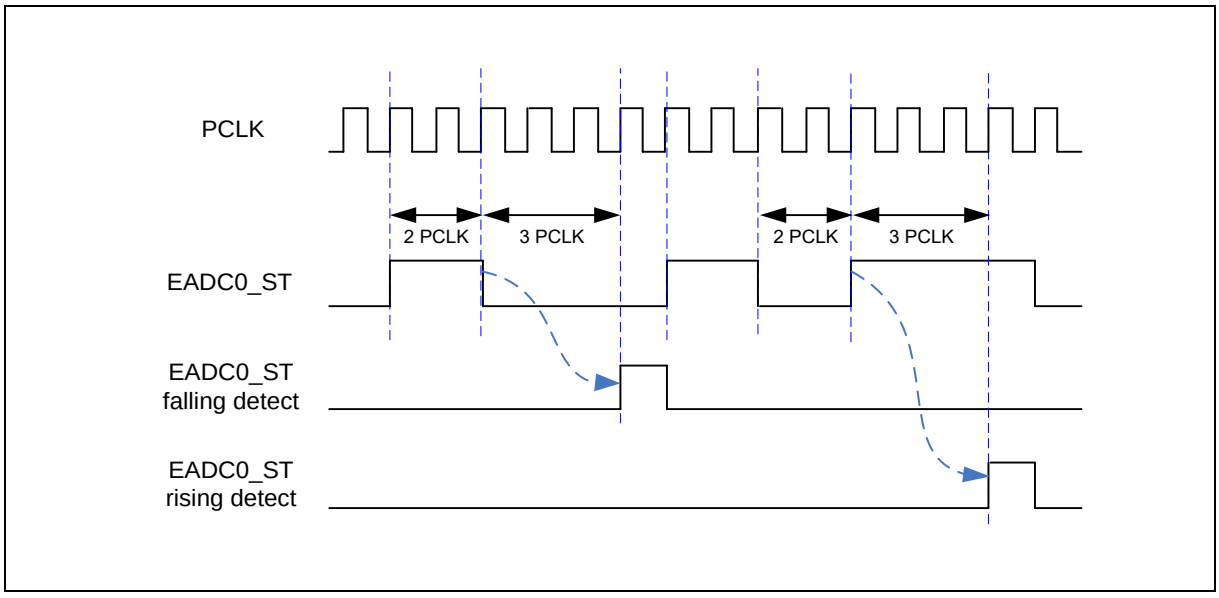


Figure 6.44-12 EADC0_ST De-bounce Timing Diagram

6.44.5.9 EADC Extend Sampling Time

When EADC operates at high EADC clock rate, the sampling time of analog input voltage may not be enough if the analog channel has heavy loading to cause fully charge time is longer. User can set extend

sampling time by writing EXTSMPT (EADC_SCTLn[31:24], n=0~18) for each sample module. The EADC extend sampling time is present between EADC controller judge which channel to be converting and EADC start to conversion. The range of setting extend sampling time is from 0~255 EADC clock. For example, EADC sample module2 is configured with extend sampling time, as shown in Figure 6.44-13.

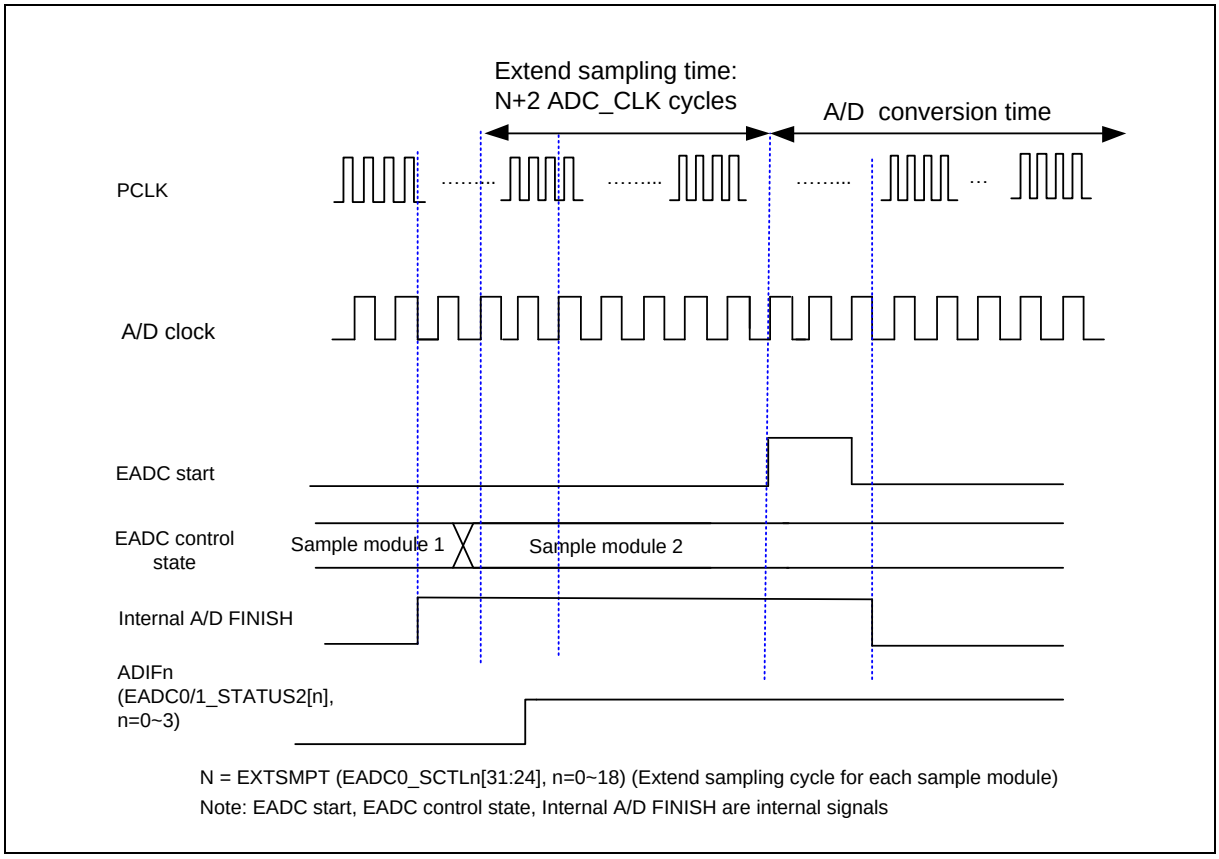


Figure 6.44-13 EADC Extend Sampling Timing Diagram

6.44.5.10 Oversampling Mode

The function provides multiple consecutive conversion results to be accumulated to a final conversion result. Setting ACU (EADC_MnCTL1[7:4], n=0~15) determine whcich sample module and the number of samples to be accumulated. It is noted that when more than 16 samples to be accumulated, the final result after accumulating will exceed the 16-bit RESULT(EADC_DATn [0:15], n=0~15) register size. In this case, the result is right shifted automatically to fit within the appropriate register size. It is noted that accumulation function is only effective without data left alignment.

ACU (EADC_MnCTL1 [7:4]), N=0~15	Number Of Samples To Be Accumulated	Right Shift Division Factor	Final Result Precision
0x0	1	1	12 bits
0x1	2	1	13 bits
0x2	4	1	14 bits
0x3	8	1	15 bits
0x4	16	1	16 bits

0x5	32	2	16 bits
0x6	64	4	16 bits
0x7	128	8	16 bits
0x8	256	16	16 bits

Table 6.44-1 Setting of Accumulation and Conversion Result Precision

6.44.5.11 Averaging Mode

The function increasing the sample accuracy with reduced sampling rate and the feature is suitable in noisy operating environment for getting more stable average conversion results. According to the accumulated final result in 6.44.5.10, setting AVG (EADC_MnCTL1[1]) can enable averaging function as shown in Table 6.44-2. The automatic division factor is decided by the number of accumulated samples and the final result will get 12 bits precision. It is noted that averaging is only effective when accumulating two more samples without data left alignment.

AVG EADC_MnCTL1 [1]	ACU EADC_MnCTL1 [7:4]	Number Of Samples To Be Accumulated	Automatic Division Factor	Final Result Precision
0x1	0x0	1	1	12 bits
0x1	0x1	2	2	12 bits
0x1	0x2	4	4	12 bits
0x1	0x3	8	8	12 bits
0x1	0x4	16	16	12 bits
0x1	0x5	32	32	12 bits
0x1	0x6	64	64	12 bits
0x1	0x7	128	128	12 bits
0x1	0x8	256	256	12 bits

Table 6.44-2 Setting of Averaging and Conversion Result Precision

6.44.5.12 Conversion Result Monitor by Compare Mode

The EADC controller provides four sets of compare registers EADC_CMP0 ~ EADC_CMP3 to monitor a maximum of four specified sample module 0~18 conversion results from EADC conversion module, as shown in Figure 6.44-14. User can select which sample module result to be monitored by set CMPSPL (EADC_CMPn[7:3], n =0~3) and CMPCOND (EADC_CMPn[2], where n =0~3) is used to check conversion result is less than specify value or greater than (equal to) value specified in CMPDAT (EADC_CMPn[27:16], where n =0~3). When the conversion of the sample module specified by CMPSPL is completed, the comparing action will be triggered one time automatically. When the compare result meets the compare condition, the internal compare match counter will increase 1. If the compare result does not meet the condition, the compare match counter will reset to 0. When counter value reach the setting of (CMPMCNT (EADC_CMPn[11:8])+1, where n =0~3) then EADCMPFn (EADC_STATUS2[7:4], where n =0~3) bit will be set to 1, if ADCMPIE (EADC_CMPn[1], n =0~3) is set then an ADINT3 interrupt request is generated. User can use it to monitor the external analog input pin voltage transition. Detailed logics diagram is shown in Figure 6.44-14.

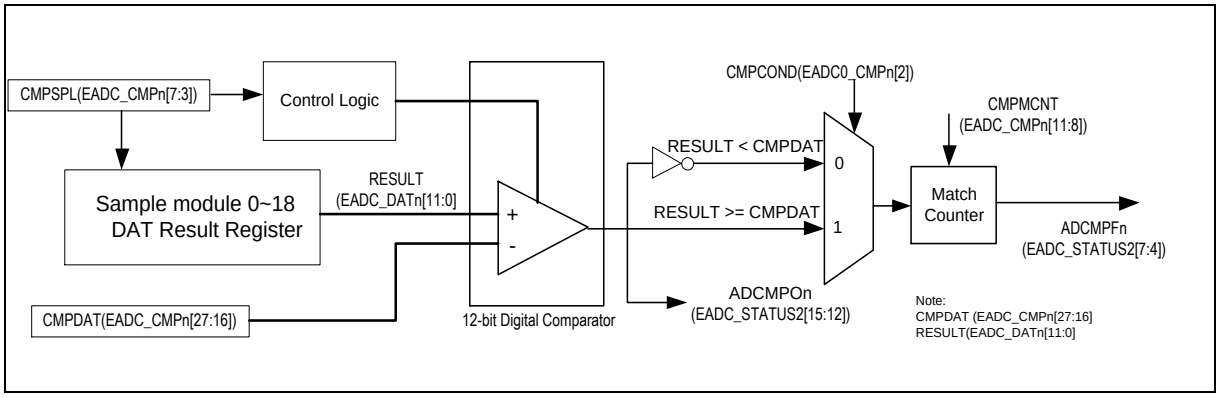


Figure 6.44-14 EADC Conversion Result Monitor Logics Diagram

The EADC controller supports a window compare mode. User can set CMPWEN (EADC_CMP0[15]/ EADC_CMP2[15]) to enable this function. If user enables this function, EADCMPF0 (EADC_STATUS2[4]) will be set when both EADC_CMP0 and EADC_CMP1 compared condition matched. EADCMPF2 (EADC_STATUS2[6]) will be set when both EADC_CMP2 and EADC_CMP3 compared condition matched.

6.44.5.13 Differential Mode

The EADC controller supports analog fully-differential mode. A fully-differential SAR converts the differential voltage across its inputs; however, in this case both inputs change dynamically and are complementary or inverted from each other. In this case, the VINP and VINM pins must be driven 180° out of phase with respect to each other, centered on a fixed common mode voltage, for example V_{REF}/2. Fully-differential EADCs have an input voltage common-mode (VCM) range specification. VCM is defined as the average voltage between the inputs: VCM = (VINP + VINM) / 2. Most fully-differential input SAR EADCs prohibit the input common-mode voltage from varying more than approximately 10 percent beyond the mid-scale input (V_{REF} /2).

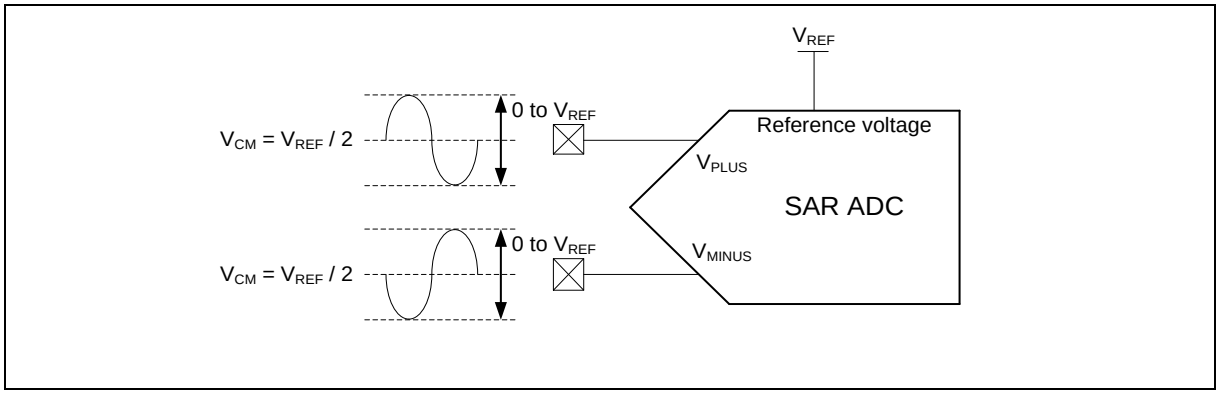


Figure 6.44-15 Input of Fully-Differential EADC

If user enables DIFFEN (EADC_CTL[8]), the differential mode will enable. The pair of analog input channel is as Table 6.44-3

Differential Analog Input Paired Channel	ADC Analog Input	
	V_{plus}	V_{minus}
0	EADC_CH0	EADC_CH1
1	EADC_CH2	EADC_CH3

2	EADC_CH4	EADC_CH5
3	EADC_CH6	EADC_CH7
4	EADC_CH8	EADC_CH9
5	EADC_CH10	EADC_CH11
6	EADC_CH12	EADC_CH13
7	EADC_CH14	EADC_CH15

Table 6.44-3 EADC Differential Model Channel Selection

In differential analog input mode, only the even number of the two corresponding channels needs to be enabled in CHSEL (EADC_SCTLn[4:0]). The conversion result will be placed to the corresponding data register of the enabled channel.

6.44.5.14 Double Buffer Mode

The EADC controller supports a double buffer mode in sample module 0~3. If user enable DBMEN (EADC_SCTLn[23], n=0~3), the double buffer mode will enable. In double buffer mode, after first time EADC convert finish, the VALID (EADC_DATn[17], n=0~3) will set to high, but VALID (EADC_DDATn[17], n=0~3) will keep low. And the second time EADC converts finish, VALID (EADC_DDATn[17], n=0~3) will set to high either. Then, user can get the EADC results from EADC_DATn and EADC_DDATn register.

6.44.5.15 PDMA Request

The EADC controller supports PDMA. PDMA could service each channel when corresponding channel PDMA transfer enable bit, PDMATEN (EADC_PDMACTL[18:0]), is activated. For example, user can enable PDMATEN for specific channels and configure PDMA channel's source address as CURDAT(EADC_CURDAT[18:0]). After enabling PDMATEN and PDMA channel, if any VALID (EADC_DATn[17],n=0~18) is high, EADC controller will send request to PDMA and PDMA will read EADC_CURDAT to get result. The EADC_CURDAT register is a shadow register of highest priority EADC_DAT register. The lower number sample module is higher priority. After PDMA read EADC_CURDAT register, the VAILD of the EADC_DAT register will be automatically cleared.

6.44.5.16 Interrupt Sources

The EADC converter generates ADIFn (EADC_STATUS2[3:0], n=0~3) at the start of conversion or the end of conversion decide by INTPOS (EADC_SCTLn[5], n=0~18). If EADCIENn (EADC_CTL[5:2], n=0~3) is set then conversion end interrupt request ADINTn (n=0~3) is generated. The controller of interrupts is shown as Figure 6.44-16.

It is noted that calibration interrupt CALINT is also shown in Figure 6.44-16.

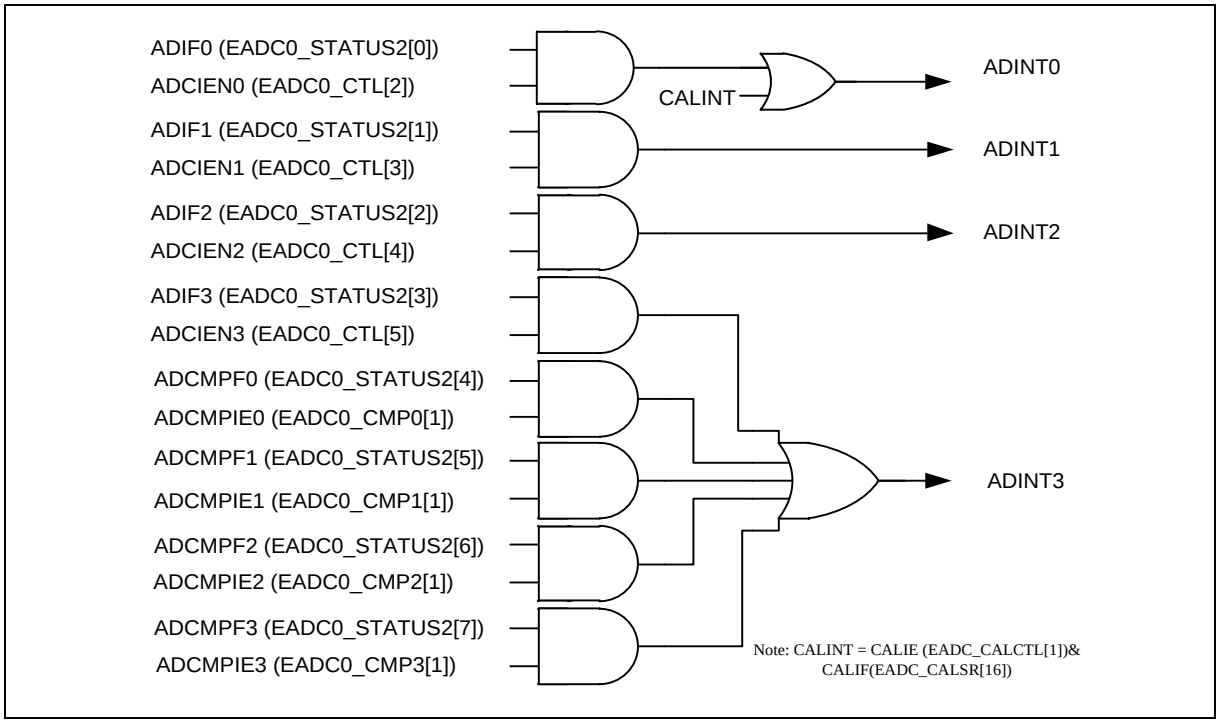


Figure 6.44-16 EADC Controller Interrupts

6.44.5.17 Calibration Mode

To decrease the effect of electrical random noise, the calibration mode performs an offset and mismatch measurement cycles. Afterwards, in normal operation mode, the calibration engine applies to the capacitor array, so that the offset and mismatch are removed. If chip power off, calibration function should be executed again.

In Calibration Mode:

1. Set CAL(EADC_CALCTL[0]) bit to 1.
2. After 3 ADC_CLK when CAL is set to 1, calibration is started.
3. After calibration is finished, the flag CALIF(EADC_CALSR[16]) will be set to 1. If CALIE (EADC_CALCTL[1]) is set, the calibration interrupt will be generated.
4. After calibration is finished, the mode will become Normal Mode and CAL is cleared by hardware automatically.

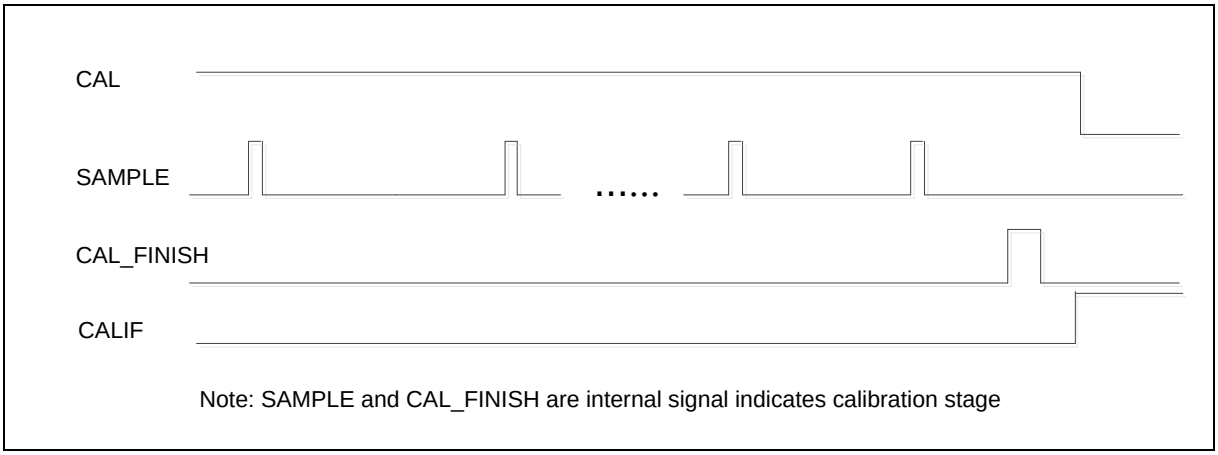


Figure 6.44-17 Calibration Mode with 16 Times Average

6.44.5.18 Reference Voltage Source Select

ADC bases on reference voltage to convert an analog signal to digital value. The reference voltage source can be selected from V_{REF} by REFSEL (EADC_VREF[1]). And the V_{REF} can select as external pin, internal reference voltage 1.6V, 2.0V, 2.5V and 3.0V by VREFCTL (SYS_VREFCTL[4:0]). Note that for the pin diagram without V_{REF} pin, the ADC reference voltage comes from AV_{DD}

6.44.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
EADC Base Address: **EADC0_BA = 0x4004_3000** **EADC1_BA = 0x4004_B000** **EADC2_BA = 0x4009_7000**				
EADC_DAT0 **x=0,1,2**	EADCx_BA+0x00	R	EADC Data Register 0 for Sample Module 0	0x0000_0000
EADC_DAT1 **x=0,1,2**	EADCx_BA+0x04	R	EADC Data Register 1 for Sample Module 1	0x0000_0000
EADC_DAT2 **x=0,1,2**	EADCx_BA+0x08	R	EADC Data Register 2 for Sample Module 2	0x0000_0000
EADC_DAT3 **x=0,1,2**	EADCx_BA+0x0C	R	EADC Data Register 3 for Sample Module 3	0x0000_0000
EADC_DAT4 **x=0,1,2**	EADCx_BA+0x10	R	EADC Data Register 4 for Sample Module 4	0x0000_0000
EADC_DAT5 **x=0,1,2**	EADCx_BA+0x14	R	EADC Data Register 5 for Sample Module 5	0x0000_0000
EADC_DAT6 **x=0,1,2**	EADCx_BA+0x18	R	EADC Data Register 6 for Sample Module 6	0x0000_0000
EADC_DAT7 **x=0,1,2**	EADCx_BA+0x1C	R	EADC Data Register 7 for Sample Module 7	0x0000_0000
EADC_DAT8 **x=0,1,2**	EADCx_BA+0x20	R	EADC Data Register 8 for Sample Module 8	0x0000_0000
EADC_DAT9 **x=0,1,2**	EADCx_BA+0x24	R	EADC Data Register 9 for Sample Module 9	0x0000_0000
EADC_DAT10 **x=0,1,2**	EADCx_BA+0x28	R	EADC Data Register 10 for Sample Module 10	0x0000_0000
EADC_DAT11 **x=0,1,2**	EADCx_BA+0x2C	R	EADC Data Register 11 for Sample Module 11	0x0000_0000
EADC_DAT12 **x=0,1,2**	EADCx_BA+0x30	R	EADC Data Register 12 for Sample Module 12	0x0000_0000
EADC_DAT13 **x=0,1,2**	EADCx_BA+0x34	R	EADC Data Register 13 for Sample Module 13	0x0000_0000
EADC_DAT14 **x=0,1,2**	EADCx_BA+0x38	R	EADC Data Register 14 for Sample Module 14	0x0000_0000
EADC_DAT15 **x=0,1,2**	EADCx_BA+0x3C	R	EADC Data Register 15 for Sample Module 15	0x0000_0000
EADC_DAT16 **x=0,1,2**	EADCx_BA+0x40	R	EADC Data Register 16 for Sample Module 16	0x0000_0000

Register	Offset	R/W	Description	Reset Value
EADC_DAT17 x=0,1,2	EADCx_BA+0x44	R	EADC Data Register 17 for Sample Module 17	0x0000_0000
EADC_DAT18 x=0,1,2	EADCx_BA+0x48	R	EADC Data Register 18 for Sample Module 18	0x0000_0000
EADC_CURDAT x=0,1,2	EADCx_BA+0x4C	R	EADC PDMA Current Transfer Data Register	0x0000_0000
EADC_CTL x=0,1,2	EADCx_BA+0x50	R/W	EADC Control Register	0x0000_0000
EADC_SWTRG x=0,1,2	EADCx_BA+0x54	W	EADC Sample Module Software Start Register	0x0000_0000
EADC_PENDSTS x=0,1,2	EADCx_BA+0x58	R/W	EADC Start of Conversion Pending Flag Register	0x0000_0000
EADC_OVSTS x=0,1,2	EADCx_BA+0x5C	R/W	EADC Sample Module Start of Conversion Overrun Flag Register	0x0000_0000
EADC_CTL1 x=0,1,2	EADCx_BA+0x60	R/W	EADC Control1 Register	0x0000_0000
EADC_SCTL0 x=0,1,2	EADCx_BA+0x80	R/W	EADC Sample Module 0 Control Register	0x0000_0011
EADC_SCTL1 x=0,1,2	EADCx_BA+0x84	R/W	EADC Sample Module 1 Control Register	0x0000_0011
EADC_SCTL2 x=0,1,2	EADCx_BA+0x88	R/W	EADC Sample Module 2 Control Register	0x0000_0011
EADC_SCTL3 x=0,1,2	EADCx_BA+0x8C	R/W	EADC Sample Module 3 Control Register	0x0000_0011
EADC_SCTL4 x=0,1,2	EADCx_BA+0x90	R/W	EADC Sample Module 4 Control Register	0x0000_0011
EADC_SCTL5 x=0,1,2	EADCx_BA+0x94	R/W	EADC Sample Module 5 Control Register	0x0000_0011
EADC_SCTL6 x=0,1,2	EADCx_BA+0x98	R/W	EADC Sample Module 6 Control Register	0x0000_0011
EADC_SCTL7 x=0,1,2	EADCx_BA+0x9C	R/W	EADC Sample Module 7 Control Register	0x0000_0011
EADC_SCTL8 x=0,1,2	EADCx_BA+0xA0	R/W	EADC Sample Module 8 Control Register	0x0000_0011
EADC_SCTL9 x=0,1,2	EADCx_BA+0xA4	R/W	EADC Sample Module 9 Control Register	0x0000_0011
EADC_SCTL10 x=0,1,2	EADCx_BA+0xA8	R/W	EADC Sample Module 10 Control Register	0x0000_0011
EADC_SCTL11 x=0,1,2	EADCx_BA+0xAC	R/W	EADC Sample Module 11 Control Register	0x0000_0011

Register	Offset	R/W	Description	Reset Value
EADC_SCTL12 x=0,1,2	EADCx_BA+0xB0	R/W	EADC Sample Module 12 Control Register	0x0000_0011
EADC_SCTL13 x=0,1,2	EADCx_BA+0xB4	R/W	EADC Sample Module 13 Control Register	0x0000_0011
EADC_SCTL14 x=0,1,2	EADCx_BA+0xB8	R/W	EADC Sample Module 14 Control Register	0x0000_0011
EADC_SCTL15 x=0,1,2	EADCx_BA+0xBC	R/W	EADC Sample Module 15 Control Register	0x0000_0011
EADC_SCTL16 x=0,1,2	EADCx_BA+0xC0	R/W	EADC Sample Module 16 Control Register	0x0000_0000
EADC_SCTL17 x=0,1,2	EADCx_BA+0xC4	R/W	EADC Sample Module 17 Control Register	0x0000_0000
EADC_SCTL18 x=0,1,2	EADCx_BA+0xC8	R/W	EADC Sample Module 18 Control Register	0x0000_0000
EADC_INTSRC0 x=0,1,2	EADCx_BA+0xD0	R/W	EADC Interrupt 0 Source Enable Control Register.	0x0000_0000
EADC_INTSRC1 x=0,1,2	EADCx_BA+0xD4	R/W	EADC Interrupt 1 Source Enable Control Register.	0x0000_0000
EADC_INTSRC2 x=0,1,2	EADCx_BA+0xD8	R/W	EADC Interrupt 2 Source Enable Control Register.	0x0000_0000
EADC_INTSRC3 x=0,1,2	EADCx_BA+0xDC	R/W	EADC Interrupt 3 Source Enable Control Register.	0x0000_0000
EADC_CMP0 x=0,1,2	EADCx_BA+0xE0	R/W	EADC Result Compare Register 0	0x0000_0000
EADC_CMP1 x=0,1,2	EADCx_BA+0xE4	R/W	EADC Result Compare Register 1	0x0000_0000
EADC_CMP2 x=0,1,2	EADCx_BA+0xE8	R/W	EADC Result Compare Register 2	0x0000_0000
EADC_CMP3 x=0,1,2	EADCx_BA+0xEC	R/W	EADC Result Compare Register 3	0x0000_0000
EADC_STATUS0 x=0,1,2	EADCx_BA+0xF0	R	EADC Status Register 0	0x0000_0000
EADC_STATUS1 x=0,1,2	EADCx_BA+0xF4	R	EADC Status Register 1	0x0000_0000
EADC_STATUS2 x=0,1,2	EADCx_BA+0xF8	R/W	EADC Status Register 2	0x0011_0000
EADC_STATUS3 x=0,1,2	EADCx_BA+0xFC	R	EADC Status Register 3	0x0000_001F
EADC_DDAT0 x=0,1,2	EADCx_BA+0x100	R	EADC Double Data Register 0 for Sample Module 0	0x0000_0000

Register	Offset	R/W	Description	Reset Value
EADC_DDAT1 x=0,1,2	EADCx_BA+0x104	R	EADC Double Data Register 1 for Sample Module 1	0x0000_0000
EADC_DDAT2 x=0,1,2	EADCx_BA+0x108	R	EADC Double Data Register 2 for Sample Module 2	0x0000_0000
EADC_DDAT3 x=0,1,2	EADCx_BA+0x10C	R	EADC Double Data Register 3 for Sample Module 3	0x0000_0000
EADC_CALCTL x=0,1,2	EADCx_BA+0x114	R/W	EADC Calibration Control Register	0x0000_0020
EADC_CALSR x=0,1,2	EADCx_BA+0x118	R/W	EADC Calibration Status Register	0x0000_0000
EADC_PDMACTL x=0,1,2	EADCx_BA+0x130	R/W	EADC PDMA Control Register	0x0000_0000
EADC_M0CTL1 x=0,1,2	EADCx_BA+0x140	R/W	EADC Sample Module0 Control Register 1	0x0000_0000
EADC_M1CTL1 x=0,1,2	EADCx_BA+0x144	R/W	EADC Sample Module1 Control Register 1	0x0000_0000
EADC_M2CTL1 x=0,1,2	EADCx_BA+0x148	R/W	EADC Sample Module2 Control Register 1	0x0000_0000
EADC_M3CTL1 x=0,1,2	EADCx_BA+0x14C	R/W	EADC Sample Module3 Control Register 1	0x0000_0000
EADC_M4CTL1 x=0,1,2	EADCx_BA+0x150	R/W	EADC Sample Module4 Control Register 1	0x0000_0000
EADC_M5CTL1 x=0,1,2	EADCx_BA+0x154	R/W	EADC Sample Module5 Control Register 1	0x0000_0000
EADC_M6CTL1 x=0,1,2	EADCx_BA+0x158	R/W	EADC Sample Module6 Control Register 1	0x0000_0000
EADC_M7CTL1 x=0,1,2	EADCx_BA+0x15C	R/W	EADC Sample Module7 Control Register 1	0x0000_0000
EADC_M8CTL1 x=0,1,2	EADCx_BA+0x160	R/W	EADC Sample Module8 Control Register 1	0x0000_0000
EADC_M9CTL1 x=0,1,2	EADCx_BA+0x164	R/W	EADC Sample Module9 Control Register 1	0x0000_0000
EADC_M10CTL1 x=0,1,2	EADCx_BA+0x168	R/W	EADC Sample Module10 Control Register 1	0x0000_0000
EADC_M11CTL1 x=0,1,2	EADCx_BA+0x16C	R/W	EADC Sample Module11 Control Register 1	0x0000_0000
EADC_M12CTL1 x=0,1,2	EADCx_BA+0x170	R/W	EADC Sample Module12 Control Register 1	0x0000_0000
EADC_M13CTL1 x=0,1,2	EADCx_BA+0x174	R/W	EADC Sample Module13 Control Register 1	0x0000_0000

Register	Offset	R/W	Description	Reset Value
EADC_M14CTL1 x=0,1,2	EADCx_BA+0x178	R/W	EADC Sample Module14 Control Register 1	0x0000_0000
EADC_M15CTL1 x=0,1,2	EADCx_BA+0x17C	R/W	EADC Sample Module15 Control Register 1	0x0000_0000
EADC_VREF x=0,1,2	EADCx_BA+0xFF8	R/W	EADC Reference Voltage Control Register	0x0000_0000

6.44.7 Register Description

EADC Data Registers (EADC_DAT0~ EADC_DAT18)

Register	Offset	R/W	Description	Reset Value
EADC_DAT0	EADCx_BA+0x00	R	EADC Data Register 0 for Sample Module 0	0x0000_0000
EADC_DAT1	EADCx_BA+0x04	R	EADC Data Register 1 for Sample Module 1	0x0000_0000
EADC_DAT2	EADCx_BA+0x08	R	EADC Data Register 2 for Sample Module 2	0x0000_0000
EADC_DAT3	EADCx_BA+0x0C	R	EADC Data Register 3 for Sample Module 3	0x0000_0000
EADC_DAT4	EADCx_BA+0x10	R	EADC Data Register 4 for Sample Module 4	0x0000_0000
EADC_DAT5	EADCx_BA+0x14	R	EADC Data Register 5 for Sample Module 5	0x0000_0000
EADC_DAT6	EADCx_BA+0x18	R	EADC Data Register 6 for Sample Module 6	0x0000_0000
EADC_DAT7	EADCx_BA+0x1C	R	EADC Data Register 7 for Sample Module 7	0x0000_0000
EADC_DAT8	EADCx_BA+0x20	R	EADC Data Register 8 for Sample Module 8	0x0000_0000
EADC_DAT9	EADCx_BA+0x24	R	EADC Data Register 9 for Sample Module 9	0x0000_0000
EADC_DAT10	EADCx_BA+0x28	R	EADC Data Register 10 for Sample Module 10	0x0000_0000
EADC_DAT11	EADCx_BA+0x2C	R	EADC Data Register 11 for Sample Module 11	0x0000_0000
EADC_DAT12	EADCx_BA+0x30	R	EADC Data Register 12 for Sample Module 12	0x0000_0000
EADC_DAT13	EADCx_BA+0x34	R	EADC Data Register 13 for Sample Module 13	0x0000_0000
EADC_DAT14	EADCx_BA+0x38	R	EADC Data Register 14 for Sample Module 14	0x0000_0000
EADC_DAT15	EADCx_BA+0x3C	R	EADC Data Register 15 for Sample Module 15	0x0000_0000
EADC_DAT16	EADCx_BA+0x40	R	EADC Data Register 16 for Sample Module 16	0x0000_0000
EADC_DAT17	EADCx_BA+0x44	R	EADC Data Register 17 for Sample Module 17	0x0000_0000
EADC_DAT18	EADCx_BA+0x48	R	EADC Data Register 18 for Sample Module 18	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**VALID**	**OV**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESULT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESULT							

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**VALID**	**Valid Flag** This bit is set to 1 when corresponding sample module channel analog input conversion is completed and cleared by hardware after EADC_DAT register is read. 0 = Data in RESULT[11:0] bits is not valid. 1 = Data in RESULT[11:0] bits is valid.
[16]	**OV**	**Overrun Flag** If converted data in RESULT[11:0] has not been read before new conversion result is loaded to this register, OV is set to 1. 0 = Data in RESULT[11:0] is recent conversion result. 1 = Data in RESULT[11:0] is overwrite. **Note:** It is cleared by hardware after EADC_DAT register is read.
[15:0]	**RESULT**	**EADC Conversion Result** This field contains 12 bits conversion result. The 12-bit EADC conversion result with unsigned format will be filled in RESULT[11:0] and zero will be filled in RESULT[15:12]. **Note:** When operating in oversampling mode, RESULT[15:0] can represent oversampling results.

EADC PDMA Current Transfer Data Register (EADC_CURDAT)

Register	Offset	R/W	Description	Reset Value
EADC_CURDAT	EADCx_BA+0x4C	R	EADC PDMA Current Transfer Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**CURDAT**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CURDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURDAT							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**CURDAT**	**EADC PDMA Current Transfer Data (Read Only)** This register is a shadow register of EADC_DATn (n=0~18) for PDMA support.

EADC Control Register (EADC_CTL)

Register	Offset	R/W	Description	Reset Value
EADC_CTL	EADCx_BA+0x50	R/W	EADC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
INTDELAY3				INTDELAY2			
23	22	21	20	19	18	17	16
INTDELAY1				INTDELAY0			
15	14	13	12	11	10	9	8
Reserved						DMOF	DIFFEN
7	6	5	4	3	2	1	0
Reserved		ADCIEN3	ADCIEN2	ADCIEN1	ADCIEN0	ADCRST	ADCEN

Bits	Description	
[31:28]	**INTDELAY3**	**ADC Start Of Conversion ADINT3 Delay Cycle Selection** Start of conversion interrupt ADINT3 will delay INTDELAY3 PCLK cycles to generate interrupt. The function supports delay 1 PCLK to 15 PCLK cycles. User can select one of the options according to the relationship of PCLK and ADC _CLK selected. 4'h0 = No delay cycle. 4'h1 = Start of conversion interrupt ADINT3 delay 1 PCLK cycle. 4'h2 = Start of conversion interrupt ADINT3 delay 2 PCLK cycles. 4'h3 = Start of conversion interrupt ADINT3 delay 3 PCLK cycles. 4'h4 = Start of conversion interrupt ADINT3 delay 4 PCLK cycles. 4'h5 = Start of conversion interrupt ADINT3 delay 5 PCLK cycles. 4'h6 = Start of conversion interrupt ADINT3 delay 6 PCLK cycles. 4'h7 = Start of conversion interrupt ADINT3 delay 7 PCLK cycles. 4'h8 = Start of conversion interrupt ADINT3 delay 8 PCLK cycles. 4'h9 = Start of conversion interrupt ADINT3 delay 9 PCLK cycles. 4'ha = Start of conversion interrupt ADINT3 delay 10 PCLK cycles. 4'hb = Start of conversion interrupt ADINT3 delay 11 PCLK cycles. 4'hc = Start of conversion interrupt ADINT3 delay 12 PCLK cycles. 4'hd = Start of conversion interrupt ADINT3 delay 13 PCLK cycles. 4'he = Start of conversion interrupt ADINT3 delay 14 PCLK cycles. 4'hf = Start of conversion interrupt ADINT3 delay 15 PCLK cycles. **Note 1:** This function is workable only when any one of INTPOS (EADC_SCTLx[5]), x=0~15 is set. **Note 2:** It is noted that the delayed interrupt ADINT3 must occur before the next ADINT3 generated when using the same sample module to control EADC conversion.

Bits	Description	
[27:24]	**INTDELAY2**	**ADC Start Of Conversion ADINT2 Delay Cycle Selection** Start of conversion interrupt ADINT2 will delay INTDELAY2 PCLK cycles to generate interrupt. The function supports delay 1 PCLK to 15 PCLK cycles. User can select one of the options according to the relationship of PCLK and ADC _CLK selected. 4'h0 = No delay cycle. 4'h1 = Start of conversion interrupt ADINT2 delay 1 PCLK cycle. 4'h2 = Start of conversion interrupt ADINT2 delay 2 PCLK cycles. 4'h3 = Start of conversion interrupt ADINT2 delay 3 PCLK cycles. 4'h4 = Start of conversion interrupt ADINT2 delay 4 PCLK cycles. 4'h5 = Start of conversion interrupt ADINT2 delay 5 PCLK cycles. 4'h6 = Start of conversion interrupt ADINT2 delay 6 PCLK cycles. 4'h7 = Start of conversion interrupt ADINT2 delay 7 PCLK cycles. 4'h8 = Start of conversion interrupt ADINT2 delay 8 PCLK cycles. 4'h9 = Start of conversion interrupt ADINT2 delay 9 PCLK cycles. 4'ha = Start of conversion interrupt ADINT2 delay 10 PCLK cycles. 4'hb = Start of conversion interrupt ADINT2 delay 11 PCLK cycles. 4'hc = Start of conversion interrupt ADINT2 delay 12 PCLK cycles. 4'hd = Start of conversion interrupt ADINT2 delay 13 PCLK cycles. 4'he = Start of conversion interrupt ADINT2 delay 14 PCLK cycles. 4'hf = Start of conversion interrupt ADINT2 delay 15 PCLK cycles. **Note 1:** This function is workable only when any one of INTPOS (EADC_SCTLx[5]), x=0~15 is set. **Note 2:** It is noted that the delayed interrupt ADINT2 must occur before the next ADINT2 generated when using the same sample module to control EADC conversion.
[23:20]	**INTDELAY1**	**ADC Start Of Conversion ADINT1 Delay Cycle Selection** Start of conversion interrupt ADINT1 will delay INTDELAY1 PCLK cycles to generate interrupt. The function supports delay 1 PCLK to 15 PCLK cycles. User can select one of the options according to the relationship of PCLK and ADC _CLK selected. 4'h0 = No delay cycle. 4'h1 = Start of conversion interrupt ADINT1 delay 1 PCLK cycle. 4'h2 = Start of conversion interrupt ADINT1 delay 2 PCLK cycles. 4'h3 = Start of conversion interrupt ADINT1 delay 3 PCLK cycles. 4'h4 = Start of conversion interrupt ADINT1 delay 4 PCLK cycles. 4'h5 = Start of conversion interrupt ADINT1 delay 5 PCLK cycles. 4'h6 = Start of conversion interrupt ADINT1 delay 6 PCLK cycles. 4'h7 = Start of conversion interrupt ADINT1 delay 7 PCLK cycles. 4'h8 = Start of conversion interrupt ADINT1 delay 8 PCLK cycles. 4'h9 = Start of conversion interrupt ADINT1 delay 9 PCLK cycles. 4'ha = Start of conversion interrupt ADINT1 delay 10 PCLK cycles. 4'hb = Start of conversion interrupt ADINT1 delay 11 PCLK cycles. 4'hc = Start of conversion interrupt ADINT1 delay 12 PCLK cycles. 4'hd = Start of conversion interrupt ADINT1 delay 13 PCLK cycles. 4'he = Start of conversion interrupt ADINT1 delay 14 PCLK cycles. 4'hf = Start of conversion interrupt ADINT1 delay 15 PCLK cycles. **Note 1:** This function is workable only when any one of INTPOS (EADC_SCTLx[5]), x=0~15 is set. **Note 2:** It is noted that the delayed interrupt ADINT1 must occur before the next ADINT1 generated when using the same sample module to control EADC conversion.

Bits	Description	
[19:16]	**INTDELAY0**	**ADC Start Of Conversion ADINT0 Delay Cycle Selection** Start of conversion interrupt ADINT0 will delay INTDELAY0 PCLK cycles to generate interrupt. The function supports delay 1 PCLK to 15 PCLK cycles. User can select one of the options according to the relationship of PCLK and ADC _CLK selected. 4'h0 = No delay cycle. 4'h1 = Start of conversion interrupt ADINT0 delay 1 PCLK cycle. 4'h2 = Start of conversion interrupt ADINT0 delay 2 PCLK cycles. 4'h3 = Start of conversion interrupt ADINT0 delay 3 PCLK cycles. 4'h4 = Start of conversion interrupt ADINT0 delay 4 PCLK cycles. 4'h5 = Start of conversion interrupt ADINT0 delay 5 PCLK cycles. 4'h6 = Start of conversion interrupt ADINT0 delay 6 PCLK cycles. 4'h7 = Start of conversion interrupt ADINT0 delay 7 PCLK cycles. 4'h8 = Start of conversion interrupt ADINT0 delay 8 PCLK cycles. 4'h9 = Start of conversion interrupt ADINT0 delay 9 PCLK cycles. 4'ha = Start of conversion interrupt ADINT0 delay 10 PCLK cycles. 4'hb = Start of conversion interrupt ADINT0 delay 11 PCLK cycles. 4'hc = Start of conversion interrupt ADINT0 delay 12 PCLK cycles. 4'hd = Start of conversion interrupt ADINT0 delay 13 PCLK cycles. 4'he = Start of conversion interrupt ADINT0 delay 14 PCLK cycles. 4'hf = Start of conversion interrupt ADINT0 delay 15 PCLK cycles. **Note 1:** This function is workable only when any one of INTPOS (EADC_SCTLx[5]), x=0~15 is set. **Note 2:** It is noted that the delayed interrupt ADINT0 must occur before the next ADINT0 generated when using the same sample module to control EADC conversion.
[15:10]	**Reserved**	Reserved.
[9]	**DMOF**	**ADC Differential Input Mode Output Format** 0 = ADC conversion result will be filled in RESULT (EADC_DATn[15:0], where n= 0 ~18) with unsigned format. 1 = ADC conversion result will be filled in RESULT (EADC_DATn[15:0], where n= 0 ~18) with 2'complement format.
[8]	**DIFFEN**	**Differential Analog Input Mode Enable Bit** 0 = Single-end analog input mode. 1 = Differential analog input mode. **Note:** In the differential mode, the input channel pair must be configured to EADC_CHx, EADC_CHx+1, x=0,2,4,6,8,10,12,14.
[7:6]	**Reserved**	Reserved.
[5]	**ADCIEN3**	**Specific Sample Module EADC ADINT3 Interrupt Enable Bit** The EADC converter generates a conversion end ADIF3 (EADC_STATUS2[3]) upon the end of specific sample module EADC conversion. If EADCIEN3 bit is set then conversion end interrupt request ADINT3 is generated. 0 = Specific sample module EADC ADINT3 interrupt function Disabled. 1 = Specific sample module EADC ADINT3 interrupt function Enabled.
[4]	**ADCIEN2**	**Specific Sample Module EADC ADINT2 Interrupt Enable Bit** The EADC converter generates a conversion end ADIF2 (EADC_STATUS2[2]) upon the end of specific sample module EADC conversion. If EADCIEN2 bit is set then conversion end interrupt request ADINT2 is generated. 0 = Specific sample module EADC ADINT2 interrupt function Disabled. 1 = Specific sample module EADC ADINT2 interrupt function Enabled.

Bits	Description	
[3]	**ADCIEN1**	**Specific Sample Module EADC ADINT1 Interrupt Enable Bit** The EADC converter generates a conversion end ADIF1 (EADC_STATUS2[1]) upon the end of specific sample module EADC conversion. If EADCIEN1 bit is set then conversion end interrupt request ADINT1 is generated. 0 = Specific sample module EADC ADINT1 interrupt function Disabled. 1 = Specific sample module EADC ADINT1 interrupt function Enabled.
[2]	**ADCIEN0**	**Specific Sample Module EADC ADINT0 Interrupt Enable Bit** The EADC converter generates a conversion end ADIF0 (EADC_STATUS2[0]) upon the end of specific sample module EADC conversion. If ADCIEN0 bit is set then conversion end interrupt request ADINT0 is generated. 0 = Specific sample module EADC ADINT0 interrupt function Disabled. 1 = Specific sample module EADC ADINT0 interrupt function Enabled.
[1]	**ADCRST**	**EADC Converter Control Circuits Reset** 0 = No effect. 1 = Cause EADC control circuits reset to initial state, but not change the EADC registers value. **Note: EADC**RST bit remains 1 during EADC reset, when EADC reset end, the EADCRST bit is automatically cleared to 0.
[0]	**ADCEN**	**EADC Converter Enable Bit** 0 = EADC Disabled. 1 = EADC Enabled. **Note:** Before starting EADC conversion function, this bit should be set to 1. Clear it to 0 to disable EADC converter analog circuit power consumption.

EADC Sample Module Software Start Register (EADC_SWTRG)

Register	Offset	R/W	Description	Reset Value
EADC_SWTRG	EADCx_BA+0x54	W	EADC Sample Module Software Start Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					**Reserved**		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**SWTRG**		
14	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SWTRG							
6	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWTRG							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**SWTRG**	**EADC Sample Module 0~18 Software Force to Start EADC Conversion** 0 = No effect. 1 = Cause an EADC conversion when the priority is given to sample module. **Note:** After writing this register to start EADC conversion, the EADC_PENDSTS register will show which sample module will conversion. If user want to disable the conversion of the sample module, user can write EADC_PENDSTS register to clear it.

EADC Sample Module Start of Conversion Pending Flag Register (EADC_PENDSTS)

Register	Offset	R/W	Description	Reset Value
EADC_PENDSTS	EADCx_BA+0x58	R/W	EADC Start of Conversion Pending Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					STPF		
15	14	13	12	11	10	9	8
STPF							
7	6	5	4	3	2	1	0
STPF							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**STPF**	**EADC Sample Module 0~18 Start of Conversion Pending Flag** Read Operation: 0 = There is no pending conversion for sample module. 1 = Sample module EADC start of conversion is pending. Write Operation: 1 = Clear pending flag & cancel the conversion for sample module. **Note:** This bit remains 1 during pending state when the respective EADC conversion is ended. The STPFn (n=0~18) bit is automatically cleared to 0.

EADC Sample Module Overrun Flag Register (EADC_OVSTS)

Register	Offset	R/W	Description	Reset Value
EADC_OVSTS	EADCx_BA+0x5C	R/W	EADC Sample Module Start of Conversion Overrun Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					SPOVF		
15	14	13	12	11	10	9	8
SPOVF							
7	6	5	4	3	2	1	0
SPOVF							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**SPOVF**	**EADC SAMPLE0~18 Overrun Flag** 0 = No sample module event overrun. 1 = Indicates a new sample module event is generated while an old one event is pending. **Note:** This bit is cleared by writing 1 to it.

EADC Control1 Register (EADC_CTL1)

Register	Offset	R/W	Description	Reset Value
EADC_CTL1	EADCx_BA+0x60	R/W	EADC Control1 Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CMP3TRG	CMP2TRG	CMP1TRG	CMP0TRG	Reserved			
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		RESSEL		Reserved			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**CMP3TRG**	**ADC Comparator 3 Trigger EPWM Brake Enable Bit** 0 = Comparator 3 trigger EPWM brake Disabled. 1 = Comparator 3 trigger EPWM brake Enabled.
[22]	**CMP2TRG**	**ADC Comparator 2 Trigger EPWM Brake Enable Bit** 0 = Comparator 2 trigger EPWM brake Disabled. 1 = Comparator 2 trigger EPWM brake Enabled.
[21]	**CMP1TRG**	**ADC Comparator 1 Trigger EPWM Brake Enable Bit** 0 = Comparator 1 trigger EPWM brake Disabled. 1 = Comparator 1 trigger EPWM brake Enabled.
[20]	**CMP0TRG**	**ADC Comparator 0 Trigger EPWM Brake Enable Bit** 0 = Comparator 0 trigger EPWM brake Disabled. 1 = Comparator 0 trigger EPWM brake Enabled.
[19:6]	**Reserved**	Reserved.
[5:4]	**RESSEL**	**Resolution Select Bits** 00 = ADC resolution 12 bits. 01 = ADC resolution 10 bits. 10 = ADC resolution 8 bits. 11 = Reserved.
[3:0]	**Reserved**	Reserved.

EADC Sample Module 0~3 Control Registers (EADC_SCTL0~EADC_SCTL3)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL0	EADCx_BA+0x80	R/W	EADC Sample Module 0 Control Register	0x0000_0011
EADC_SCTL1	EADCx_BA+0x84	R/W	EADC Sample Module 1 Control Register	0x0000_0011
EADC_SCTL2	EADCx_BA+0x88	R/W	EADC Sample Module 2 Control Register	0x0000_0011
EADC_SCTL3	EADCx_BA+0x8C	R/W	EADC Sample Module 3 Control Register	0x0000_0011

31	30	29	28	27	26	25	24
EXTSMPT							
23	22	21	20	19	18	17	16
DBMEN	EXTFEN	EXTREN	TRGSEL				
15	14	13	12	11	10	9	8
TRGDLYCNT							
7	6	5	4	3	2	1	0
TRGDLYDIV		INTPOS	CHSEL				

Bits	Description	
[31:24]	**EXTSMPT**	**EADC Sampling Time Extend** When EADC converting at high conversion rate, the sampling time of analog input voltage may not be enough if input channel loading is heavy, and user can extend EADC sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 EADC clock.
[23]	**DBMEN**	**Double Buffer Mode Enable Bit** 0 = Sample has one sample result register (default). 1 = Sample has two sample result registers.
[22]	**EXTFEN**	**EADC External Trigger Falling Edge Enable Bit** 0 = Falling edge Disabled when EADC selects EADC0_ST as trigger source. 1 = Falling edge Enabled when EADC selects EADC0_ST as trigger source.
[21]	**EXTREN**	**EADC External Trigger Rising Edge Enable Bit** 0 = Rising edge Disabled when EADC selects EADC0_ST as trigger source. 1 = Rising edge Enabled when EADC selects EADC0_ST as trigger source.

Bits	Description	
[20:16]	**TRGSEL**	**EADC Sample Module Start of Conversion Trigger Source Selection** 0H = Disable trigger. 1H = External trigger from EADC0_ST pin input. 2H = EADC ADINT0 interrupt EOC (End of conversion) pulse trigger. 3H = EADC ADINT1 interrupt EOC (End of conversion) pulse trigger. 4H = Timer0 overflow pulse trigger. 5H = Timer1 overflow pulse trigger. 6H = Timer2 overflow pulse trigger. 7H = Timer3 overflow pulse trigger. 8H = EPWM0TG0. 9H = EPWM0TG1. AH = EPWM0TG2. BH = EPWM0TG3. CH = EPWM0TG4. DH = EPWM0TG5. EH = EPWM1TG0. FH = EPWM1TG1. 10H = EPWM1TG2. 11H = EPWM1TG3. 12H = EPWM1TG4. 13H = EPWM1TG5. 14H = BPWM0TG. 15H = BPWM1TG. other = Reserved.
[15:8]	**TRGDLYCNT**	**EADC Sample Module Start of Conversion Trigger Delay Time** Trigger delay time = TRGDLYCNT x EADC_CLK period x n (n=1,2,4,16 from TRGDLYDIV setting).
[7:6]	**TRGDLYDIV**	**EADC Sample Module Start of Conversion Trigger Delay Clock Divider Selection** Trigger delay clock frequency: 00 = EADC_CLK/1. 01 = EADC_CLK/2. 10 = EADC_CLK/4. 11 = EADC_CLK/16.
[5]	**INTPOS**	**Interrupt Flag Position Select** 0 = Set ADIFn (EADC_STATUS2[n], n=0~3) at EADC end of conversion. 1 = Set ADIFn (EADC_STATUS2[n], n=0~3) at EADC start of conversion.

Bits	Description	
[4:0]	**CHSEL**	**EADC Sample Module Channel Selection** 00H = EADC_CH0. 01H = EADC_CH1. 02H = EADC_CH2. 03H = EADC_CH3. 04H = EADC_CH4. 05H = EADC_CH5. 06H = EADC_CH6. 07H = EADC_CH7. 08H = EADC_CH8. 09H = EADC_CH9. 0AH = EADC_CH10. 0BH = EADC_CH11. 0CH = EADC_CH12. 0DH = EADC_CH13. 0EH = EADC_CH14. 0FH = EADC_CH15. 10H = EADC_CH16. 11H = EADC_CH17. 12H = EADC_CH18. 13H = EADC_CH19. 14H = EADC_CH20. 15H = EADC_CH21. 16H = EADC_CH22. 17H = EADC_CH23.

EADC Sample Module 4~15 Control Registers (EADC_SCTL4~EADC_SCTL15)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL4	EADCx_BA+0x90	R/W	EADC Sample Module 4 Control Register	0x0000_0011
EADC_SCTL5	EADCx_BA+0x94	R/W	EADC Sample Module 5 Control Register	0x0000_0011
EADC_SCTL6	EADCx_BA+0x98	R/W	EADC Sample Module 6 Control Register	0x0000_0011
EADC_SCTL7	EADCx_BA+0x9C	R/W	EADC Sample Module 7 Control Register	0x0000_0011
EADC_SCTL8	EADCx_BA+0xA0	R/W	EADC Sample Module 8 Control Register	0x0000_0011
EADC_SCTL9	EADCx_BA+0xA4	R/W	EADC Sample Module 9 Control Register	0x0000_0011
EADC_SCTL10	EADCx_BA+0xA8	R/W	EADC Sample Module 10 Control Register	0x0000_0011
EADC_SCTL11	EADCx_BA+0xAC	R/W	EADC Sample Module 11 Control Register	0x0000_0011
EADC_SCTL12	EADCx_BA+0xB0	R/W	EADC Sample Module 12 Control Register	0x0000_0011
EADC_SCTL13	EADCx_BA+0xB4	R/W	EADC Sample Module 13 Control Register	0x0000_0011
EADC_SCTL14	EADCx_BA+0xB8	R/W	EADC Sample Module 14 Control Register	0x0000_0011
EADC_SCTL15	EADCx_BA+0xBC	R/W	EADC Sample Module 15 Control Register	0x0000_0011

31	30	29	28	27	26	25	24
EXTSMPT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved	**EXTFEN**	**EXTREN**	**TRGSEL**				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TRGDLYCNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TRGDLYDIV		**INTPOS**	**CHSEL**				

Bits	Description	
[31:24]	**EXTSMPT**	**EADC Sampling Time Extend** When EADC converting at high conversion rate, the sampling time of analog input voltage may not be enough if input channel loading is heavy, and software can extend EADC sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 EADC clock.
[23]	**Reserved**	Reserved.
[22]	**EXTFEN**	**EADC External Trigger Falling Edge Enable Bit** 0 = Falling edge Disabled when EADC selects EADC0_ST as trigger source. 1 = Falling edge Enabled when EADC selects EADC0_ST as trigger source.

Bits	Description	
[21]	**EXTREN**	**EADC External Trigger Rising Edge Enable Bit** 0 = Rising edge Disabled when EADC selects EADC0_ST as trigger source. 1 = Rising edge Enabled when EADC selects EADC0_ST as trigger source.
[20:16]	**TRGSEL**	**EADC Sample Module Start of Conversion Trigger Source Selection** 0H = Disable trigger. 1H = External trigger from EADC0_ST pin input. 2H = EADC ADINT0 interrupt EOC pulse trigger. 3H = EADC ADINT1 interrupt EOC pulse trigger. 4H = Timer0 overflow pulse trigger. 5H = Timer1 overflow pulse trigger. 6H = Timer2 overflow pulse trigger. 7H = Timer3 overflow pulse trigger. 8H = EPWM0TG0. 9H = EPWM0TG1. AH = EPWM0TG2. BH = EPWM0TG3. CH = EPWM0TG4. DH = EPWM0TG5. EH = EPWM1TG0. FH = EPWM1TG1. 10H = EPWM1TG2. 11H = EPWM1TG3. 12H = EPWM1TG4. 13H = EPWM1TG5. 14H = BPWM0TG. 15H = BPWM1TG. other = Reserved.
[15:8]	**TRGDLYCNT**	**EADC Sample Module Start of Conversion Trigger Delay Time** Trigger delay time = TRGDLYCNT x EADC_CLK period x n (n=1,2,4,16 from TRGDLYDIV setting).
[7:6]	**TRGDLYDIV**	**EADC Sample Module Start of Conversion Trigger Delay Clock Divider Selection** Trigger delay clock frequency: 00 = EADC_CLK/1. 01 = EADC_CLK/2. 10 = EADC_CLK/4. 11 = EADC_CLK/16.
[5]	**INTPOS**	**Interrupt Flag Position Select** 0 = Set ADIFn (EADC_STATUS2[n], n=0~3) at EADC end of conversion. 1 = Set ADIFn (EADC_STATUS2[n], n=0~3) at EADC start of conversion.

Bits	Description	
[4:0]	**CHSEL**	**EADC Sample Module Channel Selection** 00H = EADC_CH0. 01H = EADC_CH1. 02H = EADC_CH2. 03H = EADC_CH3. 04H = EADC_CH4. 05H = EADC_CH5. 06H = EADC_CH6. 07H = EADC_CH7. 08H = EADC_CH8. 09H = EADC_CH9. 0AH = EADC_CH10. 0BH = EADC_CH11. 0CH = EADC_CH12. 0DH = EADC_CH13. 0EH = EADC_CH14. 0FH = EADC_CH15. Others = Reserved.

EADC Sample Module 16~18 Control Registers (EADC_SCTL16~EADC_SCTL18)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL16	EADCx_BA+0xC0	R/W	EADC Sample Module 16 Control Register	0x0000_0000
EADC_SCTL17	EADCx_BA+0xC4	R/W	EADC Sample Module 17 Control Register	0x0000_0000
EADC_SCTL18	EADCx_BA+0xC8	R/W	EADC Sample Module 18 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
EXTSMPT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:24]	**EXTSMPT**	**ADC Sampling Time Extend** When EADC converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, SW can extend ADC sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.
[23:0]	**Reserved**	Reserved.

EADC Interrupt Source Enable Control Registers (EADC_INTSRC0~EADC_INTSRC3)

Register	Offset	R/W	Description	Reset Value
EADC_INTSRC0	EADCx_BA+0xD0	R/W	EADC Interrupt 0 Source Enable Control Register.	0x0000_0000
EADC_INTSRC1	EADCx_BA+0xD4	R/W	EADC Interrupt 1 Source Enable Control Register.	0x0000_0000
EADC_INTSRC2	EADCx_BA+0xD8	R/W	EADC Interrupt 2 Source Enable Control Register.	0x0000_0000
EADC_INTSRC3	EADCx_BA+0xDC	R/W	EADC Interrupt 3 Source Enable Control Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					SPLIE18	SPLIE17	SPLIE16
15	14	13	12	11	10	9	8
SPLIE15	SPLIE14	SPLIE13	SPLIE12	SPLIE11	SPLIE10	SPLIE9	SPLIE8
7	6	5	4	3	2	1	0
SPLIE7	SPLIE6	SPLIE5	SPLIE4	SPLIE3	SPLIE2	SPLIE1	SPLIE0

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**SPLIE18**	**Sample Module 18 Interrupt Enable Bit** 0 = Sample Module 18 interrupt Disabled. 1 = Sample Module 18 interrupt Enabled.
[17]	**SPLIE17**	**Sample Module 17 Interrupt Enable Bit** 0 = Sample Module 17 interrupt Disabled. 1 = Sample Module 17 interrupt Enabled.
[16]	**SPLIE16**	**Sample Module 16 Interrupt Enable Bit** 0 = Sample Module 16 interrupt Disabled. 1 = Sample Module 16 interrupt Enabled.
[15]	**SPLIE15**	**Sample Module 15 Interrupt Enable Bit** 0 = Sample Module 15 interrupt Disabled. 1 = Sample Module 15 interrupt Enabled.
[14]	**SPLIE14**	**Sample Module 14 Interrupt Enable Bit** 0 = Sample Module 14 interrupt Disabled. 1 = Sample Module 14 interrupt Enabled.
[13]	**SPLIE13**	**Sample Module 13 Interrupt Enable Bit** 0 = Sample Module 13 interrupt Disabled. 1 = Sample Module 13 interrupt Enabled.
[12]	**SPLIE12**	**Sample Module 12 Interrupt Enable Bit** 0 = Sample Module 12 interrupt Disabled. 1 = Sample Module 12 interrupt Enabled.

Bits	Description	
[11]	**SPLIE11**	**Sample Module 11 Interrupt Enable Bit** 0 = Sample Module 11 interrupt Disabled. 1 = Sample Module 11 interrupt Enabled.
[10]	**SPLIE10**	**Sample Module 10 Interrupt Enable Bit** 0 = Sample Module 10 interrupt Disabled. 1 = Sample Module 10 interrupt Enabled.
[9]	**SPLIE9**	**Sample Module 9 Interrupt Enable Bit** 0 = Sample Module 9 interrupt Disabled. 1 = Sample Module 9 interrupt Enabled.
[8]	**SPLIE8**	**Sample Module 8 Interrupt Enable Bit** 0 = Sample Module 8 interrupt Disabled. 1 = Sample Module 8 interrupt Enabled.
[7]	**SPLIE7**	**Sample Module 7 Interrupt Enable Bit** 0 = Sample Module 7 interrupt Disabled. 1 = Sample Module 7 interrupt Enabled.
[6]	**SPLIE6**	**Sample Module 6 Interrupt Enable Bit** 0 = Sample Module 6 interrupt Disabled. 1 = Sample Module 6 interrupt Enabled.
[5]	**SPLIE5**	**Sample Module 5 Interrupt Enable Bit** 0 = Sample Module 5 interrupt Disabled. 1 = Sample Module 5 interrupt Enabled.
[4]	**SPLIE4**	**Sample Module 4 Interrupt Enable Bit** 0 = Sample Module 4 interrupt Disabled. 1 = Sample Module 4 interrupt Enabled.
[3]	**SPLIE3**	**Sample Module 3 Interrupt Enable Bit** 0 = Sample Module 3 interrupt Disabled. 1 = Sample Module 3 interrupt Enabled.
[2]	**SPLIE2**	**Sample Module 2 Interrupt Enable Bit** 0 = Sample Module 2 interrupt Disabled. 1 = Sample Module 2 interrupt Enabled.
[1]	**SPLIE1**	**Sample Module 1 Interrupt Enable Bit** 0 = Sample Module 1 interrupt Disabled. 1 = Sample Module 1 interrupt Enabled.
[0]	**SPLIE0**	**Sample Module 0 Interrupt Enable Bit** 0 = Sample Module 0 interrupt Disabled. 1 = Sample Module 0 interrupt Enabled.

EADC Result Compare Register 0/1/2/3 (EADC_CMP0/1/2/3)

Register	Offset	R/W	Description	Reset Value
EADC_CMP0	EADCx_BA+0xE0	R/W	EADC Result Compare Register 0	0x0000_0000
EADC_CMP1	EADCx_BA+0xE4	R/W	EADC Result Compare Register 1	0x0000_0000
EADC_CMP2	EADCx_BA+0xE8	R/W	EADC Result Compare Register 2	0x0000_0000
EADC_CMP3	EADCx_BA+0xEC	R/W	EADC Result Compare Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPDAT			
23	22	21	20	19	18	17	16
CMPDAT							
15	14	13	12	11	10	9	8
CMPWEN	Reserved			CMPMCNT			
7	6	5	4	3	2	1	0
CMPSPL					CMPCOND	ADCMPIE	ADCMPEN

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:16]	**CMPDAT**	**Comparison Data** The 12 bits data is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage transition without imposing a load on software.
[15]	**CMPWEN**	**Compare Window Mode Enable Bit** 0 = EADCMPF0 (EADC_STATUS2[4]) will be set when EADC_CMP0 compared condition matched. EADCMPF2 (EADC_STATUS2[6]) will be set when EADC_CMP2 compared condition matched 1 = EADCMPF0 (EADC_STATUS2[4]) will be set when both EADC_CMP0 and EADC_CMP1 compared condition matched. EADCMPF2 (EADC_STATUS2[6]) will be set when both EADC_CMP2 and EADC_CMP3 compared condition matched. **Note:** This bit is only present in EADC_CMP0 and EADC_CMP2 register. **Note:** When in compare window mode, the CMPCNT setting only follow EADC_CMP0, EADC_CMP2 registers
[14:12]	**Reserved**	Reserved.
[11:8]	**CMPMCNT**	**Compare Match Count** When the specified EADC sample module analog conversion result matches the compare condition defined by CMPCOND (EADC_CMPn[2], n=0~3), the internal match counter will increase 1. If the compare result does not meet the compare condition, the internal compare match counter will reset to 0. When the internal counter reaches the value to (CMPMCNT +1), the EADCMPFn (EADC_STATUS2[7:4], n=0~3) will be set.

Bits	Description	
[7:3]	**CMPSPL**	**Compare Sample Module Selection** 00000 = Sample Module 0 conversion result EADC_DAT0 is selected to be compared. 00001 = Sample Module 1 conversion result EADC_DAT1 is selected to be compared. 00010 = Sample Module 2 conversion result EADC_DAT2 is selected to be compared. 00011 = Sample Module 3 conversion result EADC_DAT3 is selected to be compared. 00100 = Sample Module 4 conversion result EADC_DAT4 is selected to be compared. 00101 = Sample Module 5 conversion result EADC_DAT5 is selected to be compared. 00110 = Sample Module 6 conversion result EADC_DAT6 is selected to be compared. 00111 = Sample Module 7 conversion result EADC_DAT7 is selected to be compared. 01000 = Sample Module 8 conversion result EADC_DAT8 is selected to be compared. 01001 = Sample Module 9 conversion result EADC_DAT9 is selected to be compared. 01010 = Sample Module 10 conversion result EADC_DAT10 is selected to be compared. 01011 = Sample Module 11 conversion result EADC_DAT11 is selected to be compared. 01100 = Sample Module 12 conversion result EADC_DAT12 is selected to be compared. 01101 = Sample Module 13 conversion result EADC_DAT13 is selected to be compared. 01110 = Sample Module 14 conversion result EADC_DAT14 is selected to be compared. 01111 = Sample Module 15 conversion result EADC_DAT15 is selected to be compared. 10000 = Sample Module 16 conversion result EADC_DAT16 is selected to be compared. 10001 = Sample Module 17 conversion result EADC_DAT17 is selected to be compared. 10010 = Sample Module 18 conversion result EADC_DAT18 is selected to be compared. Others = reserved.
[2]	**CMPCOND**	**Compare Condition** 0= Set the compare condition as that when a 12-bit EADC conversion result is less than the 12-bit CMPDAT (EADC_CMPn [27:16]), the internal match counter will increase one. 1= Set the compare condition as that when a 12-bit EADC conversion result is greater or equal to the 12-bit CMPDAT (EADC_CMPn [27:16]), the internal match counter will increase one. **Note:** When the internal counter reaches the value to (CMPMCNT (EADC_CMPn[11:8], n=0~3) +1), the CMPF bit will be set.
[1]	**ADCMPIE**	**EADC Result Compare Interrupt Enable Bit** 0 = Compare function interrupt Disabled. 1 = Compare function interrupt Enabled. If the compare function is enabled and the compare condition matches the setting of CMPCOND (EADC_CMPn[2], n=0~3) and CMPMCNT (EADC_CMPn[11:8], n=0~3), EADCMPFn (EADC_STATUS2[7:4], n=0~3) will be asserted, in the meanwhile, if ADCMPIE is set to 1, a compare interrupt request is generated.
[0]	**ADCMPEN**	**EADC Result Compare Enable Bit** 0 = Compare Disabled. 1 = Compare Enabled. Set this bit to 1 to enable compare CMPDAT (EADC_CMPn[27:16], n=0~3) with specified sample module conversion result when converted data is loaded into EADC_DAT register.

EADC Status Register 0 (EADC_STATUS0)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS0	EADCx_BA+0xF0	R	EADC Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
OV							
23	22	21	20	19	18	17	16
OV							
15	14	13	12	11	10	9	8
VALID							
7	6	5	4	3	2	1	0
VALID							

Bits	Description	
[31:16]	OV	**EADC_DAT0~15 Overrun Flag** It is a mirror to OV bit in sample module EADC result data register EADC_DATn. (n=0~15).
[15:0]	VALID	**EADC_DAT0~15 Data Valid Flag** It is a mirror of VALID bit in sample module EADC result data register EADC_DATn. (n=0~15).

EADC Status Register 1 (EADC_STATUS1)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS1	EADCx_BA+0xF4	R	EADC Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					OV		
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					VALID		

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:16]	**OV**	**EADC_DAT16~18 Overrun Flag** It is a mirror to OV bit in sample module EADC result data register EADC_DATn. (n=16~18).
[15:3]	**Reserved**	Reserved.
[2:0]	**VALID**	**EADC_DAT16~18 Data Valid Flag** It is a mirror of VALID bit in sample module EADC result data register EADC_DATn. (n=16~18).

EADC Status Register 2 (EADC_STATUS2)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS2	EADCx_BA+0xF8	R/W	EADC Status Register 2	0x0011_0000

31	30	29	28	27	26	25	24
Reserved				AOV	AVALID	STOVF	ADOVIF
23	22	21	20	19	18	17	16
BUSY	Reserved		CHANNEL				
15	14	13	12	11	10	9	8
ADCMPO3	ADCMPO2	ADCMPO1	ADCMPO0	ADOVIF3	ADOVIF2	ADOVIF1	ADOVIF0
7	6	5	4	3	2	1	0
ADCMPF3	ADCMPF2	ADCMPF1	ADCMPF0	ADIF3	ADIF2	ADIF1	ADIF0

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**AOV**	**for All Sample Module EADC Result Data Register Overrun Flags Check (Read Only)** n=0~18. 0 = None of sample module data register overrun flag OVn (EADC_DATn[16]) is set to 1. 1 = Any one of sample module data register overrun flag OVn (EADC_DATn[16]) is set to 1. **Note:** This bit will keep 1 when any OVn Flag is equal to 1.
[26]	**AVALID**	**for All Sample Module EADC Result Data Register EADC_DAT Data Valid Flag Check (Read Only)** n=0~18. 0 = None of sample module data register valid flag VALIDn (EADC_DATn[17]) is set to 1. 1 = Any one of sample module data register valid flag VALIDn (EADC_DATn[17]) is set to 1. **Note:** This bit will keep 1 when any VALIDn Flag is equal to 1.
[25]	**STOVF**	**for All EADC Sample Module Start of Conversion Overrun Flags Check (Read Only)** n=0~18. 0 = None of sample module event overrun flag SPOVFn (EADC_OVSTS[n]) is set to 1. 1 = Any one of sample module event overrun flag SPOVFn (EADC_OVSTS[n]) is set to 1. **Note:** This bit will keep 1 when any SPOVFn Flag is equal to 1.
[24]	**ADOVIF**	**All EADC Interrupt Flag Overrun Bits Check (Read Only)** n=0~3. 0 = None of ADINT interrupt flag ADOVIFn, n=0~3 is overwritten to 1. 1 = Any one of ADINT interrupt flag ADOVIFn, n=0~3 is overwritten to 1. **Note:** This bit will keep 1 when any ADOVIFn Flag is equal to 1.
[23]	**BUSY**	**Busy/Idle (Read Only)** 0 = EADC is in idle state. 1 = EADC is busy at conversion. **Note:** This flag will be high after 4*EADC_CLK cycles when the trigger source is coming.
[22:21]	**Reserved**	Reserved.

Bits	Description	
[20:16]	**CHANNEL**	**Current Conversion Channel (Read Only)** This filed reflects EADC current conversion channel when BUSY=1. 00H = EADC_CH0. 01H = EADC_CH1. 02H = EADC_CH2. 03H = EADC_CH3. 04H = EADC_CH4. 05H = EADC_CH5. 06H = EADC_CH6. 07H = EADC_CH7. 08H = EADC_CH8. 09H = EADC_CH9. 0AH = EADC_CH10. 0BH = EADC_CH11. 0CH = EADC_CH12. 0DH = EADC_CH13. 0EH = EADC_CH14. 0FH = EADC_CH15. 10H = V_{BG}. 11H = V_{TEMP}. 12H = V_{BAT}/4.
[15]	**ADCMPO3**	**EADC Compare 3 Output Status (Read Only)** The 12 bits compare3 data CMPDAT3 (EADC_CMP3[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT less than CMPDAT3 setting. 1 = Conversion result in EADC_DAT great than or equal to CMPDAT3 setting.
[14]	**ADCMPO2**	**EADC Compare 2 Output Status (Read Only)** The 12 bits compare2 data CMPDAT2 (EADC_CMP2[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT less than CMPDAT2 setting. 1 = Conversion result in EADC_DAT great than or equal to CMPDAT2 setting.
[13]	**ADCMPO1**	**EADC Compare 1 Output Status (Read Only)** The 12 bits compare1 data CMPDAT1 (EADC_CMP1[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT less than CMPDAT1 setting. 1 = Conversion result in EADC_DAT great than or equal to CMPDAT1 setting.
[12]	**ADCMPO0**	**EADC Compare 0 Output Status (Read Only)** The 12 bits compare0 data CMPDAT0 (EADC_CMP0[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT less than CMPDAT0 setting. 1 = Conversion result in EADC_DAT great than or equal CMPDAT0 setting.
[11]	**ADOVIF3**	**EADC ADINT3 Interrupt Flag Overrun** 0 = ADINT3 interrupt flag is not overwritten to 1. 1 = ADINT3 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.

Bits	Description	
[10]	**ADOVIF2**	**EADC ADINT2 Interrupt Flag Overrun** 0 = ADINT2 interrupt flag is not overwritten to 1. 1 = ADINT2 interrupt flag is s overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[9]	**ADOVIF1**	**EADC ADINT1 Interrupt Flag Overrun** 0 = ADINT1 interrupt flag is not overwritten to 1. 1 = ADINT1 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[8]	**ADOVIF0**	**EADC ADINT0 Interrupt Flag Overrun** 0 = ADINT0 interrupt flag is not overwritten to 1. 1 = ADINT0 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[7]	**ADCMPF3**	**EADC Compare 3 Flag** When the specific sample module EADC conversion result meets setting condition in EADC_CMP3 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP3 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP3 register setting. **Note:** This bit is cleared by writing 1 to it.
[6]	**ADCMPF2**	**EADC Compare 2 Flag** When the specific sample module EADC conversion result meets setting condition in EADC_CMP2 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP2 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP2 register setting. **Note:** This bit is cleared by writing 1 to it.
[5]	**ADCMPF1**	**EADC Compare 1 Flag** When the specific sample module EADC conversion result meets setting condition in EADC_CMP1 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP1 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP1 register setting. **Note:** This bit is cleared by writing 1 to it.
[4]	**ADCMPF0**	**EADC Compare 0 Flag** When the specific sample module EADC conversion result meets setting condition in EADC_CMP0 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP0 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP0 register setting. **Note:** This bit is cleared by writing 1 to it.
[3]	**ADIF3**	**EADC ADINT3 Interrupt Flag** 0 = No ADINT3 interrupt pulse received. 1 = ADINT3 interrupt pulse has been received. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:**This bit indicates whether an EADC conversion of specific sample module has been completed

Bits	Description	
[2]	**ADIF2**	**EADC ADINT2 Interrupt Flag** 0 = No ADINT2 interrupt pulse received. 1 = ADINT2 interrupt pulse has been received. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:**This bit indicates whether an EADC conversion of specific sample module has been completed
[1]	**ADIF1**	**EADC ADINT1 Interrupt Flag** 0 = No ADINT1 interrupt pulse received. 1 = ADINT1 interrupt pulse has been received. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:**This bit indicates whether an EADC conversion of specific sample module has been completed
[0]	**ADIF0**	**EADC ADINT0 Interrupt Flag** 0 = No ADINT0 interrupt pulse received. 1 = ADINT0 interrupt pulse has been received. **Note 1:** This bit is cleared by writing 1 to it. **Note 2:**This bit indicates whether an EADC conversion of specific sample module has been completed

EADC Status Register 3 (EADC_STATUS3)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS3	EADCx_BA+0xFC	R	EADC Status Register 3	0x0000_001F

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			CURSPL				

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**CURSPL**	**EADC Current Sample Module (Read Only)** This register shows the current EADC is controlled by which sample module control logic modules. If the EADC is Idle, the bit filed will be set to 0x1F.

EADC Double Data Register n for Sample Module n (EADC_DDAT0~3)

Register	Offset	R/W	Description	Reset Value
EADC_DDAT0	EADCx_BA+0x100	R	EADC Double Data Register 0 for Sample Module 0	0x0000_0000
EADC_DDAT1	EADCx_BA+0x104	R	EADC Double Data Register 1 for Sample Module 1	0x0000_0000
EADC_DDAT2	EADCx_BA+0x108	R	EADC Double Data Register 2 for Sample Module 2	0x0000_0000
EADC_DDAT3	EADCx_BA+0x10C	R	EADC Double Data Register 3 for Sample Module 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						VALID	OV
15	14	13	12	11	10	9	8
RESULT							
7	6	5	4	3	2	1	0
RESULT							

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**VALID**	**Valid Flag** 0 = Double data in RESULT (EADC_DDATn[15:0]) is not valid. 1 = Double data in RESULT (EADC_DDATn[15:0]) is valid. This bit is set to 1 when corresponding sample module channel analog input conversion is completed and cleared by hardware after EADC_DDATn register is read. (n=0~3).
[16]	**OV**	**Overrun Flag** 0 = Double Data in RESULT (EADC_DDATn[15:0], n=0~3) is recent conversion result. 1 = Double Data in RESULT (EADC_DDATn[15:0], n=0~3) is overwrite. If converted data in RESULT[15:0] has not been read before new conversion result is loaded to this register, OV is set to 1. It is cleared by hardware after EADC_DDAT register is read.
[15:0]	**RESULT**	**EADC Conversion Results** This field contains 12 bits conversion results. The 12-bit EADC conversion result with unsigned format will be filled in RESULT [11:0] and zero will be filled in RESULT [15:12].

EADC Calibration Control Register (EADC_CALCTL)

Register	Offset	R/W	Description	Reset Value
EADC_CALCTL	EADCx_BA+0x114	R/W	EADC Calibration Control Register	0x0000_0020

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**CALIE**	**CAL**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**CALIE**	**Calibration Interrupt Enable Bit** 0 = Calibration interrupt Disabled. 1= Calibration interrupt Enabled.
[0]	**CAL**	**Calibration Enable Bit** 0 = = Calibration Disabled. 1 = = Calibration Enabled. **Note:** This bit is hardware auto cleared when calibration is done

EADC Calibration Status Register (EADC_CALSR)

Register	Offset	R/W	Description	Reset Value
EADC_CALSR	EADCx_BA+0x118	R/W	EADC Calibration Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**CALIF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**CALIF**	**Calibration Finish Interrupt Flag** If calibration is finished, this flag will be set to 1. It is cleared by writing 1 to it.
[15:0]	**Reserved**	Reserved.

EADC PDMA Control Register (EADC_PDMACTL)

Register	Offset	R/W	Description	Reset Value
EADC_PDMACTL	EADCx_BA+0x130	R/W	EADC PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**PDMATEN**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PDMATEN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDMATEN							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**PDMATEN**	**PDMA Transfer Enable Bit** When EADC conversion is completed, the converted data is loaded into EADC_DATn (n: 0 ~ 18) register, user can enable this bit to generate a PDMA data transfer request. 0 = PDMA data transfer Disabled. 1 = PDMA data transfer Enabled. **Note:** When setting this bit field to 1, user must set ADCIENn (EADC_CTL[5:2], n=0~3) = 0 to disable interrupt.

EADC Sample Module 0~15 Control Registers1 (EADC_M0CTL1~EADC_M15CTL1)

Register	Offset	R/W	Description	Reset Value
EADC_M0CTL1	EADCx_BA+0x140	R/W	EADC Sample Module0 Control Register 1	0x0000_0000
EADC_M1CTL1	EADCx_BA+0x144	R/W	EADC Sample Module1 Control Register 1	0x0000_0000
EADC_M2CTL1	EADCx_BA+0x148	R/W	EADC Sample Module2 Control Register 1	0x0000_0000
EADC_M3CTL1	EADCx_BA+0x14C	R/W	EADC Sample Module3 Control Register 1	0x0000_0000
EADC_M4CTL1	EADCx_BA+0x150	R/W	EADC Sample Module4 Control Register 1	0x0000_0000
EADC_M5CTL1	EADCx_BA+0x154	R/W	EADC Sample Module5 Control Register 1	0x0000_0000
EADC_M6CTL1	EADCx_BA+0x158	R/W	EADC Sample Module6 Control Register 1	0x0000_0000
EADC_M7CTL1	EADCx_BA+0x15C	R/W	EADC Sample Module7 Control Register 1	0x0000_0000
EADC_M8CTL1	EADCx_BA+0x160	R/W	EADC Sample Module8 Control Register 1	0x0000_0000
EADC_M9CTL1	EADCx_BA+0x164	R/W	EADC Sample Module9 Control Register 1	0x0000_0000
EADC_M10CTL1	EADCx_BA+0x168	R/W	EADC Sample Module10 Control Register 1	0x0000_0000
EADC_M11CTL1	EADCx_BA+0x16C	R/W	EADC Sample Module11 Control Register 1	0x0000_0000
EADC_M12CTL1	EADCx_BA+0x170	R/W	EADC Sample Module12 Control Register 1	0x0000_0000
EADC_M13CTL1	EADCx_BA+0x174	R/W	EADC Sample Module13 Control Register 1	0x0000_0000
EADC_M14CTL1	EADCx_BA+0x178	R/W	EADC Sample Module14 Control Register 1	0x0000_0000
EADC_M15CTL1	EADCx_BA+0x17C	R/W	EADC Sample Module15 Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACU				**Reserved**		**AVG**	**ALIGN**

Bits	Description	
[31:8]	**Reserved**	Reserved.

Bits	Description	
[7:4]	**ACU**	**Number of Accumulated Conversion Results Selection** 0000 = 1 conversion result will be accumulated. 0001 = 2 conversion result will be accumulated. 0010 = 4 conversion result will be accumulated. 0011 = 8 conversion result will be accumulated. 0100 = 16 conversion result will be accumulated. 0101 = 32 conversion result will be accumulated. 0110 = 64 conversion result will be accumulated. 0111 = 128 conversion result will be accumulated. 1000 = 256 conversion result will be accumulated. Others = Reserved.
[3:2]	**Reserved**	Reserved.
[1]	**AVG**	**Average Mode Selection** 0 = Conversion results will be stored in data register without averaging. 1 = Conversion results in data register will be averaged. **Note:** This bit needs to work with ACU (EADC_MnCTL1[7:4], n=0~23).
[0]	**ALIGN**	**Alignment Selection** 0 = The conversion result will be right aligned in data register. 1 = The conversion result will be left aligned in data register.

6.45 Digital to Analog Converter (DAC)

6.45.1 Overview

The DAC module is a 12-bit, voltage output digital-to-analog converter. It can be configured to 12- or 8-bit output mode and can be used in conjunction with the PDMA controller. The DAC integrates a voltage output buffer that can be used to reduce output impendence and drive external loads directly without having to add an external operational amplifier.

6.45.2 Features

- Analog output voltage range: 0~AV_{DD}.
- Supports 12- or 8-bit output mode.
- Rail to rail settle time 8us.
- Supports up to two 12-bit 1 MSPS voltage type DAC.
- Reference voltage from internal reference voltage (INT_VREF), V_{REF} pin.
- DAC maximum conversion updating rate 1 MSPS.
- Supports voltage output buffer mode and bypass voltage output buffer mode.
- Supports software and hardware trigger, including Timer0~3, EPWM0, EPWM1, and external trigger pin to start DAC conversion.
- Supports PDMA mode.
- Supports group mode of synchronized update capability for two DACs.

6.45.3 Block Diagram

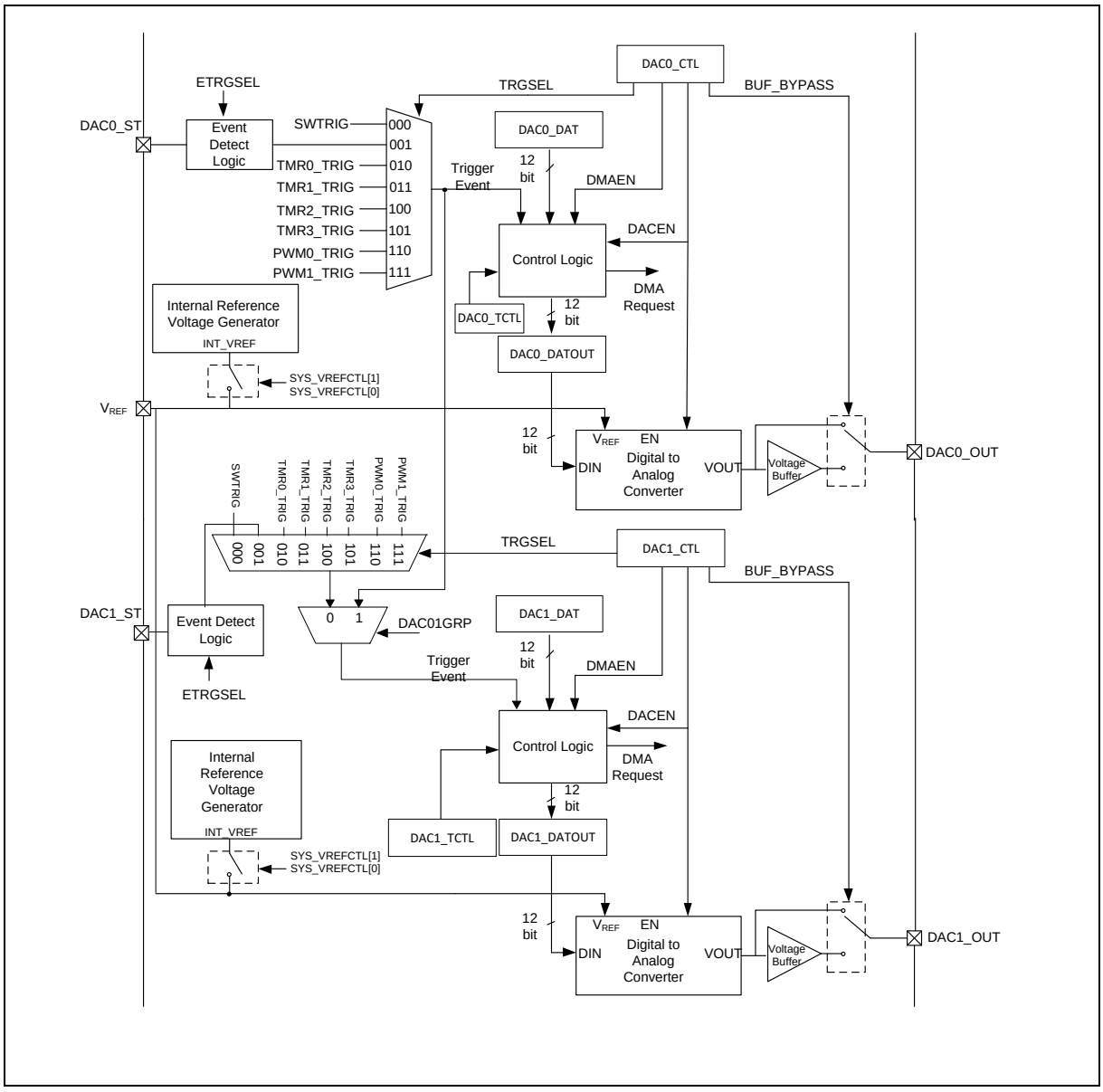


Figure 6.45-1 Digital-to-Analog Converter Block Diagram

6.45.4 Basic Configuration

6.45.4.1 DAC0 Basic Configuration

- Clock source Configuration
  - Enable DAC0 peripheral clock in DACCKEN (CLK_APBCLK1[12]).
- Reset Configuration
  - Reset DAC0 controller in DACRST (SYS_IPRST2[12]).

6.45.4.2 DAC1 Basic Configuration

- Clock source Configuration
  - Enable DAC1 peripheral clock in DACCKEN (CLK_APBCLK1[12]).

- Reset Configuration
  - Reset DAC1 controller in DACRST (SYS_IPRST2[12]).

6.45.5 Functional Description

6.45.5.1 DAC Output

The DAC is a 12-bit voltage output digital-to-analog converter and can be configured as 12- or 8-bit operation mode. The DAC integrates a voltage output buffer that can be used to reduce output impendence and drive external loads directly without having to add an external operational amplifier. The DAC channel output buffer can be enabled and disabled by BYPASS (DACn_CTL[8]), n=0, 1. The maximum DAC output voltage is limited to the selected reference voltage source.

6.45.5.2 DAC Reference Voltage

The DAC reference voltage is shared with EADC reference voltage and it is configured by VREFCTL (SYS_VREFCTL[4:0]) in system manager control registers. The reference voltage for the DAC can be configured from external reference voltage pin (V_{REF}) or internal reference voltage generator (INT_VREF).

6.45.5.3 DAC Data Format

The DAC supports conversion data left alignment or right alignment mode. Depending on the selected configuration mode, the data needs to be written into the specified register as follows:

- 12-bit left alignment: user has to load data into DACn_DAT[15:4] bits. DACn_DAT[31:16] and DACn_DAT[3:0] are ignored in DAC conversion.
- 12-bit right alignment: user has to load data into DACn_DAT[11:0] bits, DACn_DAT[31:12] are ignored in DAC conversion.

While DAC is working in 8-bit mode, alignment setting has no effect. To enable 8-bit mode, set BWSEL(DACn_CTL[15:14]) to 01. Otherwise, keep BWSEL as 00.

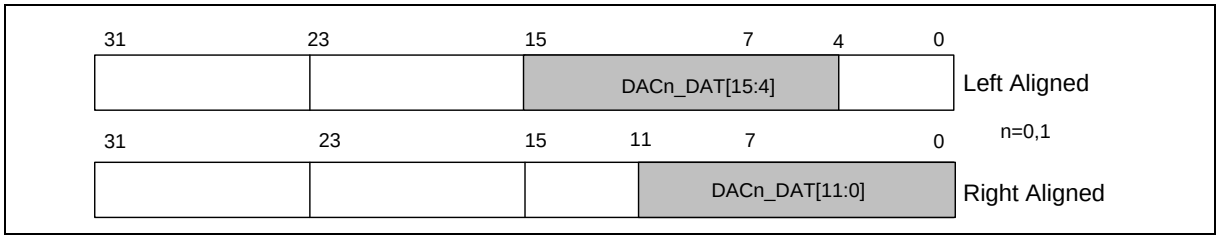


Figure 6.45-2 Data Holding Register Format

6.45.5.4 DAC Conversion

Any data transfer to the DAC channel is performed by loading the data into DACn_DAT register. Figure 6.45-3 shows the DAC conversion started by software write operation. When user writes the conversion data to data holding register DACn_DAT, the data is loaded into data output register DACn_DATOUT by hardware and DAC starts data conversion after one PCLK (APB clock) clock cycle. Figure 6.45-4 shows the DAC conversion started by hardware trigger (external pin DACn_ST, timer trigger event or EPWM timer trigger event). The data stored in the DACn_DAT register is automatically transferred to the data output buffer DACn_DATOUT after occurring one PCLK (APB clock) the event.

When DAC data output register DACn_DATOUT is loaded with the DACn_DAT contents, the analog output voltage becomes available after specified conversion settling time. The conversion settling time is 8us when 12- bit input code transition from lowest code (0x000) to highest code (0xFFF). Two adjacent codes conversion settling time is 1us. The DAC controller provides a 10-bit time counter for user to count the conversion time period. In continuous conversion operation, user needs to write appropriate

value to SETTLET (DACn_TCTL[9:0]) to define DAC conversion time period. The value must be longer than DAC conversion settling time which is specified in DAC electric characteristic table. For example, when DAC controller APB clock speed is 80 MHz and DAC conversion settling time is 8us, the selected SETTLET value must be greater than 0x280. When the conversion is started, the conversion finish flag FINISH (DACn_STATUS[0]) is cleared to 0 by hardware and set to 1 after the time counter counts to SETTLET. Note that n=0,1.

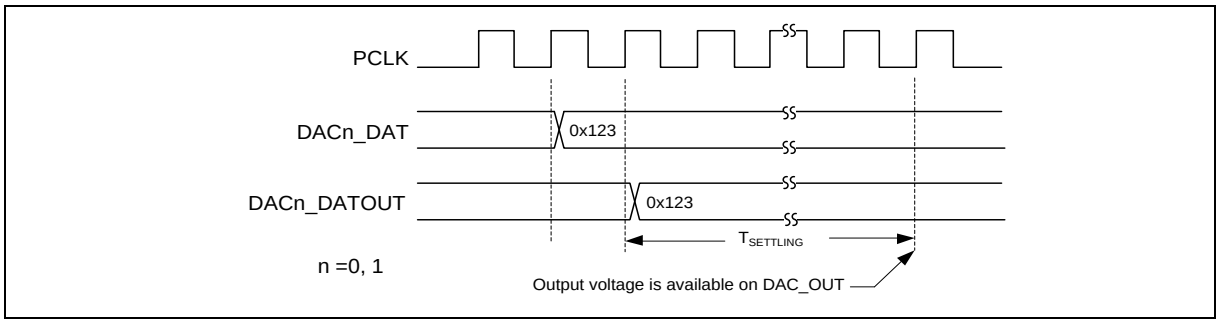


Figure 6.45-3 DAC Conversion Started by Software Write Trigger

6.45.5.5 DAC Output Voltage

Digital inputs are converted to output voltage on a linear conversion between 0 and reference voltage V_{REF}. The analog output voltage on DAC pin is determined by the following equation:

$$DACOUT = V_{REF} * \frac{DATnOUT[11:0]}{4096}, \quad n=0,1$$

6.45.5.6 DAC Trigger Selection

The DAC conversion can be started by writing DACn_DAT, software trigger or hardware trigger. When TRGEN (DACn_CTL[4]) is 0, the data conversion is started by writing DACn_DAT register. When TRGEN (DACn_CTL[4]) is 1, the data conversion is started by external DACn_ST pin, timer event, or EPWM timer event. If the software trigger is selected, the conversion starts once the SWTRG (DACn_SWTRG[0]) is set to 1. The SWTRG is cleared to 0 by hardware automatically when DACn_DATOUT has been loaded with DACDAT content. The TRGSEL (DACn_CTL[7:5]) determines which one of eight events is selected to start the conversion.

When DAC detects a rising edge on the selected trigger event input, the last data stored in DACDAT is transferred into the DACn_DATOUT[11:0] and DAC starts converting after one PCLK (APB clock) clock cycle. Note that n=0,1.

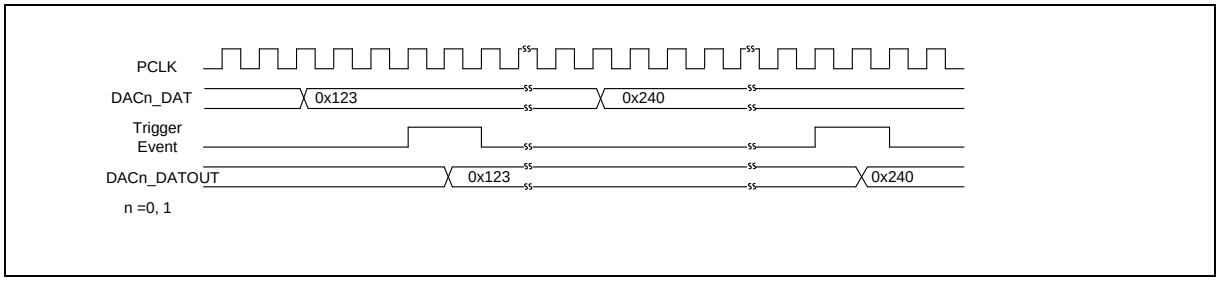


Figure 6.45-4 DAC Conversion Started by Hardware Trigger Event

6.45.5.7 DAC Group Mode

The DAC0 and DAC1 can be grouped together by setting GRPEN (DAC0_CTL[16]) to synchronize the update of each DAC output. Hardware ensures that these two DACs will be updated simultaneously in group mode. In group mode, DAC1_CTL and DAC1_TCTL has no effect. DAC1's behavior is controlled by DAC_GRPDAT, DAC0_CTL and DAC0_TCTL. In group mode, the additional option is write DAC0, DAC1 conversion data into DAC_GRPDAT rather than DAC0_DAT and DAC1_DAT, which advantage is only need one DMA channel in DMA operation instead of two DMA channel setting. It is only useful

in group mode and higher write conversion data access priority than write conversion data into DAC0_DAT and DAC1_DAT. Figure 6.45-5 shows an example of group mode with write conversion data into DAC0_DAT and DAC1_DAT and compared with normal mode.

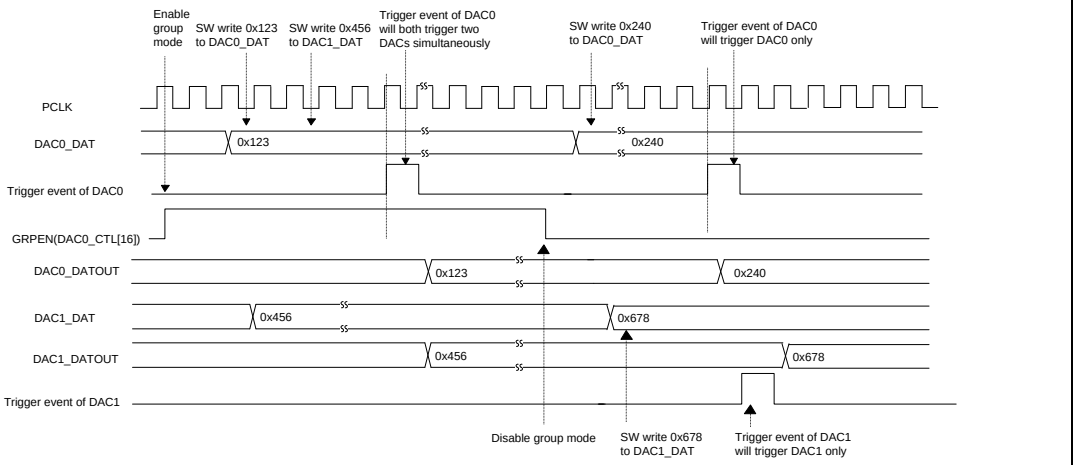


Figure 6.45-5 DAC0 and DAC1 Group and Ungroup Update Example

6.45.5.8 DMA Operation

A DAC DMA request is generated when a hardware trigger event occurs while DMAEN (DACn_CTL[2]) is set. The content of DACn_DAT is transferred to the DACn_DATOUT[11:0] and DAC starts data conversion after one PCLK (APB clock) clock cycle. The new transferred data by PDMA in DACn_DAT will be converted when next trigger event arrives. Figure 6.45-6 shows the DAC PDMA under-run condition, when the second DMA request trigger event arrives before the first conversion finish, then no new PDMA request is issued and DMA under-run flag DMAUDR (DACn_STATUS[1]) is set 1 to report the error condition. DMA data transfers are then disabled and no further DMA request is treated and DAC continues to convert last data. An interrupt is also generated if the corresponding DMAURIEN (DACn_CTL[3]) is enabled. User has to change the trigger event frequency in timer or EPWM timer and then start DAC conversion again. Note that n=0,1.

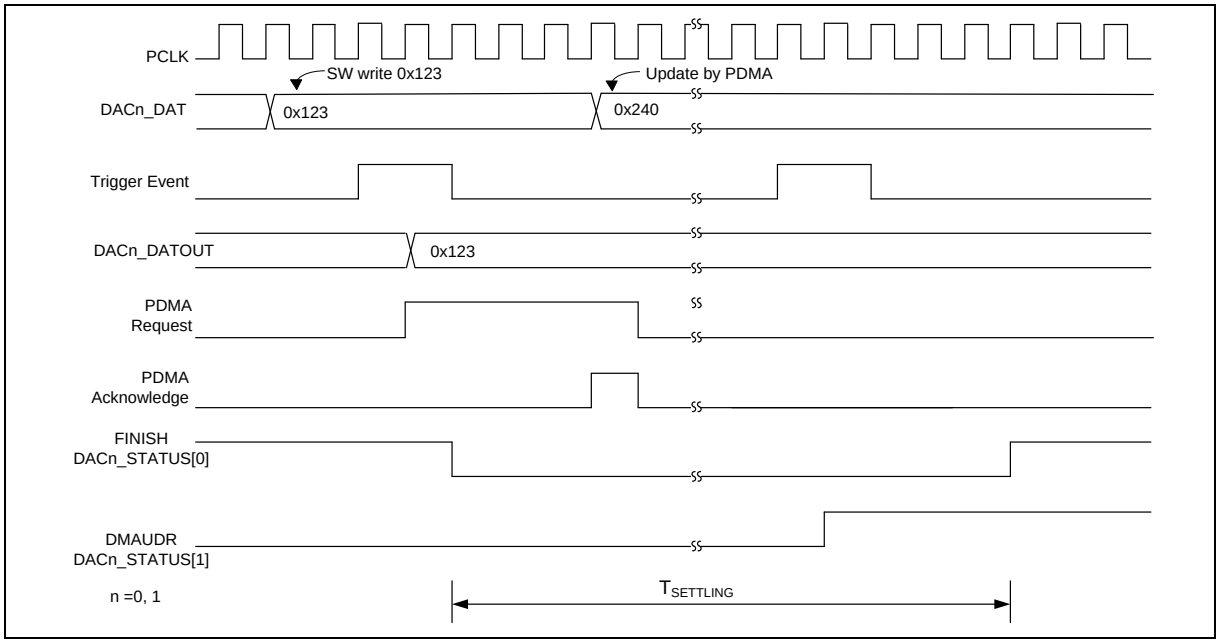


Figure 6.45-6 DAC PDMA Under-Run Condition Example

The DMA request can also be generated by software enable, user sets DMAEN (DACn_CTL[2]) to 1 and TRGEN (DACn_CTL[4]) to 0, DMA request is generated periodically according to the conversion time defined by SETTLET (DACn_TCTL[9:0]) value. DAC output is updated periodically. When user clears DMAEN (DACn_CTL[2]) to 0, DAC controller will stop issuing next new PDMA transfer request. Figure 6.45-7 provides an example of DAC continuous conversion with software PDMA mode. Note that n = 0,1.

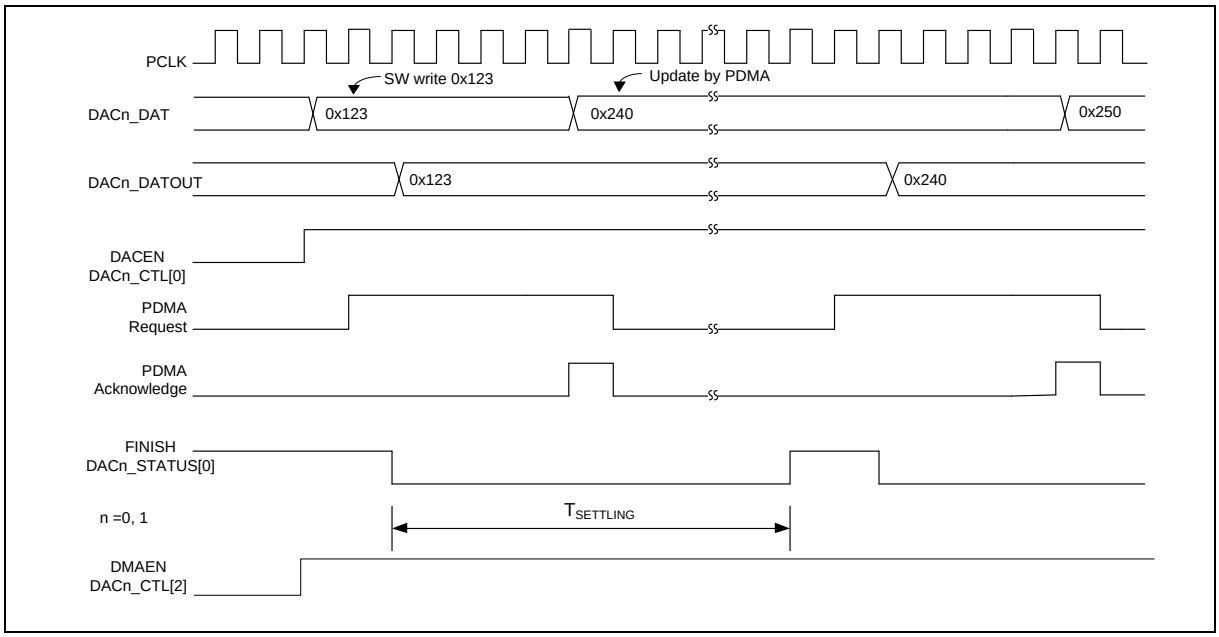


Figure 6.45-7 DAC Continuous Conversion with Software PDMA Mode

6.45.5.9 Interrupt Sources

There are two interrupt sources in the DAC controller, one is DAC data conversion finish interrupt and the other is DMA under-run interrupt as shown in Figure 6.45-8. When DAC conversion is finished, the

FINISH (DACn_STATUS[0]) is set to 1 and an interrupt occurs while DACIEN (DACn_CTL[1]) is enabled. If a new DMA trigger event occurs during DAC data conversion period, the DMA under-run flag DMAUDR (DACn_STATUS[1]) is generated and an interrupt occurs if DMAURIEN (DACn_CTL[3]) is enabled. Note that n = 0,1.

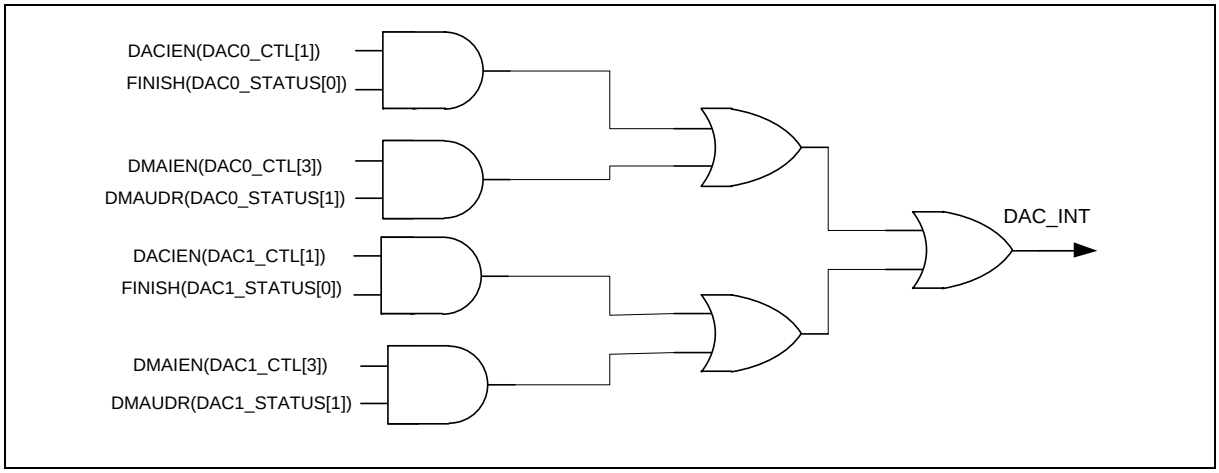


Figure 6.45-8 DAC Interrupt Source

6.45.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
DAC Base Address: **DAC_BA = 0x4004_7000**				
DAC0_CTL	DAC_BA+0x00	R/W	DAC0 Control Register	0x0000_0000
DAC0_SWTRG	DAC_BA+0x04	R/W	DAC0 Software Trigger Control Register	0x0000_0000
DAC0_DAT	DAC_BA+0x08	R/W	DAC0 Data Holding Register	0x0000_0000
DAC0_DATOUT	DAC_BA+0x0C	R	DAC0 Data Output Register	0x0000_0000
DAC0_STATUS	DAC_BA+0x10	R/W	DAC0 Status Register	0x0000_0000
DAC0_TCTL	DAC_BA+0x14	R/W	DAC0 Timing Control Register	0x0000_0000
DAC_GRPDAT	DAC_BA+0x30	R/W	DAC Group Mode Data Holding Register	0x0000_0000
DAC1_CTL	DAC_BA+0x40	R/W	DAC1 Control Register	0x0000_0000
DAC1_SWTRG	DAC_BA+0x44	R/W	DAC1 Software Trigger Control Register	0x0000_0000
DAC1_DAT	DAC_BA+0x48	R/W	DAC1 Data Holding Register	0x0000_0000
DAC1_DATOUT	DAC_BA+0x4C	R	DAC1 Data Output Register	0x0000_0000
DAC1_STATUS	DAC_BA+0x50	R/W	DAC1 Status Register	0x0000_0000
DAC1_TCTL	DAC_BA+0x54	R/W	DAC1 Timing Control Register	0x0000_0000

6.45.7 Register Description

DAC0 Control Register (DAC0_CTL)

Register	Offset	R/W	Description	Reset Value
DAC0_CTL	DAC_BA+0x00	R/W	DAC0 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							GRPEN
15	14	13	12	11	10	9	8
BWSEL		ETRGSEL		Reserved	LALIGN	Reserved	BYPASS
7	6	5	4	3	2	1	0
TRGSEL			TRGEN	DMAURIEN	DMAEN	DACIEN	DACEN

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**GRPEN**	**DAC Group Mode Enable Bit** 0 = DAC0 and DAC1 are not grouped. 1 = DAC0 and DAC1 are grouped.
[15:14]	**BWSEL**	**DAC Data Bit-width Selection** 00 = data is 12 bits. 01 = data is 8 bits. Others = reserved.
[13:12]	**ETRGSEL**	**External Pin Trigger Selection** 00 = Low level trigger. 01 = High level trigger. 10 = Falling edge trigger. 11 = Rising edge trigger.
[11]	**Reserved**	Reserved.
[10]	**LALIGN**	**DAC Data Left-aligned Enabled Bit** 0 = Right alignment. 1 = Left alignment.
[9]	**Reserved**	Reserved.
[8]	**BYPASS**	**Bypass Buffer Mode** 0 = Output voltage buffer Enabled. 1 = Output voltage buffer Disabled.
[7:5]	**TRGSEL**	**Trigger Source Selection** 000 = Software trigger. 001 = External pin DAC0_ST trigger.

		010 = Timer 0 trigger. 011 = Timer 1 trigger. 100 = Timer 2 trigger. 101 = Timer 3 trigger. 110 = EPWM0 trigger. 111 = EPWM1 trigger.
[4]	**TRGEN**	**Trigger Mode Enable Bit** 0 = DAC event trigger mode Disabled. 1 = DAC event trigger mode Enabled.
[3]	**DMAURIEN**	**DMA Under-run Interrupt Enable Bit** 0 = DMA under-run interrupt Disabled. 1 = DMA under-run interrupt Enabled.
[2]	**DMAEN**	**DMA Mode Enable Bit** 0 = DMA mode Disabled. 1 = DMA mode Enabled.
[1]	**DACIEN**	**DAC Interrupt Enable Bit** 0 = DAC interrupt Disabled. 1 = DAC interrupt Enabled.
[0]	**DACEN**	**DAC Enable Bit** 0 = DAC Disabled. 1 = DAC Enabled.

DAC0 Software Trigger Control Register (DAC0_SWTRG)

Register	Offset	R/W	Description	Reset Value
DAC0_SWTRG	DAC_BA+0x04	R/W	DAC0 Software Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**SWTRG**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SWTRG**	**Software Trigger** 0 = Software trigger Disabled. 1 = Software trigger Enabled. **Note:** User writes this bit to generate one shot pulse and it is cleared to 0 by hardware automatically; reading this bit will always get 0.

DAC0 Data Holding Register (DAC0_DAT)

Register	Offset	R/W	Description	Reset Value
DAC0_DAT	DAC_BA+0x08	R/W	DAC0 Data Holding Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DACDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DACDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DACDAT**	**DAC 12-bit Holding Data** These bits are written by user software which specifies 12-bit conversion data for DAC output. The unused bits (DAC0_DAT[3:0] in left-alignment mode and DAC0_DAT[15:12] in right alignment mode) are ignored by DAC controller hardware. 12 bit left alignment: user has to load data into DAC0_DAT[15:4] bits. 12 bit right alignment: user has to load data into DAC0_DAT[11:0] bits.

DAC0 Data Output Register (DAC0_DATOUT)

Register	Offset	R/W	Description	Reset Value
DAC0_DATOUT	DAC_BA+0x0C	R	DAC0 Data Output Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**DATOUT**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATOUT							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**DATOUT**	**DAC 12-bit Output Data** These bits are current digital data for DAC output conversion. It is loaded from DAC0_DAT register and user cannot write it directly.

DAC0 Status Register (DAC0_STATUS)

Register	Offset	R/W	Description	Reset Value
DAC0_STATUS	DAC_BA+0x10	R/W	DAC0 Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							BUSY
7	6	5	4	3	2	1	0
Reserved						DMAUDR	FINISH

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BUSY**	**DAC Busy Flag (Read Only)** 0 = DAC is ready for next conversion. 1 = DAC is busy in conversion.
[7:2]	**Reserved**	Reserved.
[1]	**DMAUDR**	**DMA Under-run Interrupt Flag** 0 = No DMA under-run error condition occurred. 1 = DMA under-run error condition occurred. **Note:** User writes 1 to clear this bit.
[0]	**FINISH**	**DAC Conversion Complete Finish Flag** 0 = DAC is in conversion state. 1 = DAC conversion finish. **Note:** This bit is set to 1 when conversion time counter counts to SETTLET. It is cleared to 0 when DAC starts a new conversion. User writes 1 to clear this bit to 0.

DAC0 Timing Control Register (DAC0_TCTL)

Register	Offset	R/W	Description	Reset Value
DAC0_TCTL	DAC_BA+0x14	R/W	DAC0 Timing Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**SETTLET**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETTLET							

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**SETTLET**	**DAC Output Settling Time** User software needs to write appropriate value to these bits to meet DAC conversion settling time base on PCLK (APB clock) speed. For example, DAC controller clock speed is 80 MHz and DAC conversion settling time is 1 us, SETTLETvalue must be greater than 0x50. SELTTLET = DAC controller clock speed x settling time.

DAC Group Data Register (DAC_GRPDAT)

Register	Offset	R/W	Description	Reset Value
DAC_GRPDAT	DAC_BA+0x30	R/W	DAC Group Mode Data Holding Register	0x0000_0000

31	30	29	28	27	26	25	24
DAC1DAT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DAC1DAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DAC0DAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DAC0DAT							

Bits	Description	
[31:16]	**DAC1DAT**	**DAC1 12-bit Holding Data** In group mode, user can write these bits for DAC1 12-bit conversion data. The unused bits (DAC_GRPDAT[3:0] in left-alignment mode and DAC_GRPDAT[15:12] in right alignment mode) are ignored by DAC controller hardware. 12 bit left alignment: user has to load data into DAC_GRPDAT[15:4] bits. 12 bit right alignment: user has to load data into DAC_GRPDAT[11:0] bits. **Note:** In group mode, user can write 12-bit conversion data for DAC1 in DAC_GRPDAT[31:16] or DAC1_DAT[15:0]. The advantage of writing 12-bit conversion data in DAC_GRPDAT[31:16] is that can share one PDMA transfer mechanism. **Note:** Write 12-bit conversion data in DAC0_DAT[15:0] or DAC1_DAT[15:0] have individual PDMA transfer mechanism between two DACs
[15:0]	**DAC0DAT**	**DAC0 12-bit Holding Data** These bits are written by user software which specifies 12-bit conversion data for DAC output. The unused bits (DAC_GRPDAT[3:0] in left-alignment mode and DAC_GRPDAT[15:12] in right alignment mode) are ignored by DAC controller hardware. 12 bit left alignment: user has to load data into DAC_GRPDAT[15:4] bits. 12 bit right alignment: user has to load data into DAC_GRPDAT[11:0] bits. **Note:** In group mode, user can write 12-bit conversion data for DAC0 in DAC_GRPDAT[15:0] or DAC0_DAT[15:0]. The advantage of writing 12-bit conversion data in DAC_GRPDAT[15:0] is that can share one PDMA transfer mechanism. **Note:** Write 12-bit conversion data in DAC0_DAT[15:0] or DAC1_DAT[15:0] have individual PDMA transfer mechanism between two DACs

DAC1 Control Register (DAC1_CTL)

Register	Offset	R/W	Description	Reset Value
DAC1_CTL	DAC_BA+0x40	R/W	DAC1 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
BWSEL		ETRGSEL		Reserved	LALIGN	Reserved	BYPASS
7	6	5	4	3	2	1	0
TRGSEL			TRGEN	DMAURIEN	DMAEN	DACIEN	DACEN

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:14]	**BWSEL**	**DAC Data Bit-width Selection** 00 = Data is 12 bits. 01 = Data is 8 bits. Others = reserved.
[13:12]	**ETRGSEL**	**External Pin Trigger Selection** 00 = Low level trigger. 01 = High level trigger. 10 = Falling edge trigger. 11 = Rising edge trigger.
[11]	**Reserved**	Reserved.
[10]	**LALIGN**	**DAC Data Left-aligned Enable Control** 0 = Right alignment. 1 = Left alignment.
[9]	**Reserved**	Reserved.
[8]	**BYPASS**	**Bypass Buffer Mode** 0 = Output voltage buffer Enabled. 1 = Output voltage buffer Disabled.
[7:5]	**TRGSEL**	**Trigger Source Selection** 000 = Software trigger. 001 = External pin DAC1_ST trigger. 010 = Timer 0 trigger. 011 = Timer 1 trigger. 100 = Timer 2 trigger. 101 = Timer 3 trigger. 110 = EPWM0 trigger.

		111 = EPWM1 trigger.
[4]	**TRGEN**	**Trigger Mode Enable Bit** 0 = DAC event trigger mode Disabled. 1 = DAC event trigger mode Enabled.
[3]	**DMAURIEN**	**DMA Under-run Interrupt Enable Bit** 0 = DMA under-run interrupt Disabled. 1 = DMA under-run interrupt Enabled.
[2]	**DMAEN**	**DMA Mode Enable Bit** 0 = DMA mode Disabled. 1 = DMA mode Enabled.
[1]	**DACIEN**	**DAC Interrupt Enable Bit** 0 = DAC interrupt Disabled. 1 = DAC interrupt Enabled.
[0]	**DACEN**	**DAC Enable Bit** 0 = DAC Disabled. 1 = DAC Enabled.

DAC1 Software Trigger Control Register (DAC1_SWTRG)

Register	Offset	R/W	Description	Reset Value
DAC1_SWTRG	DAC_BA+0x44	R/W	DAC1 Software Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**SWTRG**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SWTRG**	**Software Trigger** 0 = Software trigger Disabled. 1 = Software trigger Enabled. **Note:** User writes this bit to generate one shot pulse and it is cleared to 0 by hardware automatically; Reading this bit will always get 0.

DAC1 Data Holding Register (DAC1_DAT)

Register	Offset	R/W	Description	Reset Value
DAC1_DAT	DAC_BA+0x48	R/W	DAC1 Data Holding Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DACDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DACDAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DACDAT**	**DAC 12-bit Holding Data** These bits are written by user software which specifies 12-bit conversion data for DAC output. The unused bits (DAC1_DAT[3:0] in left-alignment mode and DAC1_DAT[15:12] in right alignment mode) are ignored by DAC controller hardware. 12 bit left alignment: user has to load data into DAC1_DAT[15:4] bits. 12 bit right alignment: user has to load data into DAC1_DAT[11:0] bits.

DAC1 Data Output Register (DAC1_DATOUT)

Register	Offset	R/W	Description	Reset Value
DAC1_DATOUT	DAC_BA+0x4C	R	DAC1 Data Output Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				DATOUT			
7	6	5	4	3	2	1	0
DATOUT							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**DATOUT**	**DAC 12-bit Output Data** These bits are current digital data for DAC output conversion. It is loaded from DAC1_DAT register and user cannot write it directly.

DAC1 Status Register (DAC1_STATUS)

Register	Offset	R/W	Description	Reset Value
DAC1_STATUS	DAC_BA+0x50	R/W	DAC1 Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							BUSY
7	6	5	4	3	2	1	0
Reserved						DMAUDR	FINISH

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BUSY**	**DAC Busy Flag (Read Only)** 0 = DAC is ready for next conversion. 1 = DAC is busy in conversion.
[7:2]	**Reserved**	Reserved.
[1]	**DMAUDR**	**DMA Under-run Interrupt Flag** 0 = No DMA under-run error condition occurred. 1 = DMA under-run error condition occurred. **Note:** User writes 1 to clear this bit.
[0]	**FINISH**	**DAC Conversion Complete Finish Flag** 0 = DAC is in conversion state. 1 = DAC conversion finished. **Note:** This bit set to 1 when conversion time counter counts to SETTLET. It is cleared to 0 when DAC starts a new conversion. User writes 1 to clear this bit to 0.

DAC1 Timing Control Register (DAC1_TCTL)

Register	Offset	R/W	Description	Reset Value
DAC1_TCTL	DAC_BA+0x54	R/W	DAC1 Timing Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**SETTLET**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETTLET							

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**SETTLET**	**DAC Output Settling Time** User software needs to write appropriate value to these bits to meet DAC conversion settling time base on PCLK (APB clock) speed. For example, DAC controller clock speed is 80 MHz and DAC conversion settling time is 1 us, SETTLET value must be greater than 0x50. SELTTLET = DAC controller clock speed x setting time.

6.46 Analog Comparator Controller (ACMP)

6.46.1 Overview

The chip provides two ACMP controllers, each controller supports 2 comparators. Controller ACMP01 supports ACMP0 and ACMP1; controller ACMP23 supports ACMP2 and ACMP3. The comparator output is logic 1 when positive input is greater than negative input; otherwise, the output is 0. Each comparator can be configured to generate an interrupt when the comparator output value changes.

6.46.2 Features

- Analog input voltage range: 0 ~ AV_{DD} (voltage of AV_{DD} pin)
- Up to four rail-to-rail analog comparators
- Supports hysteresis function
  - Supports programmable hysteresis window: 0mV, 20mV, 40mV
- Supports wake-up function
- Selectable input sources of positive input and negative input
- ACMP0 supports:
  - 4 multiplexed I/O pins at positive sources:
    - ACMP0_P0, ACMP0_P1, ACMP0_P2, or ACMP0_P3
  - 4 negative sources:
    - ACMP0_N
    - Supports Comparator Reference Voltage (CRV0)
    - Internal band-gap voltage (V_{BG})
    - DAC0 output (DAC0_OUT)
- ACMP1 supports:
  - 4 multiplexed I/O pins at positive sources:
    - ACMP1_P0, ACMP1_P1, ACMP1_P2, or ACMP1_P3
  - 4 negative sources:
    - ACMP1_N
    - Supports Comparator Reference Voltage (CRV1)
    - Internal band-gap voltage (V_{BG})
    - DAC0 output (DAC0_OUT)
- ACMP2 supports:
  - 4 multiplexed I/O pins at positive sources:
    - ACMP2_P0, ACMP2_P1, ACMP2_P2, or ACMP2_P3
  - 4 negative sources:
    - ACMP2_N
    - Supports Comparator Reference Voltage (CRV2)
    - Internal band-gap voltage (V_{BG})
    - DAC1 output (DAC1_OUT)

- ACMP3 supports:
  - 4 multiplexed I/O pins at positive sources:
    - ACMP3_P0, ACMP3_P1, ACMP3_P2, or ACMP3_P3
  - 4 negative sources:
    - ACMP3_N
    - Supports Comparator Reference Voltage (CRV3)
    - Internal band-gap voltage (V_{BG})
    - DAC1 output (DAC1_OUT)
- Shares one ACMP interrupt vector for two comparators, ACMP01_INT is for ACMP0 and ACMP1, ACMP23_INT is for ACMP2 and ACMP3
- Interrupts generated when compare results change (Interrupt event condition is programmable)
- Supports triggers for break events and cycle-by-cycle control for EPWM
- Supports window compare mode and window latch mode
- Supports offset calibration

6.46.3 Block Diagram

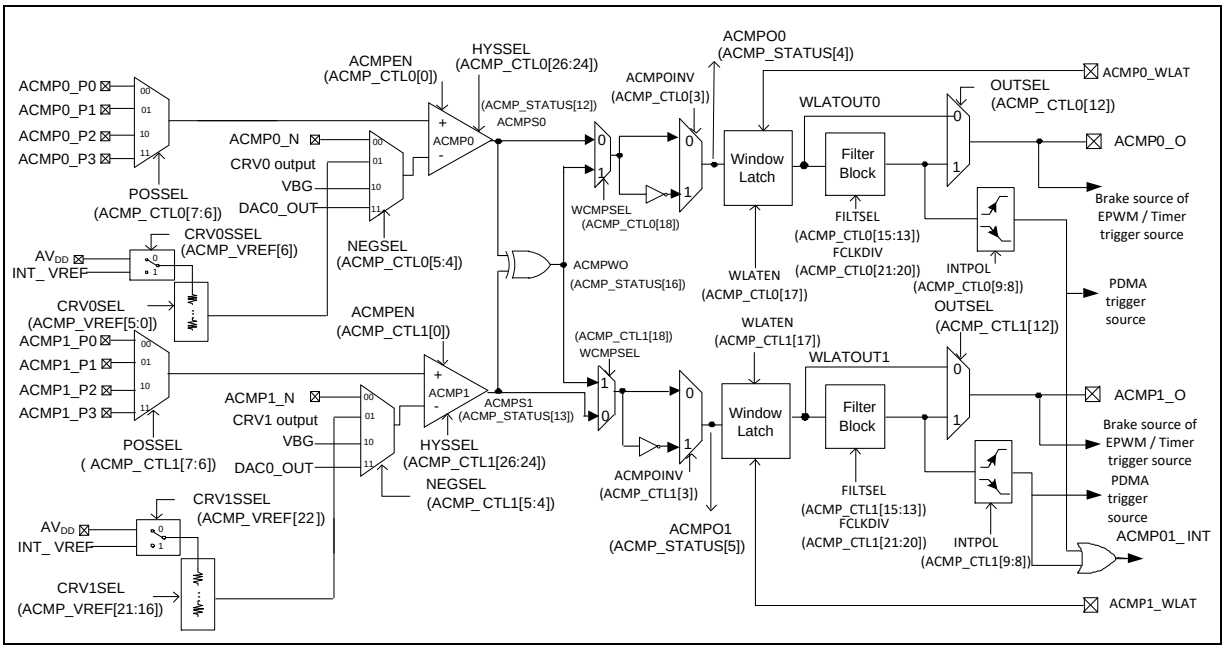


Figure 6.46-1 ACMP01 Block Diagram

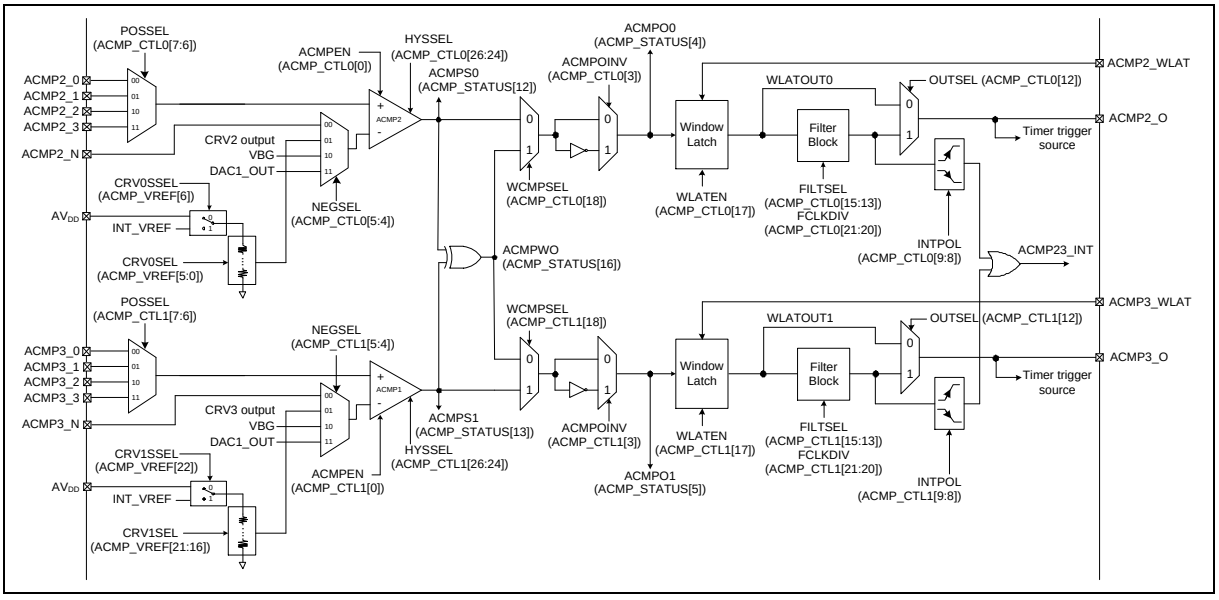


Figure 6.46-2 ACMP23 Block Diagram

6.46.4 Basic Configuration

6.46.4.1 ACMP0 Basic Configuration

- Clock Source Configuration
  - Enable ACMP0 peripheral clock in ACMP01CKEN (CLK_APBCLK0[7]).
- Reset Configuration
  - Reset ACMP0 controller in ACMP01RST (SYS_IPRST1[7]).

6.46.4.2 ACMP1 Basic Configuration

- Clock Source Configuration
  - Enable ACMP1 peripheral clock in ACMP01CKEN (CLK_APBCLK0[7]).
- Reset Configuration
  - Reset ACMP1 controller in ACMP01RST (SYS_IPRST1[7]).

6.46.4.3 ACMP2 Basic Configuration

- Clock Source Configuration
  - Enable ACMP2 peripheral clock in ACMP23CKEN (CLK_APBCLK2[7]).
- Reset Configuration
  - Reset ACMP2 controller in ACMP23RST (SYS_IPRST3[7]).

6.46.4.4 ACMP3 Basic Configuration

- Clock Source Configuration
  - Enable ACMP3 peripheral clock in ACMP23CKEN (CLK_APBCLK2[7]).
- Reset Configuration
  - Reset ACMP3 controller in ACMP23RST (SYS_IPRST3[7]).

6.46.5 Functional Description

6.46.5.1 Hysteresis Function

The analog comparator provides the hysteresis function to make the comparator to have a stable output transition and it can refer to Figure 6.46-3. If comparator output is 0, it will not be changed to 1 until the positive input voltage exceeds the negative input voltage by a high threshold voltage. Similarly, if comparator output is 1, it will not be changed to 0 until the positive input voltage drops below the negative input voltage by a low threshold voltage. User can determine the threshold voltage by setting HYSSEL (ACMP_CTL0/1[26:24]).

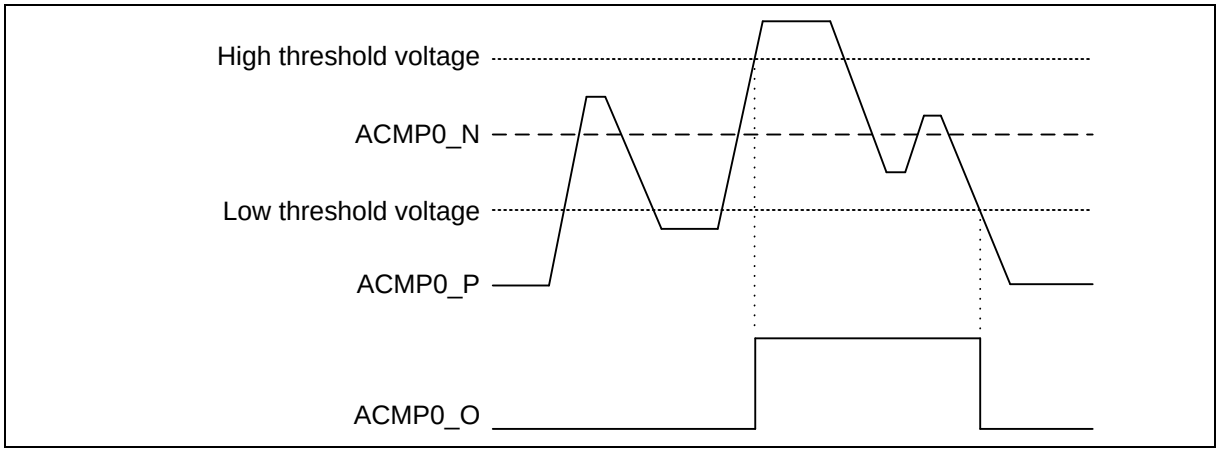


Figure 6.46-3 Comparator Hysteresis Function of ACMP0

6.46.5.2 Window Latch Mode

Figure 6.46-4 shows the example of ACMP01 comparator operation in window latch mode. Window latch mode can be enabled by setting WLATEN (ACMP_CTL0/1[17]) to 1. When window latch function enabled, ACMP0/1_WLAT pin is used to control the output WLATOUT0/1. When ACMP0/1_WLAT pin

is high, ACMPO0/1 passes through to WLATOUT0/1. When ACMP0/1_WLAT pin is low, WLATOUT0/1 will keep last state of WLATOUT0/1. It is noted that ACMP2/3 window latch mode is the same, and which WLATOUT2/3 are decided by ACMP2/3_WLAT pin.

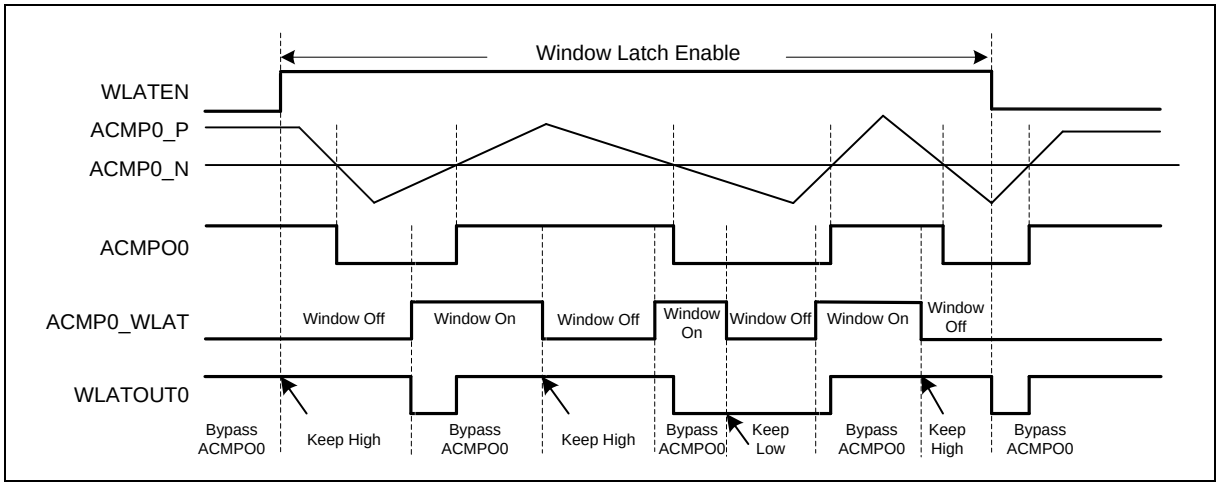


Figure 6.46-4 Window Latch Mode

6.46.5.3 Filter Function

The analog comparator provides filter function to avoid the un-stable state of comparator output. User can set the OUTSEL (ACMP_CTL0/1[12]) to enable filter function.

By setting FILTSEL (ACMP_CTL0/1[15:13]), the comparator output would be sampled by consecutive PCLKs. With longer sample clocks, the comparator output would be more stable. But the sensitivity of comparator output would be reduced. Under the condition of setting FILTSEL to 3'h7 the comparator output is sampled 64 consecutive PCLKs, filter sample clock can be divided by 2 or 4 by setting FCLKDIV(ACMP_CTL0/1[21:20]) to get sampled 128, 256 consecutive PCLKs.

Figure 6.46-5 shows an example of filter function of ACMP0 with FILTSEL = 3 (4 PCLK). In this example, the comparing result is sampled by PCLK. All result must keep for 4 PCLK clocks before it can be output to ACMPO0. If the comparing result is shorter than 4 PCLK, it will be filtered.

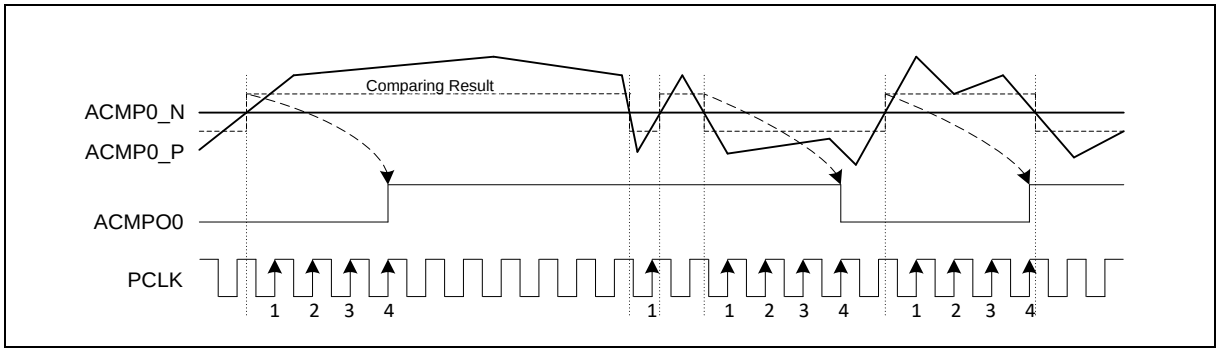


Figure 6.46-5 Example of Filter Function

6.46.5.4 Interrupt Sources

The outputs of ACMP0 and ACMP1 are reflected at ACMPO0 (ACMP_STATUS[4]) and ACMPO1 (ACMP_STATUS[5]) respectively in ACMP01. And the outputs of ACMP2 and ACMP3 are reflected at ACMPO0 (ACMP_STATUS[4]) and ACMPO1 (ACMP_STATUS[5]) respectively in ACMP23. User can

set ACMPOINV (ACMP_CTL0/1[3]) to enable the inverse function for the output result.

Processed by window latch and filter functions, additionally the output signal could be utilized to assert interrupts. Refer to Figure 6.46-6 and Figure 6.46-7, if ACMPIE (ACMP_CTL0/1[1]) register is set to 1, the interrupt will be enabled. If the output state ACMPO0/1 is changed as the setting of INTPOL (ACMP_CTL0/1[9:8]), the comparator interrupt will be asserted and the corresponding flag, ACMPIF0 (ACMP_STATUS[0]), ACMPIF1 (ACMP_STATUS[1]) will be set to 1. The interrupt flag can be cleared to 0 by writing 1.

For ACMP01 and ACMP23, both of them have two WKIF flags. WKIF0(ACMP_STATUS[8]) and WKIF1(ACMP_STATUS[9]) will be set according to the setting of INTPOL (ACMP_CTL0/1[9:8]) if ACMP wakeup function is enabled. User can set WKEN (ACMP_CTL0/1[16]) to enable wakeup function. WKIF flag also causes interrupt rising that makes system wake-up from power down by ACMP. Figure 6.46-6 and Figure 6.46-7 shows the interrupt sources of ACMP.

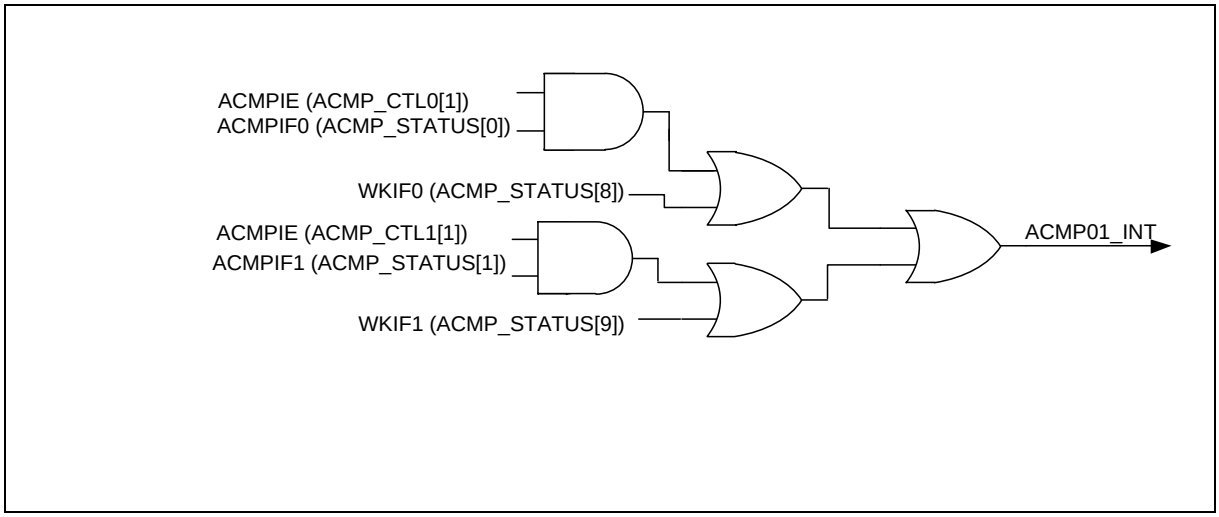


Figure 6.46-6 ACMP01 Interrupt Diagram

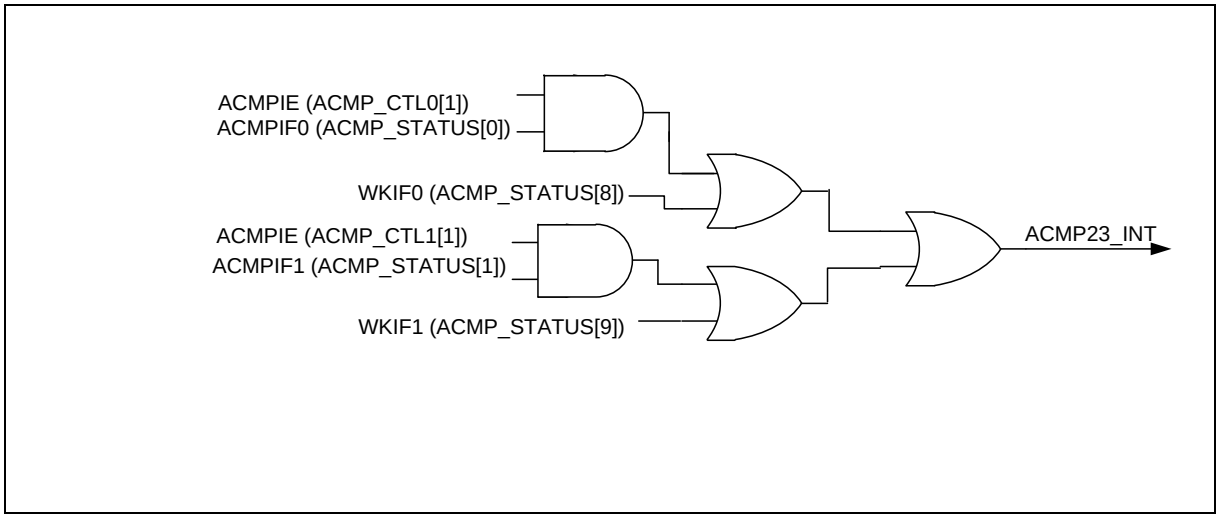


Figure 6.46-7 ACMP23 Interrupt Diagram

6.46.5.5 Comparator Reference Voltage (CRV)

The comparator reference voltage (CRV) module is responsible for generating reference voltage for comparators. The CRV module consists of resistor ladder and analog switch. User can set the CRV output voltage by setting CRV0SEL (ACMP_VREF[5:0]), CRV1SEL (ACMP_VREF[21:16]), The CRV output voltage can be selected as the negative input of comparator by setting NEGSEL (ACMP_CTL0[5:4], ACMP_CTL1[5:4]). Figure 6.46-8 shows the block diagram of Comparator Reference Voltage.

The resistor ladder will be disabled by hardware to reduce power consumption when NEGSEL (ACMP_CTL0/1[5:4]) is not selected to CRV module. The reference voltage of resistor ladder can be the voltage of AV_{DD} pin or the INT_VREF voltage which is controlled by VREFCTL (SYS_VREFCTL[4:0]) register. Figure 6.46-8 shows comparator0 CRV0 output. For example, if user wants to get comparator1 CRV1 output, CRV1SSEL (ACMP_VREF[22]), CRV1SEL (ACMP_VREF[21:16]) must be set, so as the generation of CRV2, CRV3 output.

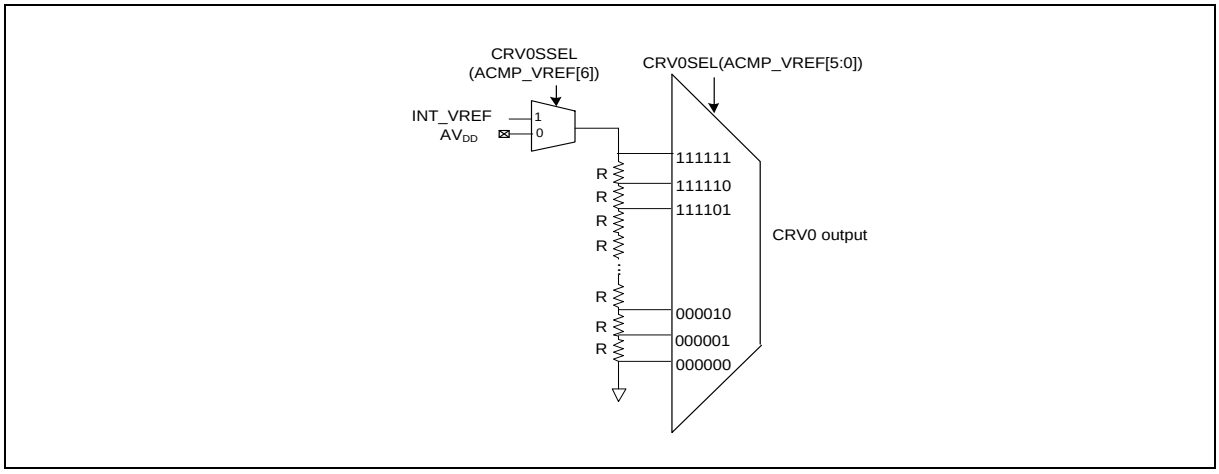


Figure 6.46-8 Comparator0 Reference Voltage Block Diagram

6.46.5.6 Window Compare Mode

The chip with four comparators provide 2 sets of window compare mode. It is composed of ACMP0/1, the other one is composed of ACMP2/3. Take ACMP0/1 window compare mode for example. When window compare mode is enabled by setting WCMPSEL (ACMP_CTL0/1[18]) to 1, user can monitor a specific analog voltage source with a designated range. User can connect the specific analog voltage source to either the positive inputs of both comparators or the negative inputs of both comparators. The upper bound and lower bound of the designated range are determined by the voltages applied to the other inputs of both comparators. If the output of a comparator is low and the other comparator outputs high, which means two comparators imply the upper and lower bound. User can directly monitor a specific analog voltage source via ACMPWO (ACMP_STATUS[16]). If ACMPWO is high, it implies a specific analog voltage source is in the range of upper and lower bound, which are called as the analog voltage is in the window.

Figure 6.46-9 illustrates an example of window compare mode. In this example, once window compare mode is selected, user can choose one of four positive input sources of each comparator and connect these two inputs together outside the chip.

If ACMPS0 outputs high and ACMPS1 outputs low, it means the voltage source is in the range of lower bound and upper bound, which are called as the voltage source in the window. Otherwise, the voltage source is outside the window.

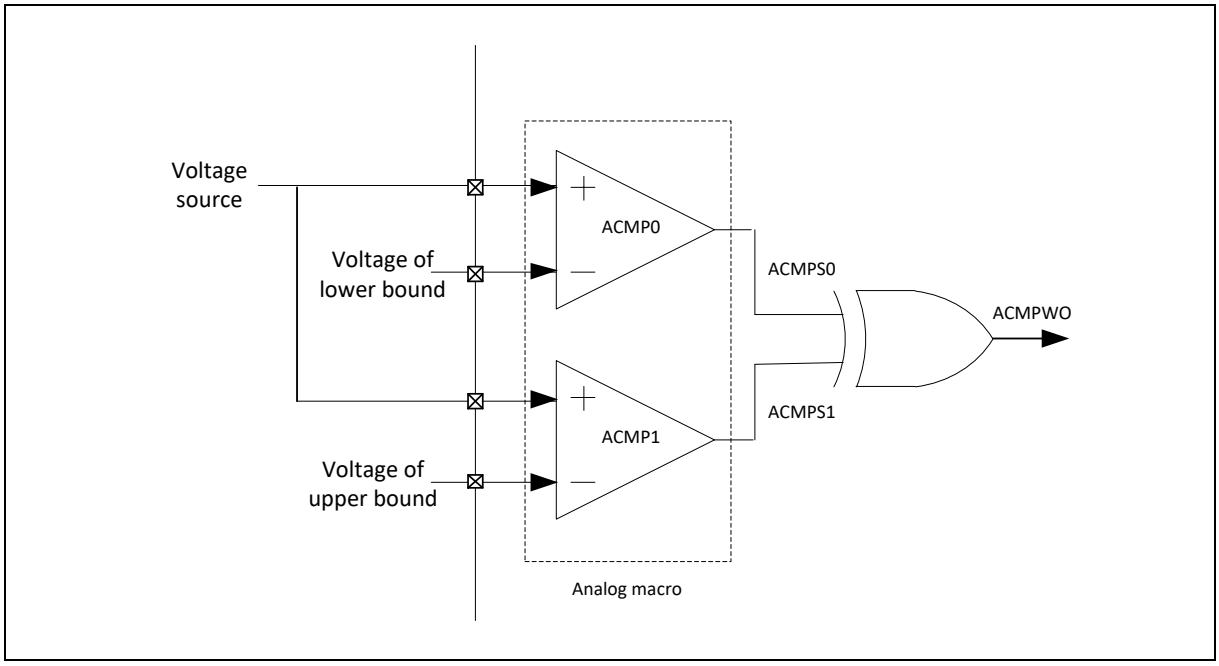


Figure 6.46-9 Example of Window Compare Mode

The comparator window output (ACMPWO) can be shown in ACMP_STATUS[16] and the truth table of window compare logic is shown in Table 6.46-1.

ACMPS0	ACMPS1	ACMPWO
0	0	0
0	1	1
1	1	0
1	0	1

Table 6.46-1 Truth Table of Window Compare Logic

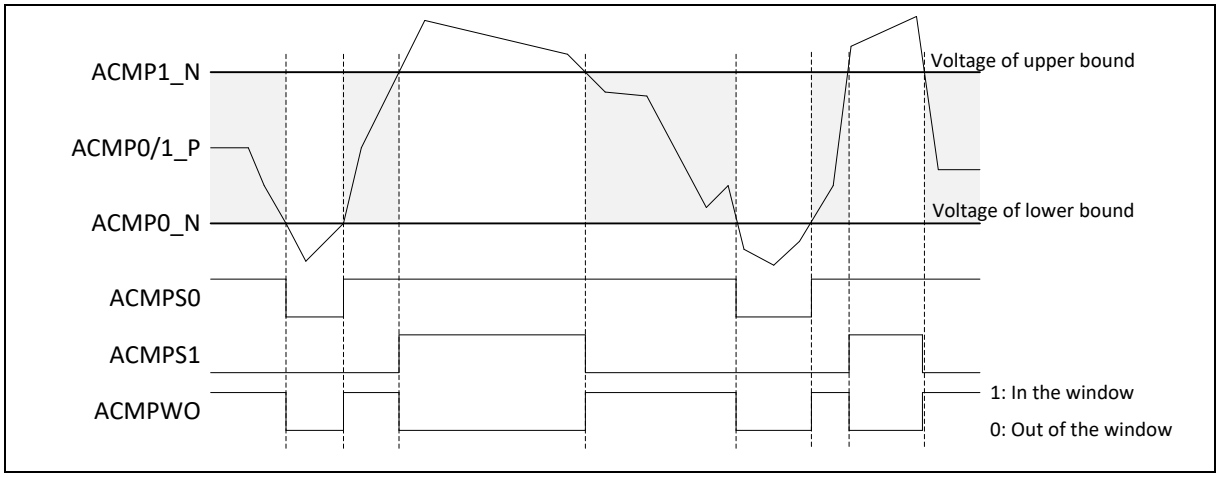


Figure 6.46-10 Example of Window Compare Mode

As shown in Figure 6.46-10, if ACMPWO equals 1, it means positive input voltage is inside the window. Otherwise, the positive input voltage is outside the window. Therefore, ACMPWO can be used to monitor

voltage transition of external analog pin. Furthermore, ACMPWO still can be applied to window latch, filter functions and interrupt of ACMP.

Note that negative inputs must choose different source. Otherwise, the function will be meaningless.

6.46.5.7 Calibration Function

The four comparators have its own four trim bits which can be used to calibrate the offset voltage. The offset voltage comes from both mismatch of NMOS-type differential and PMOS-type differential input stages. Calibration can be started by setting CALTRG0 (ACMP_CALCTL[0]), CALTRG1 (ACMP_CALCTL[1]). The flags of DONE0 (ACMP_CALSTS[0]) and DONE1 (ACMP_CALSTS[4]) reveal calibration status.

The common mode input range is achieved by using an NMOS and a PMOS differential pairs connected in parallel. After calibration, reading the ACMP_CALSTS register can monitor calibration status, including calibration done status, NMOS, and PMOS calibration result status. Take a brief calibration flow for example: Once ACMPEN (ACMP_CTL0/1[0]) is enabled, hardware will automatically load default calibrated trim values to compensate offset voltage. Therefore, user can directly use comparator without doing additional calibration action. If user wants to start calibration function again, the newer calibrated trim value will be updated after calibration done. Noted that every time ACMPEN is set, user must set CALTRG0 (ACMP_CALCTL[0]) or CALTRG1 (ACMP_CALCTL[1]) and NEGSEL (ACMP_CTL0/1[5:4]) equal to 2'b01 to start calibration function getting newer offset trim values, or comparator will operating with elder offset trim values.

6.46.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write

Register	Offset	R/W	Description	Reset Value
ACMP Base Address: **ACMPx_BA = 0x4004_5000 + 0x8_4000 * x** **x = 0, 1** **x set 0 is for ACMP0 and ACMP1** **x set 1 is for ACMP2 and ACMP3**				
ACMP_CTL0	ACMPx_BA+0x00	R/W	Analog Comparator Control Register 0	0x1000_0000
ACMP_CTL1	ACMPx_BA+0x04	R/W	Analog Comparator Control Register 1	0x1000_0000
ACMP_STATUS	ACMPx_BA+0x08	R/W	Analog Comparator Status Register	0x0000_0000
ACMP_VREF	ACMPx_BA+0x0C	R/W	Analog Comparator Reference Voltage Control Register	0x0000_0000
ACMP_CALCTL	ACMPx_BA+0x10	R/W	Analog Comparator Calibration Control Register	0x0000_0050
ACMP_CALSTS	ACMPx_BA+0x14	R	Analog Comparator Calibration Status Register	0x0000_0000

6.46.7 Register Description

Analog Comparator Control Register 0 (ACMP_CTL0)

Register	Offset	R/W	Description	Reset Value
ACMP_CTL0	ACMP01_BA+0x00	R/W	Analog Comparator 0 Control Register	0x1000_0000

31	30	29	28	27	26	25	24
Reserved		MODESEL		Reserved	HYSSEL		
23	22	21	20	19	18	17	16
Reserved		FCLKDIV		Reserved	WCMPSEL	WLATEN	WKEN
15	14	13	12	11	10	9	8
FILTSEL			OUTSEL	Reserved		INTPOL	
7	6	5	4	3	2	1	0
POSSEL		NEGSEL		ACMPOINV	Reserved	ACMPIE	ACMPEN

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**MODESEL**	**ACMP Operating Mode Selection** The register affects analog characteristics as operating current, input offset voltage, setup time, propagation delay, etc. 00 = ACMP0 or ACMP2 operating mode0. 01 = ACMP0 or ACMP2 operating mode1. 10 = ACMP0 or ACMP2 operating mode2. 11 = ACMP0 or ACMP2 operating mode3. **Note:** Please refer to Analog comparator (ACMP) of Analog characteristics in the Datasheet.
[27]	**Reserved**	Reserved.
[26:24]	**HYSSEL**	**Hysteresis Mode Selection** 000 = Hysteresis is 0mV. 010 = Hysteresis is 20mV. 100 = Hysteresis is 40mV. Others = Reserved.
[23:22]	**Reserved**	Reserved.
[21:20]	**FCLKDIV**	**ACMP Output Filter Clock Divider** 00 = ACMP0 or ACMP2 output filter clock = PCLK. 01 = ACMP0 or ACMP2 output filter clock = PCLK/2. 10 = ACMP0 or ACMP2 output filter clock = PCLK/4. 11 = Reserved. **Note:** Use FCLKDIV under the condition of FILTSEL = 0x7, then set FCLKDIV to get the effect of filtering 128,256 consecutive PCLKs.
[19]	**Reserved**	Reserved.
[18]	**WCMPSEL**	**Window Compare Mode Selection** 0 = Window Compare Mode Disabled.

		1 = Window Compare Mode is Selected.
[17]	**WLATEN**	**Window Latch Mode Enable Bit** 0 = Window Latch Mode Disabled. 1 = Window Latch Mode Enabled.
[16]	**WKEN**	**Power-down Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.
[15:13]	**FILTSEL**	**ACMP Output Filter Count Selection** 000 = Filter function is Disabled. 001 = ACMP0 or ACMP2 output is sampled 1 consecutive PCLK. 010 = ACMP0 or ACMP2 output is sampled 2 consecutive PCLKs. 011 = ACMP0 or ACMP2 output is sampled 4 consecutive PCLKs. 100 = ACMP0 or ACMP2 output is sampled 8 consecutive PCLKs. 101 = ACMP0 or ACMP2 output is sampled 16 consecutive PCLKs. 110 = ACMP0 or ACMP2 output is sampled 32 consecutive PCLKs. 111 = ACMP0 or ACMP2 output is sampled 64 consecutive PCLKs.
[12]	**OUTSEL**	**ACMP Output Selection** 0 = ACMP0 or ACMP2 output to ACMP0_O or ACMP2_O pin is unfiltered ACMP output. 1 = ACMP0 or ACMP2 output to ACMP0_O or ACMP2_O pin is from filter output.
[11:10]	**Reserved**	Reserved.
[9:8]	**INTPOL**	**Interrupt Condition Polarity Selection** ACMPIF0 will be set to 1 when ACMP output edge condition is detected. 00 = Rising edge or falling edge. 01 = Rising edge. 10 = Falling edge. 11 = Reserved.
[7:6]	**POSSEL**	**ACMP Positive Input Selection** 00 = Input from ACMP0_P0 or ACMP2_P0. 01 = Input from ACMP0_P1 or ACMP2_P1. 10 = Input from ACMP0_P2 or ACMP2_P2. 11 = Input from ACMP0_P3 or ACMP2_P3.
[5:4]	**NEGSEL**	**ACMP Negative Input Selection** 00 = ACMP0_N or ACMP2_N pin. 01 = Internal comparator reference voltage (CRV0/2). 10 = Band-gap voltage. 11 = DAC0 output (ACMP0) or DAC1 output (ACMP1). **Note:** NEGSEL must select 2'b01 in calibration mode.
[3]	**ACMPOINV**	**ACMP Output Inverse** 0 = ACMP0 or ACMP2 output inverse Disabled. 1 = ACMP0 or ACMP2 output inverse Enabled.
[2]	**Reserved**	Reserved.
[1]	**ACMPIE**	**ACMP Interrupt Enable Bit** 0 = ACMP0 or ACMP2 interrupt Disabled. 1 = ACMP0 or ACMP2 interrupt Enabled. **Note:**If WKEN (ACMP_CTL0[16]) is set to 1, the wake-up interrupt function will be enabled as well.

[0]	**ACMPEN**	**ACMP Enable Bit** 0 = ACMP0 or ACMP2 Disabled. 1 = ACMP0 or ACMP2 Enabled. **Note:** There is a setup time T_{Setup} after ACMP is enabled or MODESEL is changed. User should wait this T_{Setup} to make ACMP working normally, or user may get unstable ACMP output value.

Analog Comparator Control Register 1 (ACMP_CTL1)

Register	Offset	R/W	Description	Reset Value
ACMP_CTL1	ACMP01_BA+0x04	R/W	Analog Comparator 1 Control Register	0x1000_0000

31	30	29	28	27	26	25	24
Reserved		MODESEL		Reserved	HYSSEL		
23	22	21	20	19	18	17	16
Reserved		FCLKDIV		Reserved	WCMPSEL	WLATEN	WKEN
15	14	13	12	11	10	9	8
FILTSEL			OUTSEL	Reserved		INTPOL	
7	6	5	4	3	2	1	0
POSSEL		NEGSEL		ACMPOINV	Reserved	ACMPIE	ACMPEN

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**MODESEL**	**ACMP Operating Mode Selection** The register affects analog characteristics as operating current, input offset voltage, setup time, propagation delay, etc. 00 = ACMP1 or ACMP3 operating mode0. 01 = ACMP1 or ACMP3 operating mode1. 10 = ACMP1 or ACMP3 operating mode2. 11 = ACMP1 or ACMP3 operating mode3. **Note:** Please refer to Analog comparator (ACMP) of Analog characteristics in the Datasheet.
[27]	**Reserved**	Reserved.
[26:24]	**HYSSEL**	**Hysteresis Mode Selection** 000 = Hysteresis is 0mV. 010 = Hysteresis is 20mV. 100 = Hysteresis is 40mV. Others = Reserved.
[23:22]	**Reserved**	Reserved.
[21:20]	**FCLKDIV**	**ACMP Output Filter Clock Divider** 00 = ACMP1 or ACMP3 output filter clock = PCLK. 01 = ACMP1 or ACMP3 output filter clock = PCLK/2. 10 = ACMP1 or ACMP3 output filter clock = PCLK/4. 11 = Reserved. **Note:** Use FCLKDIV under the condition of FILTSEL = 3'h7, then set FCLKDIV to get the effect of filtering 128,256 consecutive PCLKs.
[19]	**Reserved**	Reserved.
[18]	**WCMPSEL**	**Window Compare Mode Selection** 0 = Window Compare Mode Disabled. 1 = Window Compare Mode is Selected.

Bits	Description	
[17]	**WLATEN**	**Window Latch Mode Enable Bit** 0 = Window Latch Mode Disabled. 1 = Window Latch Mode Enabled.
[16]	**WKEN**	**Power-down Wakeup Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.
[15:13]	**FILTSEL**	**ACMP Output Filter Count Selection** 000 = Filter function is Disabled. 001 = ACMP1 or ACMP3 output is sampled 1 consecutive PCLK. 010 = ACMP1 or ACMP3 output is sampled 2 consecutive PCLKs. 011 = ACMP1 or ACMP3 output is sampled 4 consecutive PCLKs. 100 = ACMP1 or ACMP3 output is sampled 8 consecutive PCLKs. 101 = ACMP1 or ACMP3 output is sampled 16 consecutive PCLKs. 110 = ACMP1 or ACMP3 output is sampled 32 consecutive PCLKs. 111 = ACMP1 or ACMP3 output is sampled 64 consecutive PCLKs.
[12]	**OUTSEL**	**ACMP Output Selection** 0 = ACMP1 or ACMP3 output to ACMP1_O or ACMP3_O pin is unfiltered ACMP output. 1 = ACMP1 or ACMP3 output to ACMP1_O or ACMP3_O pin is from filter output.
[11:10]	**Reserved**	Reserved.
[9:8]	**INTPOL**	**Interrupt Condition Polarity Selection** ACMPIF1 will be set to 1 when ACMP output edge condition is detected. 00 = Rising edge or falling edge. 01 = Rising edge. 10 = Falling edge. 11 = Reserved.
[7:6]	**POSSEL**	**ACMP Positive Input Selection** 00 = Input from ACMP1_P0 or ACMP3_P0. 01 = Input from ACMP1_P1 or ACMP3_P1. 10 = Input from ACMP1_P2 or ACMP3_P2. 11 = Input from ACMP1_P3 or ACMP3_P3.
[5:4]	**NEGSEL**	**ACMP Negative Input Selection** 00 = ACMP1_N or ACMP3_N pin. 01 = Internal comparator reference voltage (CRV1/3). 10 = Band-gap voltage. 11 = DAC0 output (ACMP1) or DAC1 output (ACMP3). **Note:** NEGSEL must select 2'b01 in calibration mode.
[3]	**ACMPOINV**	**ACMP Output Inverse** 0 = ACMP1 or ACMP3 output inverse Disabled. 1 = ACMP1 or ACMP3 output inverse Enabled.
[1]	**ACMPIE**	**ACMP Interrupt Enable Bit** 0 = ACMP1 or ACMP3 interrupt Disabled. 1 = ACMP1 or ACMP3 interrupt Enabled. **Note:**If WKEN (ACMP_CTL1[16]) is set to 1, the wake-up interrupt function will be enabled as well.
[0]	**ACMPEN**	**ACMP Enable Bit**

Bits	Description	
		0 = ACMP1 or ACMP3 Disabled. 1 = ACMP1 or ACMP3 Enabled. **Note:** There is a setup time T_{Setup} after ACMP is enabled or MODESEL is changed. User should wait this T_{Setup} to make ACMP working normally, or user may get unstable ACMP output value.

Analog Comparator Status Register (ACMP_STATUS)

Register	Offset	R/W	Description	Reset Value
ACMP_STATUS	ACMP01_BA+0x08	R/W	Analog Comparator Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							ACMPWO
15	14	13	12	11	10	9	8
Reserved		ACMPS1	ACMPS0	Reserved		WKIF1	WKIF0
7	6	5	4	3	2	1	0
Reserved		ACMPO1	ACMPO0	Reserved		ACMPIF1	ACMPIF0

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**ACMPWO**	**ACMP Window Output** This bit shows the output status of window compare mode 0 = The positive input voltage is outside the window. 1 = The positive input voltage is in the window.
[15:14]	**Reserved**	Reserved.
[13]	**ACMPS1**	**ACMP Status 1** Synchronized to the PCLK to allow reading by software. Cleared when the ACMP1 or ACMP3 is disabled, i.e. ACMPEN (ACMP_CTL1[0]) is cleared to 0.
[12]	**ACMPS0**	**ACMP Status 0** Synchronized to the PCLK to allow reading by software. Cleared when the ACMP0 or ACMP2 is disabled, i.e. ACMPEN (ACMP_CTL0[0]) is cleared to 0.
[11:10]	**Reserved**	Reserved.
[9]	**WKIF1**	**ACMP Power-down Wake-up Interrupt Flag 1** This bit will be set to 1 when ACMP1 or ACMP3 wake-up interrupt event occurs. 0 = No power-down wake-up occurred. 1 = Power-down wake-up occurred. **Note:** Write 1 to clear this bit to 0.
[8]	**WKIF0**	**ACMP Power-down Wake-up Interrupt Flag 0** This bit will be set to 1 when ACMP0 or ACMP2 wake-up interrupt event occurs. 0 = No power-down wake-up occurred. 1 = Power-down wake-up occurred. **Note:** Write 1 to clear this bit to 0.
[7:6]	**Reserved**	Reserved.
[5]	**ACMPO1**	**ACMP Output 1** Synchronized to the PCLK to allow reading by software. Cleared when the ACMP1 or ACMP3 is disabled, i.e. ACMPEN (ACMP_CTL1[0]) is cleared to 0.

Bits	Description	
[4]	**ACMPO0**	**ACMP Output 0** Synchronized to the PCLK to allow reading by software. Cleared when the ACMP0 or ACMP2 is disabled, i.e. ACMPEN (ACMP_CTL0[0]) is cleared to 0.
[3:2]	**Reserved**	Reserved.
[1]	**ACMPIF1**	**ACMP Interrupt Flag 1** This bit is set by hardware when the edge condition defined by INTPOL (ACMP_CTL1[9:8]) is detected on ACMP1 or ACMP3 output. This will cause an interrupt if ACMPIE (ACMP_CTL1[1]) is set to 1. **Note:** Write 1 to clear this bit to 0.
[0]	**ACMPIF0**	**ACMP Interrupt Flag 0** This bit is set by hardware when the edge condition defined by INTPOL (ACMP_CTL0[9:8]) is detected on ACMP0 or ACMP2 output. This will generate an interrupt if ACMPIE (ACMP_CTL0[1]) is set to 1. **Note:** Write 1 to clear this bit to 0.

ACMP Reference Voltage Control Register (ACMP_VREF)

Register	Offset	R/W	Description	Reset Value
ACMP_VREF	ACMP01_BA+0x0C	R/W	Analog Comparator Reference Voltage Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CLAMPEN	Reserved						CRV1EN
23	22	21	20	19	18	17	16
Reserved	CRV1SSEL	CRV1SEL					
15	14	13	12	11	10	9	8
Reserved							CRV0EN
7	6	5	4	3	2	1	0
Reserved	CRV0SSEL	CRV0SEL					

Bits	Description	
[31]	**CLAMPEN**	**ACMP Speed Up Function Enable Bit** 0 = ACMP0 or ACMP1 or ACMP2 or ACMP3 Speed Up Function Disabled. 1 = ACMP0 or ACMP1 or ACMP2 or ACMP3 Speed Up Function Enabled. **Note:** ACMP speed up function only support MODESEL(ACMP_CTL0 [29:28])=2b'11 & 2b'10.
[30:25]	**Reserved**	Reserved.
[24]	**CRV1EN**	**CRV1 Enable Bit** 0 = CRV1/3 Disabled. 1 = CRV1/3 Enabled.
[23]	**Reserved**	Reserved.
[22]	**CRV1SSEL**	**CRV1 Source Voltage Selection** 0 = AV_{DD} is selected as CRV1/3 source voltage. 1 = The reference voltage defined by VREFCTL (SYS_VREFCTL[4:0]) register is selected as CRV1/3 source voltage.
[21:16]	**CRV1SEL**	**ACMP Reference Voltage Setting 1** Set CRV output voltage for ACMP1 or ACMP3 negtive input. CRV1/3 = CRV1/3 source voltage * (CRV1SEL)/ 63.
[15:9]	**Reserved**	Reserved.
[8]	**CRV0EN**	**CRV0 Enable Bit** 0 = CRV0/2 Disabled. 1 = CRV0/2 Enabled.
[7]	**Reserved**	Reserved.
[6]	**CRV0SSEL**	**CRV0 Source Voltage Selection** 0 = AV_{DD} is selected as CRV0/2 source voltage. 1 = The reference voltage defined by VREFCTL (SYS_VREFCTL[4:0]) register is selected as CRV0/2 source voltage.
[5:0]	**CRV0SEL**	**ACMP Reference Voltage Setting 0**

Bits	Description	
		Set CRV output voltage for ACMP0 or ACMP2 negtive input. CRV0/2 = CRV0/2 source voltage * (CRV0SEL)/ 63.

Analog Comparator Calibration Control Register (ACMP_CALCTL)

Register	Offset	R/W	Description	Reset Value
ACMP_CALCTL	ACMP01_BA+0x10	R/W	Analog Comparator Calibration Control Register	0x0000_0050

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						CALTRG1	CALTRG0

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**CALTRG1**	**ACMP CalibrationTrigger 1** Set ACMP1 or ACMP3 calibration trigger. 0 = Calibration is stopped. 1 = Calibration is triggered. **Note 1:** Before this bit is enabled, ACMPEN(ACMP_CTL1[0]) should be set and the internal high speed RC oscillator (HIRC) should be enabled in advance. **Note 2:** Hardware will automatically clear this bit when the next calibration is triggered by software. **Note 3:** If user must trigger calibration twice or more times, the second trigger has to wait until the previous calibration is done.
[0]	**CALTRG0**	**ACMP Calibration Trigger 0** Set ACMP0 or ACMP2 calibration trigger. 0 = Calibration is stopped. 1 = Calibration is triggered. **Note 1:** Before this bit is enabled, ACMPEN(ACMP_CTL0[0]) should be set and the internal high speed RC oscillator (HIRC) should be enabled in advance. **Note 2:** Hardware will automatically clear this bit when the next calibration is triggered by software. **Note 3:** If user must trigger calibration twice or more times, the second trigger has to wait until the previous calibration is done.

Analog Comparator Calibration Status Register (ACMP_CALSTS)

Register	Offset	R/W	Description	Reset Value
ACMP_CALSTS	ACMP01_BA+0x14	R	Analog Comparator Calibration Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			DONE1	Reserved			DONE0

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**DONE1**	**ACMP Calibration Done Status 1** This bit will be set when ACMP1 or ACMP3 calibration is done. 0 = Calibrating. 1 = Calibration done. **Note:** Write 1 to clear this bit to 0.
[3:1]	**Reserved**	Reserved.
[0]	**DONE0**	**ACMP Calibration Done Status 0** This bit will be set when ACMP0 or ACMP2 calibration is done. 0 = Calibrating. 1 = Calibration done. **Note:** Write 1 to clear this bit to 0.

6.47 Peripherals Interconnection

6.47.1 Overview

Some peripherals have interconnections which allow autonomous communication or synchronous action between peripherals without needing to involve the CPU. Peripherals interact without CPU saves CPU resources, reduces power consumption, operates with no software latency and fast responds.

6.47.2 Peripherals Interconnect Matrix Table

Source	Destination									
	BPWM	DAC	EADC	ECAP	EPWM	HIRC	IRCTRIM	RC48M	TIMERPWM	QEI
ACMP	-	-	-	-	3	-	-	-	3, 6	
BOD	-	-	-	-	3	-	-	-	3	
BPWM	4	-	1	-	4	-	-	-	-	
Clock Fail	-	-	-	-	3	-	-	-	3	
CPU Lockup	-	-	-	-	3	-	-	-	3	
EADC	-	-	-	-	3	-	-	-	-	
EPWM	4	1	1	-	4	-	-	-	-	
IRCTRIM	-	-	-	-	-	2	-	2	-	
LIRC	-	-	-	-	-	-	-	-	6	
LXT	-	-	-	-	-	-	2	-	-	
QEI	-	-	-	8	-	-	-	-	-	
SRAM	-	-	-	-	3	-	-	-	3	
TIMERPWM	5	1	1	-	5	-	-	-	7	9
USB11Device	-	-	-	-	-	-	2	-	-	

Table 6.47-1 Peripherals Interconnect Matrix table

6.47.3 Functional Description

1. From EPWM, TIMER to EADC/DAC

   - EPWM Trigger EADC Conversion

     EPWM can be one of the EADC conversion trigger source.

     Setting the EADC external hardware trigger input source from EPWM trigger described in section 6.44.5.7.

     The detail EPWM trigger conditions are described in 6.12.5.27.

   - EPWM Trigger DAC Conversion

     EPWM can also be used to trigger DAC conversion.

     Setting the DAC hardware trigger input source from EPWM trigger described in section 6.45.5.6.

     The detail EPWM trigger conditions are described in 6.12.5.27.

- BPWM Trigger EADC Conversion

  BPWM can be one of the EADC conversion trigger source.

  Setting the EADC external hardware trigger input source from BPWM trigger described in section 6.44.5.7.

  The detail BPWM trigger conditions are described in 6.11.5.16(or 6.11.5.17.

- Timer Trigger EADC Conversion

  Timer0 ~ Timer3 can be one of the EADC conversion trigger source. When timer counter value matches the timer compared value or when the TMx_EXT pin edge transition meets setting, timer will trigger the ADC to start the conversion.

  Setting the EADC external hardware trigger input source from timer trigger described in section 6.44.5.8.

  The detail Timer trigger conditions are described in section 6.10.5.10.

- Timer Trigger DAC Conversion

  Setting the DAC hardware trigger input source from TIMER trigger described in section 6.45.5.6.

  The detail Timer trigger conditions are described in section 6.10.5.10.

2. From LXT and USB 1.1 Device to HIRC TRIM & RC 48 MHz

   - Use LXT or USB Synchronous Mode to system auto-trim HIRC circuit

     This chip supports auto-trim function: the HIRC trim (12 MHz RC oscillator) and RC 48 MHz oscillator, according to the accurate external 32.768 kHz crystal oscillator or internal USB synchronous mode, automatically gets accurate output frequency, 0.25 % deviation within all temperature ranges.

     The detail of HIRC trim setting is described in section 6.2.9.

3. From ACMP, BOD, Clock Fail,SRAM Parity Error and CPU Lockup to EPWM/TIMERPWM

   - EPWM Brake Source

     EPWM brake source can be ACMP0/1_O output signal or EADC result monitor or several different system fail conditions include clock fail, Brown-out detect, and Core lockup and SRAM Parity Error. When system fault, EPWM brake signal generated, EPWM output will be set to protect the power switch controlled by EPWM.

     The detail setting of EPWM brake function is described in section 6.12.5.23.

   - TIMERPWM Brake Source

     TIMERPWM brake source can be ACMP0/1_O output signal or several different system fail conditions include clock fail, Brown-out detect, and Core lockup and SRAM Parity Error. When system fault, EPWM brake signal generated, EPWM output will be set to protect the power switch controlled by EPWM.

     The detail setting of TIMERPWM brake function is described in section 6.10.6.17.

4. From EPWM/ BPWM to EPWM/ BPWM

   - EPWM Synchronous Function

     Select synchronous source from EPWM0 or EPWM1 or BPWM0 or BPWM1, and select EPWM channels. The chosen EPWM channels will start counting at the same time once the synchronous start function is enabled and CNTSEN(EPWM_SSTRG[0]) is set.

     The detail setting of EPWM synchronous function is described in section 6.12.5.19.

- BPWM Synchronous Function

  Select synchronous source from EPWM0 or EPWM1 or BPWM0 or BPWM1, and select BPWM channels. The chosen BPWM channels will start counting at the same time once the synchronous start function is enabled and CNTSEN(BPWM_SSTRG[0]) is set.

  The detail setting of BPWM synchronous start function is described in section 6.11.5.11.

5. From TIMER to EPWM/BPWM

   - Timer Generates Trigger Pulses as EPWM External Clock Source

     Timer0 ~ Timer3 can generates trigger pules as EPWM/BPWM external clock source.

     When timer counter value matches the timer compared value or when the TMx_EXT pin edge transition meets setting, timer can generate a trigger pulse by setting described in section 6.10.5.10.

     The setting of EPWM/BPWM clock source are described in Figure 6.12-3 EPWM Clock Source Control/Figure 6.11-2 BPWM Clock Source Control.

6. From ACMP and LIRC to Timer Capture Function

   - Measure the Time Interval of ACMP0/1 Output Signal or LIRC clock Speed

     Sets the timer capture source from ACMP0/1 output signal or LIRC clock and measures the time interval of the signal by using timer capture function. Users can use the results of time interval to trim LIRC through software or to get the ACMP0/1 output pulse width.

     The detail of time capture function setting are described in section 6.10.5.8 and 6.10.5.9.

7. From Timer0/2 to Timer1/3

   - Inter-Timer Trigger Capture Mode

     Timer0/2 will be forced in event counting mode, counting with external event, and will generate an internal signal (INTR_TMR_TRG) to trigger Timer1/3 start or stop counting. The Timer1/3 will be forced in capture mode and start/stop trigger-counting by Timer0/2 counter status.

     The detail of inter-timer trigger capture mode are described in section 6.10.5.11

8. From EQEI to ECAP

   - ECAP Input noisc filter

     The architecture of ECAP input noise filter is similar to that one used for EQEI. With 6 sampling-rate options, it supports a wide range of filtering noise, the duration of filtered noise and the duration of the signal that is guaranteed to be sampled.

     The detail setting of modulation is described in section 6.13.5.1.

9. From TIMER to EQEI

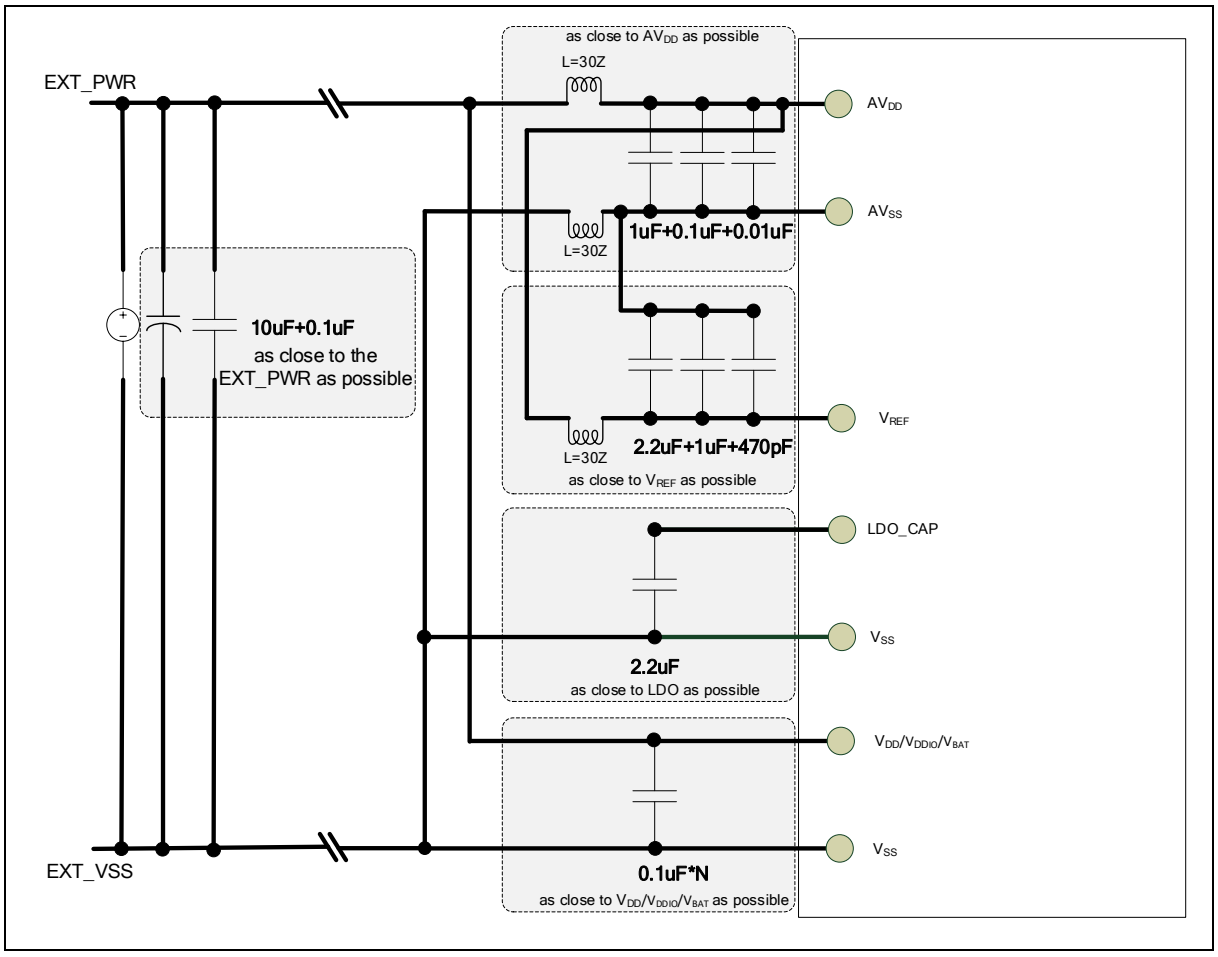
   - TIMER TIF event o EQEI

     When EQEI bit HOLDCNT(QEI_CTL[24]) set, the CNT(QEI_CNT[31:0]) content will be captured into EQEI Counter Hold Register CNTHOLD(QEI_CNTHOLD[31:0]), the data will be held until the next HOLDCNT (QEI_CTL[24]) trigger comes. The bit HOLDCNT can be set by writing 1 to it or the rising edge of timers interrupt flags TIF (TIMERx_INTSTS[0])The detail setting of modulation is described in section 6.13.5.16.

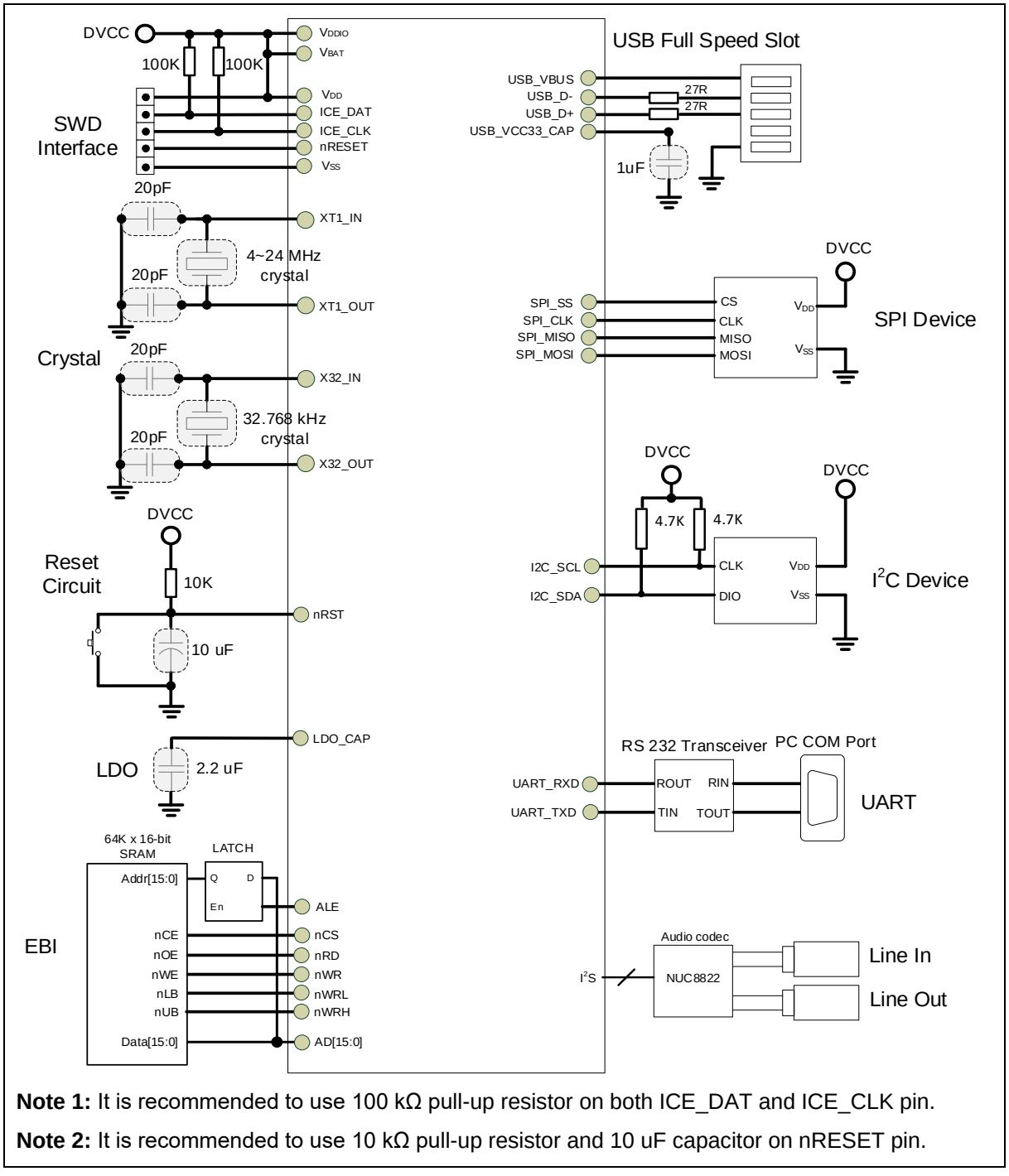
     The detail setting of modulation is described in section 6.10.5.1.

7 APPLICATION CIRCUIT

7.1 Power Supply Scheme



7.2 Peripheral Application Scheme



Note 1: It is recommended to use 100 kΩ pull-up resistor on both ICE_DAT and ICE_CLK pin.

Note 2: It is recommended to use 10 kΩ pull-up resistor and 10 uF capacitor on nRESET pin.

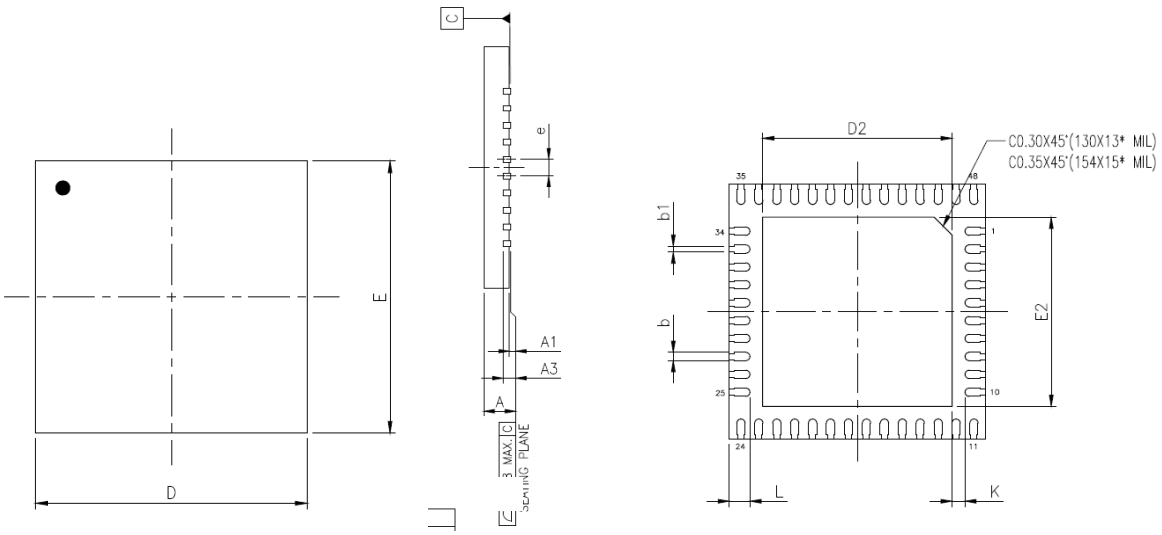
8 ELECTRICAL CHARACTERISTICS

Please refer to the relative Datasheet for detailed information about the M463/M467 series electrical characteristics.

9 PACKAGE DIMENSIONS

Package is Halogen-free, RoHS-compliant and TSCA-compliant.

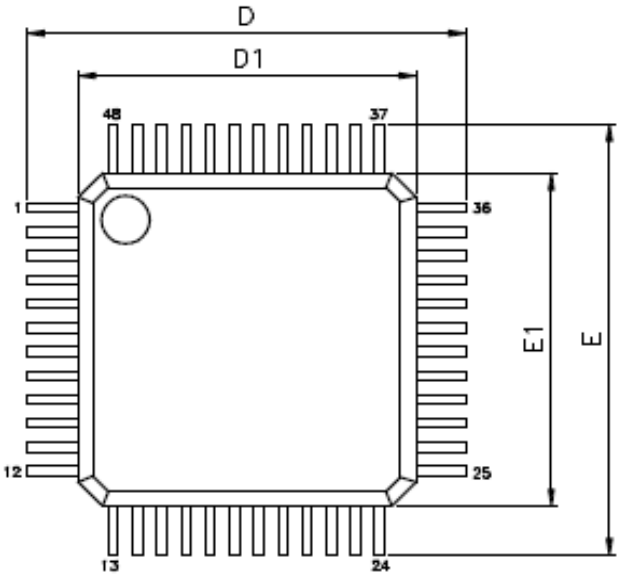
9.1 QFN48L 5x5mm, Thickness 0.80mm, Lead_Pitch 0.35mm, Lead_Length 0.40mm, EP Size 3.75x3.75mm

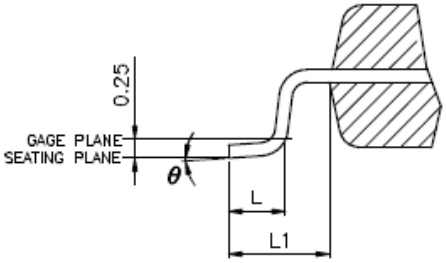


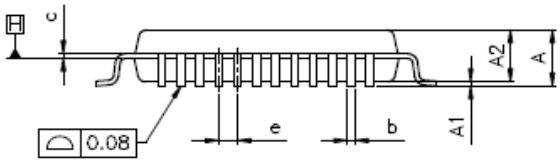
SYMBOLS	MIN.	NOM.	MAX.
A	0.70	0.75	0.80
A1	0.00	0.02	0.05
A3		0.203 REF.	
b	0.13	0.18	0.23
b1	0.07	0.12	0.17
D		5.00 BSC	
E		5.00 BSC	
e		0.35 BSC	
K	0.20	—	—

PAD SIZE	D2			E2			L		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
130X13* MIL	3.65	3.70	3.75	3.65	3.70	3.75	0.35	0.40	0.45
154X15* MIL	3.65	3.70	3.75	3.65	3.70	3.75	0.35	0.40	0.45

9.2 LQFP48L 7x7mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm



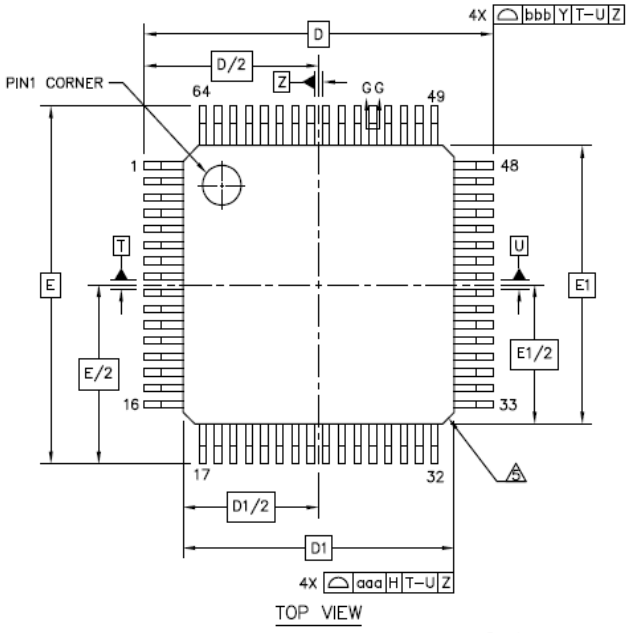


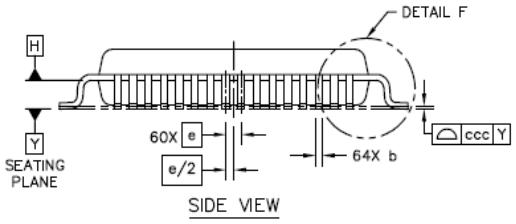


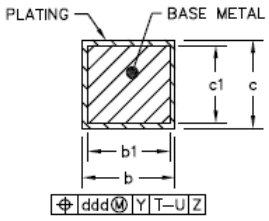
VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

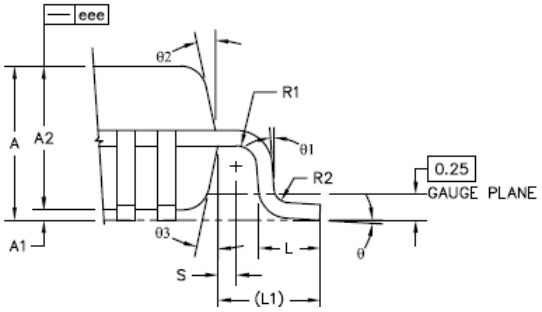
SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.17	0.22	0.27
c	0.09	--	0.20
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.50 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
S	0.20	--	--
θ	0°	3.5°	7°
θ1	0°	--	--
θ2	11°	12°	13°
θ3	11°	12°	13°

9.3 LQFP64L 7x7mm, Thickness 1.40mm, Lead_Pitch 0.40mm, Lead_Length 1.00mm



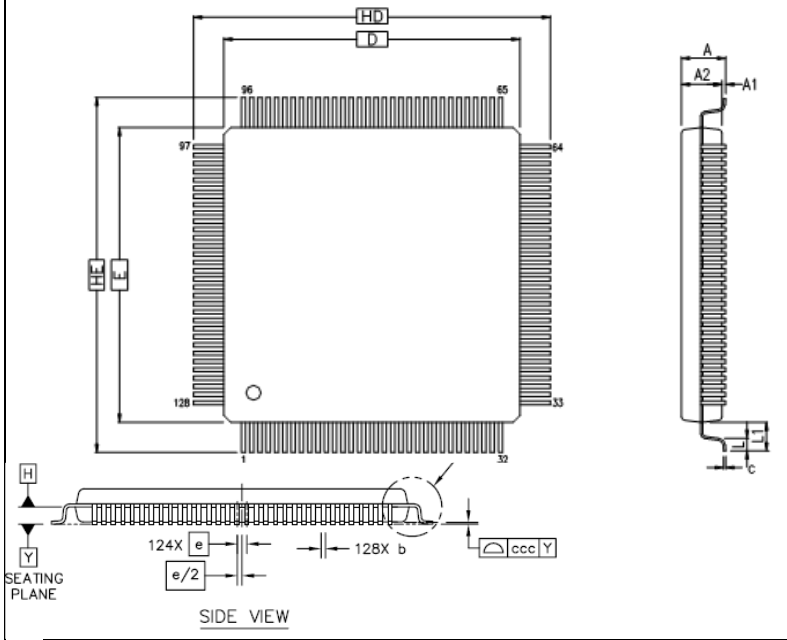






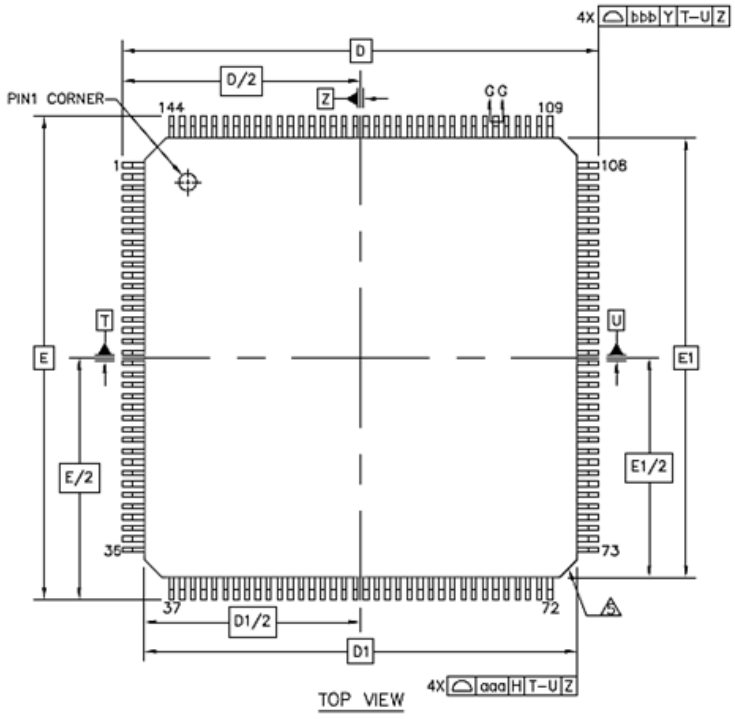
		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.13	0.18	0.23
LEAD WIDTH		b1	0.13	0.16	0.19
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D	9 BSC		
	Y	E	9 BSC		
	X	D1	7 BSC		
	Y	E1	7 BSC		
LEAD PITCH		e	0.4 BSC		
		L	0.45	0.6	0.75
FOOTPRINT		L1	1 REF		
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2	---	---
PACKAGE EDGE TOLERANCE		aaa	0.2		
LEAD EDGE TOLERANCE		bbb	0.2		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.07		
MOLD FLATNESS		eee	0.05		

9.4 LQFP128L 14x14mm, Thickness 1.40mm, Lead_Pitch 0.40mm, Lead_Length 1.00mm

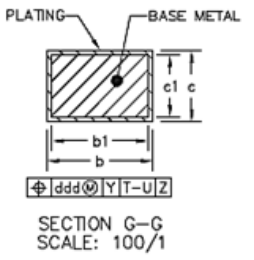


SYMBOL	MIN	NOM.	MAX.
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
HD	15.85	16.00	16.15
D	13.85	14.00	14.15
HE	15.85	16.00	16.15
E	13.85	14.00	14.15
b	0.13	0.16	0.23
e	-	0.40	-
θ°	0	3.5	7
c	0.09	-	0.20
L	0.45	0.60	0.75
L1	-	1.00	-
Y	-	-	0.10

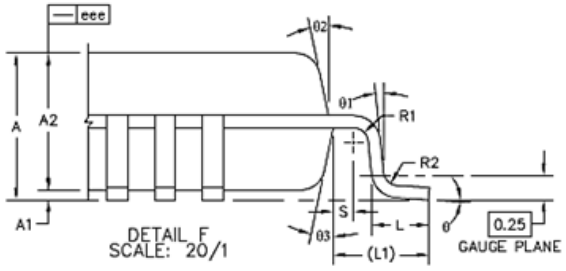
9.5 LQFP144L 20x20mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm



		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.17	0.22	0.27
LEAD WIDTH		b1	0.17	0.2	0.23
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D	22 BSC		
	Y	E	22 BSC		
	X	D1	20 BSC		
	Y	E1	20 BSC		
LEAD PITCH		e	0.5 BSC		
		L	0.45	0.6	0.75
FOOTPRINT		L1	1 REF		
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2	---	---
PACKAGE EDGE TOLERANCE		aaa	0.1		
LEAD EDGE TOLERANCE		bbb	0.2		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.08		
MOLD FLATNESS		eee	0.05		

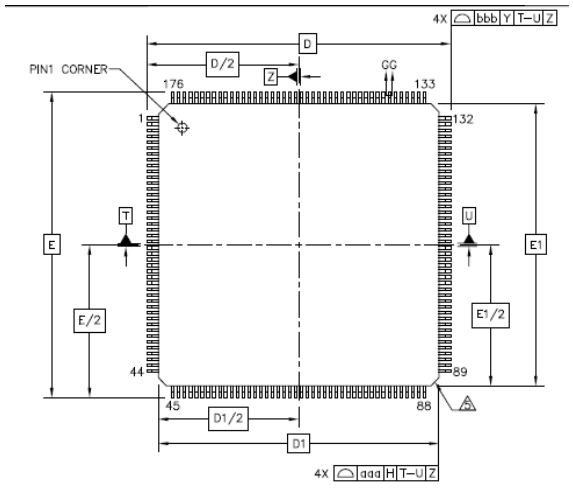


SECTION G-G
SCALE: 100/1



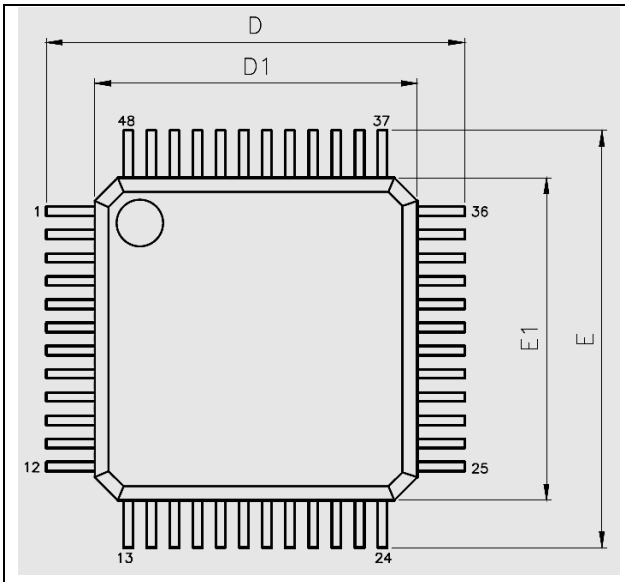
DETAIL F
SCALE: 20/1

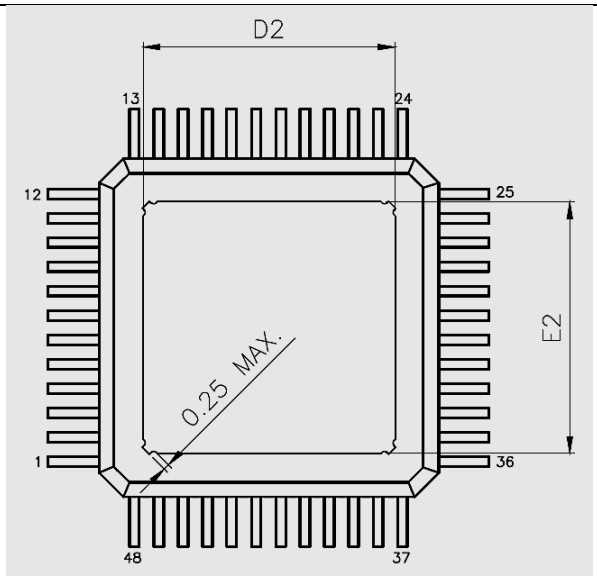
9.6 LQFP176L 24x24mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm

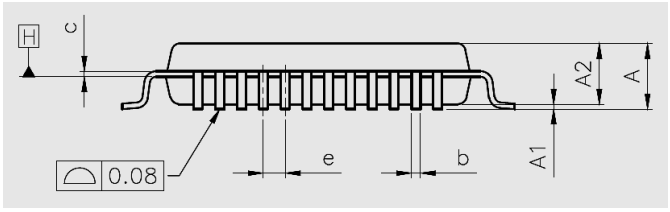


		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.17	0.22	0.27
LEAD WIDTH		b1	0.17	0.2	0.23
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D	26 BSC		
	Y	E	26 BSC		
	X	D1	24 BSC		
	Y	E1	24 BSC		
LEAD PITCH		e	0.5 BSC		
		L	0.45	0.6	0.75
FOOTPRINT		L1	1 REF		
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2 REF.		
PACKAGE EDGE TOLERANCE		aaa	0.1		
LEAD EDGE TOLERANCE		bbb	0.2		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.07		
MOLD FLATNESS		eee	0.05		

9.7 LQFP48L EPAD 7x7mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm, EP_SIZE 5.36x5.36mm

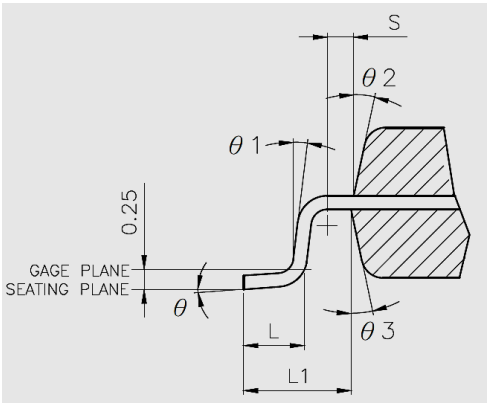






VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.17	0.22	0.27
c	0.09	--	0.20
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
e	0.50 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
S	0.20	--	--
θ	0°	3.5°	7°
θ1	0°	--	--
θ2	11°	12°	13°
θ3	11°	12°	13°



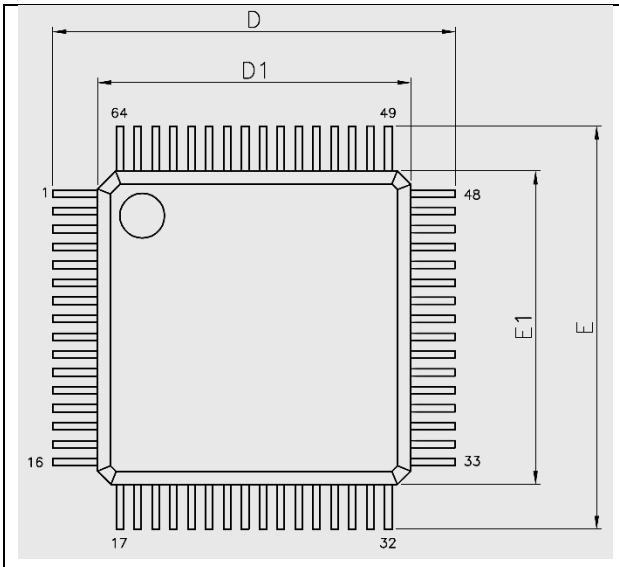
THERMALLY ENHANCED DIMENSIONS(SHOWN IN MM)

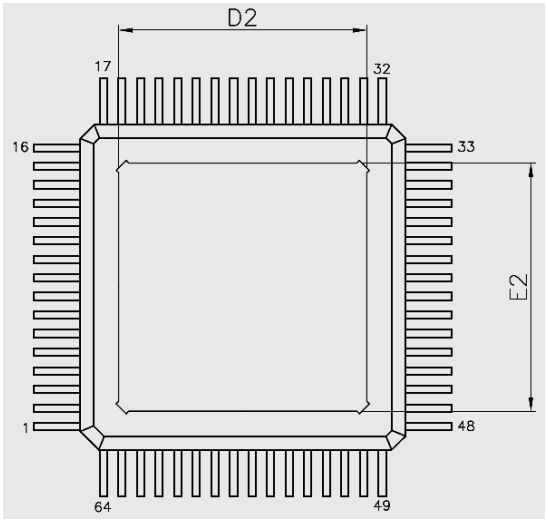
	PAD SIZE	D2			E2		
		MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
3	20*X20*	4.31	5.21	5.36	4.31	5.21	5.36

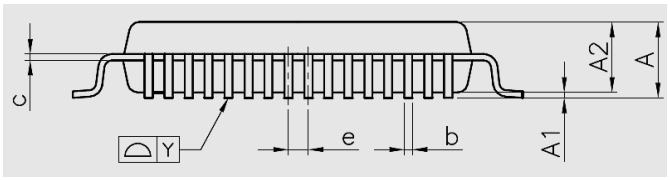
"*"表示汎用字元,此汎用字元可能被其它不同字元所取代,
實際的字元請參照bonding diagram所示.
"*" is an universal character, which means maybe replaced by specific character, the actual character please refers to the bonding diagram.

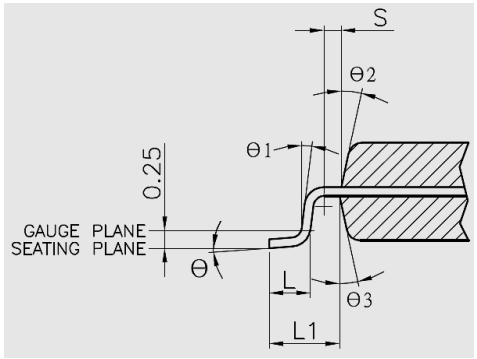
NOTES:
1. JEDEC OUTLINE :
   MS-026 BBC
   MS-026 BBC-HD(THERMALLY ENHANCED VARIATIONS ONLY)
2. DATUM PLANE H IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.
3. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.
4. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

9.8 LQFP64L EPAD 7x7mm, Thickness 1.40mm, Lead_Pitch 0.40mm, Lead_Length 1.00mm, EP_SIZE 5.48x5.48mm









NOTES:

1. JEDEC OUTLINE : MS-026 BBD-HD
2. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25mm PER SIDE. D1 AND E1 ARE MAXIMUM PLASTIC BODY SIZE DIMENSIONS INCLUDING MOLD MISMATCH.
3. DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08mm.
4. DIMENSION Y IS 0.08 MAX BEFORE FT AND 0.089mm MAX AFTER FT.

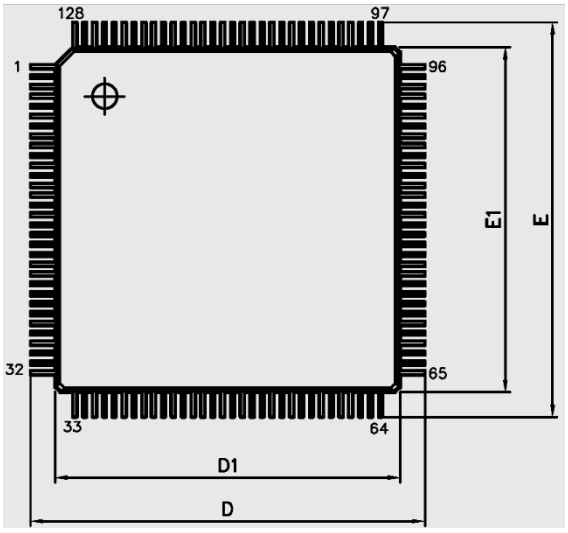
VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

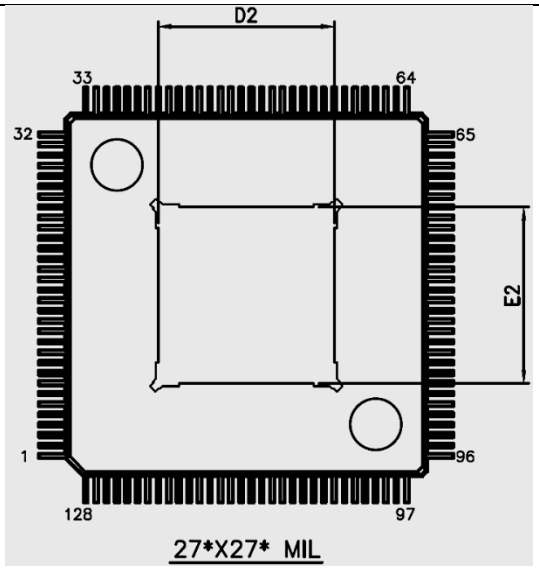
SYMBOLS	MIN.	NOM.	MAX.
A	—	—	1.60
A1	0.05	—	0.15
A2	1.35	1.40	1.45
b	0.13	0.18	0.23
c	0.09	—	0.20
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
e	0.324	0.40	0.476
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
L	0.45	0.60	0.75
L1	1.00 REF		
θ	0°	3.5°	7°
θ1	0°	—	—
θ2	11°	12°	13°
θ3	11°	12°	13°
S	0.20	—	—
Y	—	—	0.08

	PAD SIZE	D2			E2		
		MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
3	21*X21* MIL	5.03	5.33	5.48	5.03	5.33	5.48

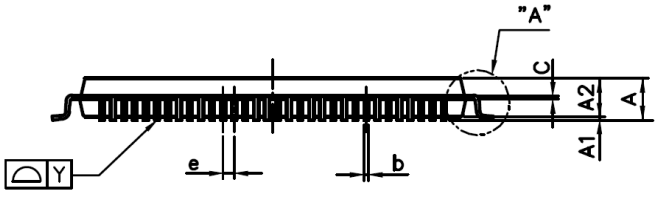
"*"表示汎用字元,此汎用字元可能被其它不同字元所取代,
實際的字元請參照bonding diagram所示.
"*" is an universal character, which means maybe replaced by specific character, the actual character please refers to the bonding diagram.

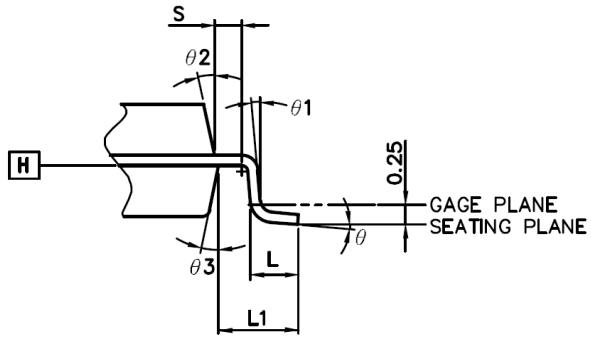
9.9 LQFP128L EPAD 14x14mm, Thickness 1.40mm, Lead_Pitch 0.40mm, Lead_Length 1.00mm, EP_SIZE 7.15x7.15mm





27*X27* MIL





DETAIL A

VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

SYMBOLS	MIN.	NOM.	MAX.
A	--	--	1.60
A1	0.05	--	0.15
A2	1.35	1.40	1.45
b	0.13	--	0.23
c	0.09	--	0.20
D	15.80	16.00	16.20
D1	13.90	14.00	14.10
E	15.80	16.00	16.20
E1	13.90	14.00	14.10
e	0.40 BSC		
L	0.45	0.60	0.75
L1	1.00 REF		
θ	0°	3.5°	7°
θ1	0°	--	--
θ2	11°	12°	13°
θ3	11°	12°	13°
S	0.20	--	--
Y	0.08		

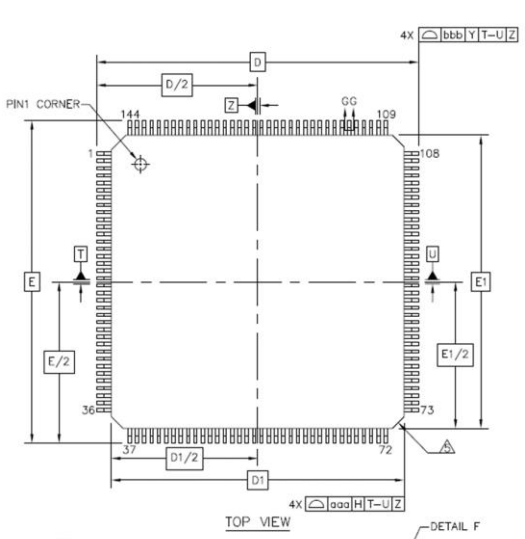
NOTES:
1.JEDEC NO. : N/A.
2.DATUM PLANE [H] IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.
3.DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 mm PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE [H].
4.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

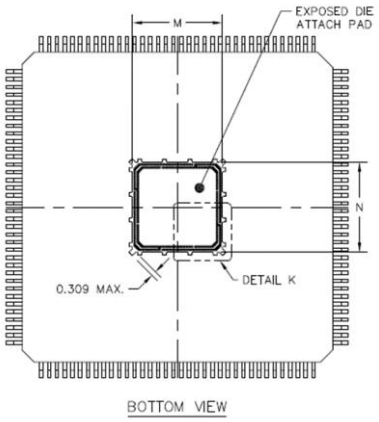
THERMALLY ENHANCED DIMENSIONS(SHOWN IN MM)

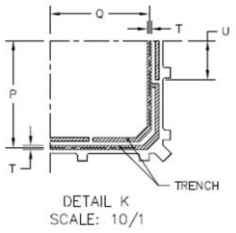
PAD SIZE	D2			E2		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
⚠4 27*X27* MIL	6.70	7.00	7.15	6.70	7.00	7.15

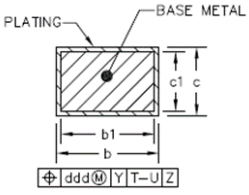
"*"表示汎用字元,此汎用字元可能被其它不同字元所取代,
實際的字元請參照bonding diagram所示.
"*" is an universal character, which means maybe replaced by specific character, the actual character please refers to the bonding diagram.

9.10 LQFP144L EPAD 20x20mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm, EP_SIZE 5.9x5.9mm

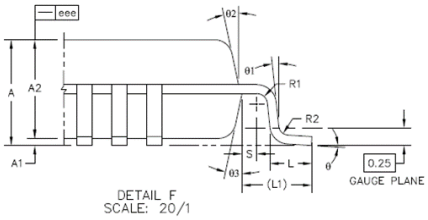








SECTION G–G
SCALE: 100/1



DETAIL F
SCALE: 20/1

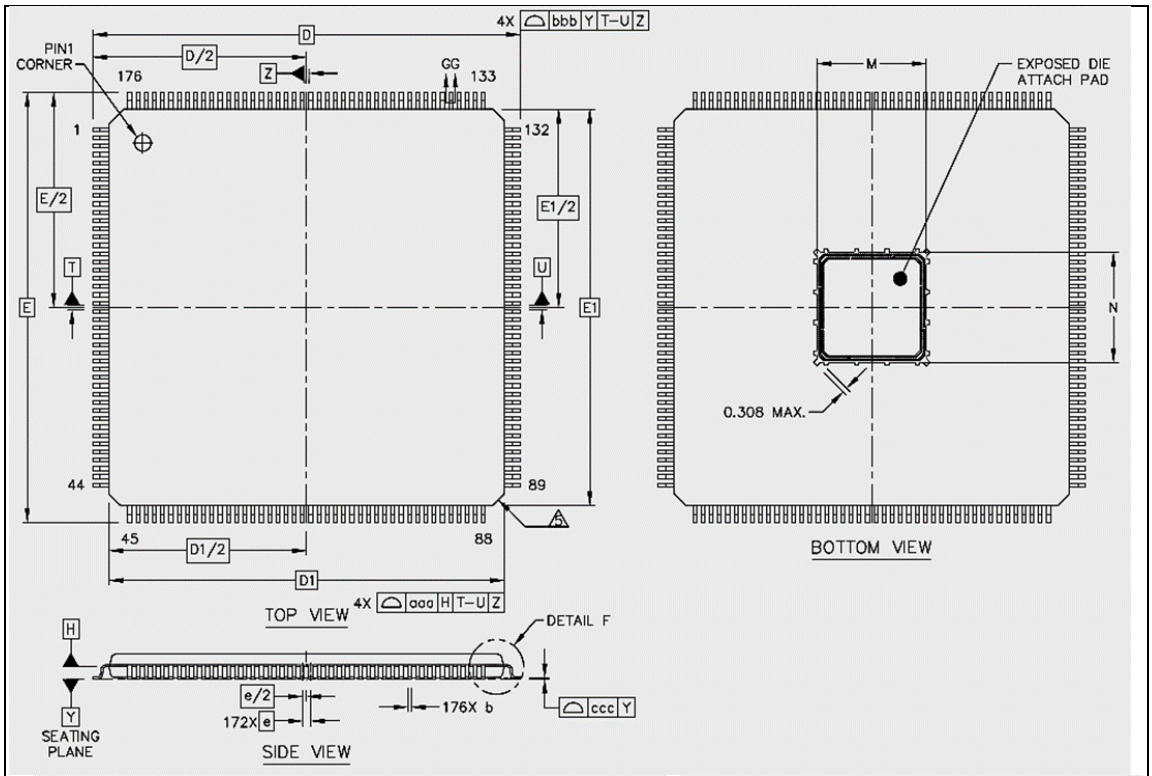
NOTES

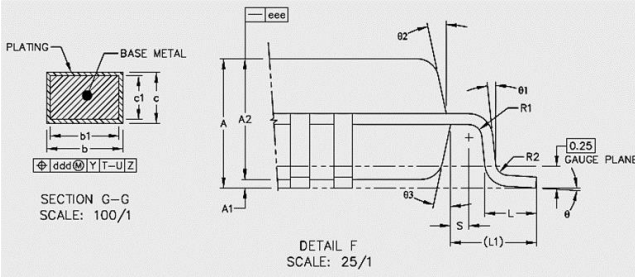
1. DATUM T, U, AND Z TO BE DETERMINED AT DATUM PLANE H.
2. DIMENSIONS D AND E TO BE DETERMINED AT SEATING PLANE DATUM Y.
3. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLDPROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE DATUM H.
4. DIMENSION b DOES NOT INCLUDE DAM BAR PROTRUSION. ALLOWABLE DAM BAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUM b DIMENSION BY MORE THAN 0.08 mm. DAM BAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND AN ADJACENT LEAD IS 0.07 mm.

⚠ EXACT SHAPE OF EACH CORNER IS OPTIONAL.

		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.17	0.22	0.27
LEAD WIDTH		b1	0.17	0.2	0.23
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D	22 BSC		
	Y	E	22 BSC		
	X	D1	20 BSC		
	Y	E1	20 BSC		
LEAD PITCH		e	0.5 BSC		
		L	0.45	0.6	0.75
FOOTPRINT		L1	1 REF		
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2	---	---
EP SIZE	X	M	5.7	5.8	5.9
	Y	N	5.7	5.8	5.9
		P	2.7	2.75	2.8
		Q	2.5	2.55	2.6
		T	0.05	---	0.15
		U	0.95	---	1.05
PACKAGE EDGE TOLERANCE		aaa	0.1		
LEAD EDGE TOLERANCE		bbb	0.2		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.08		
MOLD FLATNESS		eee	0.05		

9.11 LQFP176L EPAD 24x24mm, Thickness 1.40mm, Lead_Pitch 0.50mm, Lead_Length 1.00mm, EP_SIZE 6.75x6.75mm





		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.17	0.22	0.27
LEAD WIDTH		b1	0.17	0.2	0.23
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D		26 BSC	
	Y	E		26 BSC	
	X	D1		24 BSC	
	Y	E1		24 BSC	
LEAD PITCH		e		0.5 BSC	
		L	0.45	0.6	0.75
FOOTPRINT		L1		1 REF	
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2	---	---
EP SIZE	X	M	6.55	6.65	6.75
	Y	N	6.55	6.65	6.75
PACKAGE EDGE TOLERANCE		aaa		0.1	
LEAD EDGE TOLERANCE		bbb		0.2	
COPLANARITY		ccc		0.08	
LEAD OFFSET		ddd		0.08	
MOLD FLATNESS		eee		0.05	

NOTES
1. DATUM T, U, AND Z TO BE DETERMINED AT DATUM PLANE H.
2. DIMENSIONS D AND E TO BE DETERMINED AT SEATING PLANE DATUM Y.
3. DIMENSIONS D1 AND E1 DO NOT INCLUDE MOLDPROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS D1 AND E1 DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE DATUM H.
4. DIMENSION b DOES NOT INCLUDE DAM BAR PROTRUSION. ALLOWABLE DAM BAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED THE MAXIMUMb DIMENSION BY MORE THAN 0.08 mm. DAM BAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND AN ADJACENT LEAD IS 0.07 mm.

5 EXACT SHAPE OF EACH CORNER IS OPTIONAL.

10 ABBREVIATIONS

10.1 Abbreviations

Acronym	Description
ACMP	Analog Comparator Controller
ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
APB	Advanced Peripheral Bus
AHB	Advanced High-Performance Bus
BOD	Brown-out Detection
CAN	Controller Area Network
CCAP	Camera Capture Interface
DAP	Debug Access Port
DES	Data Encryption Standard
EADC	Enhanced Analog-to-Digital Converter
EBI	External Bus Interface
EMAC	Ethernet MAC Controller
EPWM	Enhanced Pulse Width Modulation
FIFO	First In, First Out
FMC	Flash Memory Controller
FPU	Floating-point Unit
GPIO	General-Purpose Input/Output
HCLK	The Clock of Advanced High-Performance Bus
HIRC	12 MHz Internal High Speed RC Oscillator
HXT	4~24 MHz External High Speed Crystal Oscillator
IAP	In Application Programming
ICP	In Circuit Programming
ISP	In System Programming
LDO	Low Dropout Regulator
LIN	Local Interconnect Network
LIRC	10 kHz internal low speed RC oscillator (LIRC)
MPU	Memory Protection Unit
NVIC	Nested Vectored Interrupt Controller
PCLK	The Clock of Advanced Peripheral Bus
PDMA	Peripheral Direct Memory Access
PLL	Phase-Locked Loop

PWM	Pulse Width Modulation
EQEI	Enhanced Quadrature Encoder Interface
SD	Secure Digital
SPI	Serial Peripheral Interface
SPS	Samples per Second
TDES	Triple Data Encryption Standard
TK	Touch Key
TMR	Timer Controller
UART	Universal Asynchronous Receiver/Transmitter
UCID	Unique Customer ID
USB	Universal Serial Bus
WDT	Watchdog Timer
WWDT	Window Watchdog Timer

Table 10.1-1 List of Abbreviations

11 REVISION HISTORY

Date	Revision	Description
2022.12.26	1.00	• Initial version.
2023.08.24	1.01	• Added new parts M463L2GCAC, M463S2GCAC, M463K2GCAC, M467S2JHAE, M467K2JHAE, M467J2JHAE and M467H2JHAE in section 3.1.1, 3.3.1 and 3.3.2.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.