

# M071Q/M071V Series CMSIS BSP Guide

Directory Introduction for 32-bit NuMicro® Family

## Directory Information

Please extract the “M071Q M071V Series BSP CMSIS V3.00.003.zip” file firstly, and then put the “M071Q M071V Series BSP CMSIS V3.00.003” folder into the working folder (e.g. .\Nuvoton\BSP Library).

### This BSP folder contents:

<b>Document</b>	Device driver reference manual and reversion history.
<b>Library</b>	Device driver header and source files.
<b>SampleCode</b>	Device driver sample code.

*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 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](http://www.nuvoton.com)

**TABLE OF CONTENTS**

**1 DOCUMENT.....4**

**2 LIBRARY .....5**

**3 SAMPLECODE .....6**

**4 SAMPLECODE\ISP .....7**

**5 SAMPLECODE\REGBASED .....8**

**System Manager (SYS) .....8**

**Clock Controller (CLK).....8**

**Flash Memory Controller (FMC).....8**

**General Purpose I/O (GPIO).....8**

**PDMA Controller (PDMA) .....9**

**Timer Controller (TIMER).....9**

**Watchdog Timer (WDT) .....10**

**Window Watchdog Timer (WWDT) .....10**

**Real Timer Clock (RTC) .....10**

**PWM Generator and Capture Timer (PWM).....10**

**UART Interface Controller (UART).....10**

**Smart Card Host Interface (SC).....11**

**I<sup>2</sup>S Controller (I<sup>2</sup>S).....12**

**Serial Peripheral Interface (SPI).....12**

**I<sup>2</sup>C Serial Interface Controller (I<sup>2</sup>C) .....13**

**Universal Serial Control Interface Controller - UART Mode (USCI-UART) .....13**

**Universal Serial Control Interface Controller - SPI Mode (USCI-SPI) .....14**

**Universal Serial Control Interface Controller - I<sup>2</sup>C Mode (USCI-I2C) .....14**

**External Bus Interface (EBI) .....15**

**CRC Controller (CRC) .....15**

**Analog-to-Digital Converter (ADC) .....15**

**Analog Comparator Controller (ACMP) .....16**

**Hardware Divider (HDIV).....16**

**6 SAMPLECODE\STDDRIVER..... 17**

- System Manager (SYS) .....17
- Clock Controller (CLK).....17
- Flash Memory Controller (FMC).....17
- General Purpose I/O (GPIO).....17
- PDMA Controller (PDMA) .....18
- Timer Controller (TIMER).....18
- Watchdog Timer (WDT) .....19
- Window Watchdog Timer (WWDT) .....19
- Real Timer Clock (RTC) .....19
- PWM Generator and Capture Timer (PWM).....19
- UART Interface Controller (UART).....20
- Smart Card Host Interface (SC).....20
- I<sup>2</sup>S Controller (I<sup>2</sup>S).....21
- Serial Peripheral Interface (SPI).....21
- I<sup>2</sup>C Serial Interface Controller (I<sup>2</sup>C) .....22
- Universal Serial Control Interface Controller - UART Mode (USCI-UART) .....22
- Universal Serial Control Interface Controller - SPI Mode (USCI-SPI) .....23
- Universal Serial Control Interface Controller - I<sup>2</sup>C Mode (USCI-I2C) .....23
- External Bus Interface (EBI) .....24
- CRC Controller (CRC) .....24
- Analog-to-Digital Converter (ADC) .....24
- Analog Comparator Controller (ACMP) .....25
- Hardware Divider (HDIV).....25

## 1 Document

<p><b>CMSIS.html</b></p>	<p>Introduction of CMSIS version 4.5.0. CMSIS components included CMSIS-CORE, CMSIS-Driver, CMSIS-DSP, etc.</p> <ul style="list-style-type: none"> <li>● CMSIS-CORE: API for the Cortex-M0 processor core and peripherals.</li> <li>● CMSIS-Driver: Defines generic peripheral driver interfaces for middleware making it reusable across supported devices.</li> </ul> <p>CMSIS-DSP: DSP Library Collection with over 60 Functions for various data types: fix-point (fractional q7, q15, q31) and single precision floating-point (32-bit).</p>
<p><b>Revision History.pdf</b></p>	<p>The revision history of M071Q/M071V Series BSP.</p>
<p><b>NuMicro M071Q M071V Series Driver Reference Guide.chm</b></p>	<p>The usage of drivers in M071Q/M071V Series BSP.</p>

## 2 Library

<b>CMSIS</b>	Cortex <sup>®</sup> Microcontroller Software Interface Standard (CMSIS) V4.5.0 definitions by ARM <sup>®</sup> Corp.
<b>Device</b>	CMSIS compliant device header file.
<b>SmartcardLib</b>	Library for accessing a smartcard.
<b>StdDriver</b>	All peripheral driver header and source files.

### 3 SampleCode

<b>Hard_Fault_Sample</b>	<p>Show hard fault information when hard fault happened.</p> <p>The hard fault handler show some information included program counter, which is the address where the processor was executing when the hard fault occur. The listing file (or map file) can show what function and instruction that was.</p> <p>It also shows the Link Register (LR), which contains the return address of the last function call. It can show the status where CPU comes from to get to this point.</p>
<b>ISP</b>	Sample codes for In-System-Programming.
<b>Semihost</b>	Show how to print and get character through IDE console window.
<b>RegBased</b>	The sample code able to access control registers directly.
<b>StdDriver</b>	Demonstrate the usage of M071Q/M071V series MCU peripheral driver APIs.
<b>Template</b>	A project template for M071Q/M071V series MCU.

## **4 SampleCode\ISP**

ISP_I2C	In-System-Programming Sample code through I2C interface.
ISP_RS485	In-System-Programming Sample code through RS485 interface.
ISP_SPI	In-System-Programming Sample code through SPI interface.
ISP_UART	In-System-Programming Sample code through UART interface.

## 5 SampleCode\RegBased

### System Manager (SYS)

<b>SYS_BODWakeup</b>	Show how to wake up system form Power-down mode by brown-out detector interrupt.
<b>SYS_PLLClockOutput</b>	Change system clock to different PLL frequency and output system clock from CLKO pin.
<b>SYS_PowerDown_MinCurrent</b>	Demonstrate how to minimize power consumption when entering power down mode.
<b>SYS_VoltageDetector</b>	Show how to use voltage detector to detect pin input voltage.

### Clock Controller (CLK)

<b>CLK_ClockDetector</b>	Show the usage of clock fail detector and clock frequency monitor function.
--------------------------	---

### Flash Memory Controller (FMC)

<b>FMC_ExecInSRAM</b>	Implement a code and execute the code in SRAM to program embedded Flash (support KEIL MDK only).
<b>FMC_IAP</b>	Show how to call LDROM functions from APROM. The code in APROM will look up the table at 0x100E00 to get the address of function of LDROM and call the function.
<b>FMC_MultiBoot</b>	Implement a multi-boot system to boot from different applications in APROM. A LDROM code and four APROM code are implemented in this sample code.
<b>FMC_RW</b>	Demonstrate how to read/program embedded Flash by ISP function.

### General Purpose I/O (GPIO)

<b>GPIO_EINTAndDebounce</b>	Show the usage of GPIO external interrupt function and de-bounce function.
-----------------------------	--

<b>GPIO_INT</b>	Show the usage of GPIO interrupt function.
<b>GPIO_OutputInput</b>	Show how to set GPIO pin mode and use pin data input/output control.
<b>GPIO_PowerDown</b>	Show how to wake up system from Power-down mode by GPIO interrupt.

**PDMA Controller (PDMA)**

<b>PDMA</b>	Use PDMA Channel 2 to transfer data from memory to memory.
<b>PDMA_ScatterGather_PingPong Buffer</b>	Use PDMA to implement Ping-Pong buffer by scatter-gather mode (memory to memory).
<b>PDMA_Scatter_Gather</b>	Use PDMA Channel 4 to transfer data from memory to memory by scatter-gather mode.

**Timer Controller (TIMER)**

<b>TIMER_CaptureCounter</b>	Show how to use the timer2 capture function to capture timer2 counter value.
<b>TIMER_EventCounter</b>	Show how to use the timer2 capture function to capture timer2 counter value.
<b>TIMER_PeriodicINT</b>	Implement timer counting in periodic mode.
<b>TIMER_PWM_ChangeDuty</b>	Change duty cycle and period of output waveform in PWM down count type.
<b>TIMER_PWM_DeadTime</b>	Demonstrate Timer PWM Complementary mode and Dead-Time function.
<b>TIMER_PWM_OuputWaveform</b>	Demonstrate output different duty waveform in Timer0~Timer3 PWM.
<b>TIMER_TimeoutWakeup</b>	Use timer0 periodic time-out interrupt event to wake up system.

### Watchdog Timer (WDT)

<b>WDT_TimeoutWakeupAndReset</b>	Implement WDT time-out interrupt event to wake up system and generate time-out reset system event while WDT time-out reset delay period expired.
----------------------------------	--

### Window Watchdog Timer (WWDT)

<b>WWDT_CompareINT</b>	Show how to reload the WWDT counter value.
------------------------	--

### Real Timer Clock (RTC)

<b>RTC_AlarmWakeup</b>	Use RTC alarm interrupt event to wake up system.
<b>RTC_TimeAndTick</b>	Show the current RTC data/time per tick.

### PWM Generator and Capture Timer (PWM)

<b>PWM_Capture</b>	Capture the PWM1 Channel 0 waveform by PWM1 Channel 2.
<b>PWM_DeadZone</b>	Demonstrate how to use PWM Dead Zone function.
<b>PWM_DoubleBuffer_PeriodLoadingMode</b>	Change duty cycle and period of output waveform by PWM Double Buffer function (Period loading mode).
<b>PWM_DutySwitch</b>	Change duty cycle of output waveform by configured period.
<b>PWM_OutputWaveform</b>	Demonstrate how to use PWM output waveform.
<b>PWM_PDMA_Capture</b>	Capture the PWM1 Channel 0 waveform by PWM1 Channel 2, and use PDMA to transfer captured data.
<b>PWM_SyncStart</b>	Demonstrate how to use PWM counter synchronous start function.

### UART Interface Controller (UART)

<b>UART_AutoBaudRate_Master</b>	Show how to use auto baud rate detection function. This sample code needs to work with
---------------------------------	--

	UART_AutoBaudRate_Slave.
<b>UART_AutoBaudRate_Slave</b>	Show how to use auto baud rate detection function. This sample code needs to work with UART_AutoBaudRate_Master.
<b>UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Slave.
<b>UART_Autoflow_Slave</b>	Transmit and receive data with auto flow control. This sample code needs to work with UART_Autoflow_Master.
<b>UART_IrDA_Master</b>	Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Slave.
<b>UART_IrDA_Slave</b>	Transmit and receive data in UART IrDA mode. This sample code needs to work with UART_IrDA_Master.
<b>UART_LIN</b>	Transmit LIN frame including header and response in UART LIN mode.
<b>UART_PDMA</b>	Transmit and receive UART data with PDMA.
<b>UART_RS485_Master</b>	Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Slave.
<b>UART_RS485_Slave</b>	Transmit and receive data in UART RS485 mode. This sample code needs to work with UART_RS485_Master.
<b>UART_TxRxFunction</b>	Transmit and receive data from PC terminal through RS232 interface.
<b>UART_Wakeup</b>	Show how to wake up system from Power-down mode by UART interrupt.

**Smart Card Host Interface (SC)**

<b>SCUART_TxRx</b>	Show Smartcard UART by connecting PA.0 and PA.1 pins.
<b>SC_ReadATR</b>	Read Smartcard ATR from the SC0 port.

**I<sup>2</sup>S Controller (I<sup>2</sup>S)**

<b>I2S_Master</b>	Configure SPI1 as I2S Master mode and demonstrate how I2S works in Master mode. This sample code needs to work with I2S_Slave.
<b>I2S_PDMA_NAU8822</b>	This is an I2S demo with PDMA function connected with NAU8822 codec.
<b>I2S_PDMA_Play</b>	This is an I2S demo for playing data and demonstrating how I2S works with PDMA.
<b>I2S_PDMA_PlayRecord</b>	This is an I2S demo for playing and recording data with PDMA function.
<b>I2S_PDMA_Record</b>	This is an I2S demo for recording data and demonstrating how I2S works with PDMA.
<b>I2S_Slave</b>	Configure SPI1 as I2S Slave mode and demonstrate how I2S works in Slave mode. This sample code needs to work with I2S_Master.

**Serial Peripheral Interface (SPI)**

<b>SPI_Loopback</b>	Implement SPI Master loop back transfer. This sample code needs to connect SPI0_MISO0 pin and SPI0_MOSI0 pin together. It will compare the received data with transmitted data.
<b>SPI_MasterFifoMode</b>	Configure SPI0 as Master mode and demonstrate how to communicate with an off-chip SPI Slave device with FIFO mode. This sample code needs to work with SPI_SlaveFifoMode.
<b>SPI_PDMA_LoopTest</b>	Demonstrate SPI data transfer with PDMA. SPI0 will be configured as Master mode and SPI1 will be configured as Slave mode. Both TX PDMA function and RX PDMA function will be enabled.
<b>SPI_SlaveFifoMode</b>	Configure SPI0 as Slave mode and demonstrate how to communicate with an off-chip SPI Master device with FIFO mode. This sample code needs to work with SPI_MasterFifoMode.

**I<sup>2</sup>C Serial Interface Controller (I<sup>2</sup>C)**

<b>I2C_EEPROM</b>	Demonstrate how to access EEPROM through a I2C interface
<b>I2C_GCMode_Master</b>	Demonstrate how a Master uses I2C address 0x0 to write data to I2C Slave. This sample code needs to work with I2C_GCMode_Slave.
<b>I2C_GCMode_Slave</b>	Demonstrate how to receive Master data in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.
<b>I2C_Loopback</b>	Demonstrate how a Master accesses a Slave.
<b>I2C_Master</b>	Demonstrate how a Master accesses a Slave. This sample code needs to work with I2C_Slave.
<b>I2C_Master_PDMA</b>	Demonstrate how a Master accesses Slave using PDMA TX mode and PDMA RX mode.
<b>I2C_Slave</b>	Demonstrate how to set I2C in slave mode to receive data from a Master. This sample code needs to work with I2C_Master.
<b>I2C_Slave_PDMA</b>	Demonstrate how a Slave uses PDMA Rx mode to receive data from a Master.
<b>I2C_Wakeup_Slave</b>	Demonstrate how to set I2C to wake up MCU from Power-down mode. This sample code needs to work with I2C_Master.

**Universal Serial Control Interface Controller - UART Mode (USCI-UART)**

<b>USCI_UART_AutoBaudRate_Master</b>	Show how to use auto baud rate detection function. This sample code needs to work with USCI_UART_AutoBaudRate_Slave.
<b>USCI_UART_AutoBaudRate_Slave</b>	Show how to use auto baud rate detection function. This sample code needs to work with USCI_UART_AutoBaudRate_Master.
<b>USCI_UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. This sample code needs to work with

	USCI_UART_AutoFlow_Slave.
<b>USCI_UART_AutoFlow_Slave</b>	Transmit and receive data with auto flow control. This sample code needs to work with USCI_UART_AutoFlow_Master.
<b>USCI_UART_RS485_Master</b>	Transmit and receive data in RS485 mode. This sample code needs to work with USCI_UART_RS485_Slave.
<b>USCI_UART_RS485_Slave</b>	Transmit and receive data in RS485 mode. This sample code needs to work with USCI_UART_RS485_Master.
<b>USCI_UART_TxRxFuntion</b>	Transmit and receive data from PC terminal through RS232 interface.
<b>USCI_UART_Wakeup</b>	Show how to wake up system from Power-down mode by USCI interrupt in UART mode.

**Universal Serial Control Interface Controller - SPI Mode (USCI-SPI)**

<b>USCI_SPI_Loopback</b>	Implement USCI_SPI1 Master loop back transfer. This sample code needs to connect USCI_SPI1_MISO pin and USCI_SPI1_MOSI pin together. It will compare the received data with transmitted data.
<b>USCI_SPI_MasterMode</b>	Configure USCI_SPI1 as Master mode and demonstrate how to communicate with an off-chip SPI Slave device. This sample code needs to work with USCI_SPI_SlaveMode.
<b>USCI_SPI_SlaveMode</b>	Configure USCI_SPI1 as Slave mode and demonstrate how to communicate with an off-chip SPI Master device. This sample code needs to work with USCI_SPI_MasterMode.

**Universal Serial Control Interface Controller - I<sup>2</sup>C Mode (USCI-I2C)**

<b>USCI_I2C_EEPROM</b>	Demonstrate how to access EEPROM through a USCI_I2C interface
<b>USCI_I2C_Loopback</b>	Demonstrate how a Master accesses Slave.
<b>USCI_I2C_Loopback_10bit</b>	Demonstrate how a Master uses 10-bit addressing

	access Slave.
<b>USCI_I2C_Master</b>	Demonstrate how a Master accesses Slave. This sample code needs to work with I2C_Slave.
<b>USCI_I2C_Master_10bit</b>	Demonstrate how a Master uses 10-bit addressing access Slave. This sample code needs to work with I2C_Slave.
<b>USCI_I2C_Slave</b>	Demonstrate how to set I2C in Slave mode to receive data from a Master. This sample code needs to work with I2C_Master.
<b>USCI_I2C_Slave_10bit</b>	Demonstrate how to set I2C in 10-bit addressing slave mode to receive data from a Master. This sample code needs to work with I2C_Master.
<b>USCI_I2C_Wakeup_Slave</b>	Demonstrate how to set I2C to wake up MCU from Power-down mode. This sample code needs to work with I2C_Master.

**External Bus Interface (EBI)**

<b>EBI_NOR</b>	Configure EBI interface to access MX29LV320T (NOR Flash) on EBI interface.
<b>EBI_SRAM</b>	Configure EBI interface to access BS616LV4017 (SRAM) on EBI interface.

**CRC Controller (CRC)**

<b>CRC_CCITT</b>	Implement CRC in CRC-CCITT mode and get CRC checksum results.
<b>CRC_CRC32</b>	Implement CRC in CRC-32 mode with PDMA transfer.
<b>CRC_CRC8</b>	Implement CRC in CRC-8 mode and get CRC checksum results.

**Analog-to-Digital Converter (ADC)**

<b>ADC_ContinuousScanMode</b>	Perform A/D Conversion with ADC continuous scan
-------------------------------	---

	mode.
<b>ADC_PDMA_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan mode and transfer result by PDMA.
<b>ADC_PwmTrigger</b>	Demonstrate how to trigger ADC by PWM.
<b>ADC_ResultMonitor</b>	Monitor the conversion result of Channel 2 by the digital compare function.
<b>ADC_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan mode.
<b>ADC_SingleMode</b>	Perform A/D Conversion with ADC single mode.

**Analog Comparator Controller (ACMP)**

<b>ACMP</b>	Demonstrate how Analog Comparator (ACMP) works with internal band-gap voltage.
<b>ACMP_Wakeup</b>	Show how to wake up MCU from Power-down mode by ACMP wake-up function.

**Hardware Divider (HDIV)**

<b>HDIV</b>	Show how to use divider API and how to use hardware divider by control registers.
-------------	---

## 6 SampleCode\StdDriver

### System Manager (SYS)

<b>SYS_BODWakeup</b>	Show how to wake up system form Power-down mode by brown-out detector interrupt.
<b>SYS_PLLClockOutput</b>	Change system clock to different PLL frequency and output system clock from CLKO pin.
<b>SYS_PowerDown_MinCurrent</b>	Demonstrate how to minimize power consumption when entering power down mode.
<b>SYS_VoltageDetector</b>	Show how to use voltage detector to detect pin input voltage.

### Clock Controller (CLK)

<b>CLK_ClockDetector</b>	Show the usage of clock fail detector and clock frequency monitor function.
--------------------------	---

### Flash Memory Controller (FMC)

<b>FMC_ExecInSRAM</b>	Implement a code and execute the code in SRAM to program embedded Flash (support KEIL MDK only).
<b>FMC_IAP</b>	Show how to call LDROM functions from APROM. The code in APROM will look up the table at 0x100E00 to get the address of function of LDROM and call the function.
<b>FMC_RW</b>	Demonstrate how to read/program embedded Flash by ISP function.

### General Purpose I/O (GPIO)

<b>GPIO_EINTAndDebounce</b>	Show the usage of GPIO external interrupt function and de-bounce function.
<b>GPIO_INT</b>	Show the usage of GPIO interrupt function.
<b>GPIO_OutputInput</b>	Show how to set GPIO pin mode and use pin data

	input/output control.
<b>GPIO_PowerDown</b>	Show how to wake up system from Power-down mode by GPIO interrupt.

**PDMA Controller (PDMA)**

<b>PDMA</b>	Use PDMA Channel 2 to transfer data from memory to memory.
<b>PDMA_ScatterGather_PingPong Buffer</b>	Use PDMA to implement Ping-Pong buffer by scatter-gather mode (memory to memory).
<b>PDMA_Scatter_Gather</b>	Use PDMA Channel 4 to transfer data from memory to memory by scatter-gather mode.

**Timer Controller (TIMER)**

<b>TIMER_ACMPTrigger</b>	Show how to use ACMP0 to trigger Timer capture event.
<b>TIMER_CaptureCounter</b>	Show how to use the timer2 capture function to capture timer2 counter value.
<b>TIMER_Delay</b>	Show how to use timer0 to create various delay time.
<b>TIMER_EventCounter</b>	Implement timer1 event counter function to count the external input event.
<b>TIMER_InterTimerTriggerMode</b>	Demonstrate how to use Inter-Timer trigger function.
<b>TIMER_PeriodicINT</b>	Implement timer counting in periodic mode.
<b>TIMER_PWM_Brake</b>	Generate Timer brake event by Timer brake pin.
<b>TIMER_PWM_ChangeDuty</b>	Change duty cycle and period of output waveform in PWM down count type.
<b>TIMER_PWM_DeadTime</b>	Demonstrate Timer PWM Complementary mode and Dead-Time function.
<b>TIMER_PWM_OuputWaveform</b>	Demonstrate output different duty waveform in Timer0~Timer3 PWM.

<b>TIMER_TimeoutWakeup</b>	Use timer0 periodic time-out interrupt event to wake up system.
<b>TIMER_ToggleOut</b>	Implement timer counting in toggle-output mode.

**Watchdog Timer (WDT)**

<b>WDT_TimeoutWakeupAndReset</b>	Implement WDT time-out interrupt event to wake up system and generate time-out reset system event while WDT time-out reset delay period expired.
----------------------------------	--

**Window Watchdog Timer (WWDT)**

<b>WWDT_CompareINT</b>	Show how to reload the WWDT counter value.
------------------------	--

**Real Timer Clock (RTC)**

<b>RTC_AlarmWakeup</b>	Use RTC alarm interrupt event to wake up system.
<b>RTC_TimeAndTick</b>	Show the current RTC data/time per tick.

**PWM Generator and Capture Timer (PWM)**

<b>PWM_Capture</b>	Capture the PWM1 Channel 0 waveform by PWM1 Channel 2.
<b>PWM_DeadZone</b>	Demonstrate how to use PWM Dead Zone function.
<b>PWM_DoubleBuffer_PeriodLoadingMode</b>	Change duty cycle and period of output waveform by PWM Double Buffer function (Period loading mode).
<b>PWM_DutySwitch</b>	Change duty cycle of output waveform by configured period.
<b>PWM_OutputWaveform</b>	Demonstrate how to use PWM output waveform.
<b>PWM_PDMA_Capture</b>	Capture the PWM1 Channel 0 waveform by PWM1 Channel 2, and use PDMA to transfer captured data.
<b>PWM_SyncStart</b>	Demonstrate how to use PWM counter synchronous

start function.

**UART Interface Controller (UART)**

<b>TIMER_ACMPTrigger</b>	Show how to use ACMP0 to trigger Timer capture event.
<b>TIMER_CaptureCounter</b>	Show how to use the timer2 capture function to capture timer2 counter value.
<b>TIMER_Delay</b>	Show how to use timer0 to create various delay time.
<b>TIMER_EventCounter</b>	Implement timer1 event counter function to count the external input event.
<b>TIMER_InterTimerTriggerMode</b>	Demonstrate how to use Inter-Timer trigger function.
<b>TIMER_PeriodicINT</b>	Implement timer counting in periodic mode.
<b>TIMER_PWM_Brake</b>	Generate Timer brake event by Timer brake pin.
<b>TIMER_PWM_ChangeDuty</b>	Change duty cycle and period of output waveform in PWM down count type.
<b>TIMER_PWM_DeadTime</b>	Demonstrate Timer PWM Complementary mode and Dead-Time function.
<b>TIMER_PWM_OuputWaveform</b>	Demonstrate output different duty waveform in Timer0~Timer3 PWM.
<b>TIMER_TimeoutWakeup</b>	Use timer0 periodic time-out interrupt event to wake up system.
<b>TIMER_ToggleOut</b>	Implement timer counting in toggle-output mode.

**Smart Card Host Interface (SC)**

<b>SCUART_TxRx</b>	Show Smartcard UART by connecting PA.0 and PA.1 pins.
<b>SC_ReadATR</b>	Read Smartcard ATR from the SC0 port.
<b>SC_ReadSimPhoneBook</b>	Read SIM phone book from the SC0 port.

**I<sup>2</sup>S Controller (I<sup>2</sup>S)**

<b>I2S_Master</b>	Configure SPI1 as I2S Master mode and demonstrate how I2S works in Master mode. This sample code needs to work with I2S_Slave.
<b>I2S_PDMA_NAU8822</b>	This is an I2S demo with PDMA function connected with NAU8822 codec.
<b>I2S_PDMA_Play</b>	This is an I2S demo for playing data and demonstrating how I2S works with PDMA.
<b>I2S_PDMA_PlayRecord</b>	This is an I2S demo for playing and recording data with PDMA function.
<b>I2S_PDMA_Record</b>	This is an I2S demo for recording data and demonstrating how I2S works with PDMA.
<b>I2S_Slave</b>	Configure SPI1 as I2S Slave mode and demonstrate how I2S works in Slave mode. This sample code needs to work with I2S_Master.

**Serial Peripheral Interface (SPI)**

<b>SPI_Loopback</b>	Implement SPI Master loop back transfer. This sample code needs to connect SPI0_MISO0 pin and SPI0_MOSI0 pin together. It will compare the received data with transmitted data.
<b>SPI_MasterFifoMode</b>	Configure SPI0 as Master mode and demonstrate how to communicate with an off-chip SPI Slave device with FIFO mode. This sample code needs to work with SPI_SlaveFifoMode.
<b>SPI_PDMA_LoopTest</b>	Demonstrate SPI data transfer with PDMA. SPI0 will be configured as Master mode and SPI1 will be configured as Slave mode. Both TX PDMA function and RX PDMA function will be enabled.
<b>SPI_SlaveFifoMode</b>	Configure SPI0 as Slave mode and demonstrate how to communicate with an off-chip SPI Master device with FIFO mode. This sample code needs to work with SPI_MasterFifoMode.

**I<sup>2</sup>C Serial Interface Controller (I<sup>2</sup>C)**

<b>I2C_EEPROM</b>	Demonstrate how to access EEPROM through a I2C interface
<b>I2C_GCMode_Master</b>	Demonstrate how a Master uses I2C address 0x0 to write data to I2C Slave. This sample code needs to work with I2C_GCMode_Slave.
<b>I2C_GCMode_Slave</b>	Demonstrate how to receive Master data in GC (General Call) mode. This sample code needs to work with I2C_GCMode_Master.
<b>I2C_Loopback</b>	Demonstrate how a Master accesses Slave.
<b>I2C_Master</b>	Demonstrate how a Master accesses Slave. This sample code needs to work with I2C_Slave.
<b>I2C_Master_PDMA</b>	Demonstrate how a Master accesses Slave using PDMA TX mode and PDMA RX mode.
<b>I2C_Slave</b>	Demonstrate how to set I2C in slave mode to receive the data from a Master. This sample code needs to work with I2C_Master.
<b>I2C_Slave_PDMA</b>	Demonstrate how a Slave uses PDMA Rx mode receive data from a Master.
<b>I2C_Wakeup_Slave</b>	Demonstrate how to set I2C to wake up MCU from Power-down mode. This sample code needs to work with I2C_Master.

**Universal Serial Control Interface Controller - UART Mode (USCI-UART)**

<b>USCI_UART_AutoBaudRate_Master</b>	Show how to use auto baud rate detection function. This sample code needs to work with USCI_UART_AutoBaudRate_Slave.
<b>USCI_UART_AutoBaudRate_Slave</b>	Show how to use auto baud rate detection function. This sample code needs to work with USCI_UART_AutoBaudRate_Master.
<b>USCI_UART_Autoflow_Master</b>	Transmit and receive data with auto flow control. This sample code needs to work with

	USCI_UART_AutoFlow_Slave.
<b>USCI_UART_AutoFlow_Slave</b>	Transmit and receive data with auto flow control. This sample code needs to work with USCI_UART_AutoFlow_Master.
<b>USCI_UART_RS485_Master</b>	Transmit and receive data in RS485 mode. This sample code needs to work with USCI_UART_RS485_Slave.
<b>USCI_UART_RS485_Slave</b>	Transmit and receive data in RS485 mode. This sample code needs to work with USCI_UART_RS485_Master.
<b>USCI_UART_TxRxFuntion</b>	Transmit and receive data from PC terminal through a RS232 interface.
<b>USCI_UART_Wakeup</b>	Show how to wake up system from Power-down mode by USCI interrupt in UART mode.

**Universal Serial Control Interface Controller - SPI Mode (USCI-SPI)**

<b>USCI_SPI_Loopback</b>	Implement USCI_SPI1 Master loop back transfer. This sample code needs to connect USCI_SPI1_MISO pin and USCI_SPI1_MOSI pin together. It will compare the received data with transmitted data.
<b>USCI_SPI_MasterMode</b>	Configure USCI_SPI1 as Master mode and demonstrate how to communicate with an off-chip SPI Slave device. This sample code needs to work with USCI_SPI_SlaveMode.
<b>USCI_SPI_SlaveMode</b>	Configure USCI_SPI1 as Slave mode and demonstrate how to communicate with an off-chip SPI Master device. This sample code needs to work with USCI_SPI_MasterMode.

**Universal Serial Control Interface Controller - I<sup>2</sup>C Mode (USCI-I2C)**

<b>USCI_I2C_EEPROM</b>	Demonstrate how to access EEPROM through a USCI_I2C interface.
<b>USCI_I2C_Loopback</b>	Demonstrate how a Master accesses Slave.
<b>USCI_I2C_Loopback_10bit</b>	Demonstrate how a Master uses 10-bit addressing

	access Slave.
<b>USCI_I2C_Master</b>	Demonstrate how a Master access Slave. This sample code needs to work with I2C_Slave.
<b>USCI_I2C_Master_10bit</b>	Demonstrate how a Master use 10-bit addressing access Slave. This sample code needs to work with I2C_Slave.
<b>USCI_I2C_Slave</b>	Demonstrate how to set I2C in slave mode to receive the data from a Master. This sample code needs to work with I2C_Master.
<b>USCI_I2C_Slave_10bit</b>	Demonstrate how to set I2C in 10-bit addressing slave mode to receive the data from a Master. This sample code needs to work with I2C_Master.
<b>USCI_I2C_Wakeup_Slave</b>	Demonstrate how to set I2C to wake up MCU from Power-down mode. This sample code needs to work with I2C_Master.

**External Bus Interface (EBI)**

<b>EBI_NOR</b>	Configure EBI interface to access MX29LV320T (NOR Flash) on EBI interface.
<b>EBI_SRAM</b>	Configure EBI interface to access BS616LV4017 (SRAM) on EBI interface.

**CRC Controller (CRC)**

<b>CRC_CCITT</b>	Implement CRC in CRC-CCITT mode and get CRC checksum results.
<b>CRC_CRC32</b>	Implement CRC in CRC-32 mode with PDMA transfer.
<b>CRC_CRC8</b>	Implement CRC in CRC-8 mode and get CRC checksum results.

**Analog-to-Digital Converter (ADC)**

<b>ADC_ContinuousScanMode</b>	Perform A/D Conversion with ADC continuous scan
-------------------------------	---

	mode.
<b>ADC_PDMA_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan mode and transfer result by PDMA.
<b>ADC_PwmTrigger</b>	Demonstrate how to trigger ADC by PWM.
<b>ADC_ResultMonitor</b>	Monitor the conversion result of Channel 2 by the digital compare function.
<b>ADC_SingleCycleScanMode</b>	Perform A/D Conversion with ADC single cycle scan mode.
<b>ADC_SingleMode</b>	Perform A/D Conversion with ADC single mode.

**Analog Comparator Controller (ACMP)**

<b>ACMP</b>	Demonstrate how Analog Comparator (ACMP) works with internal band-gap voltage.
<b>ACMP_Wakeup</b>	Show how to wake up MCU from Power-down mode by ACMP wake-up function.

**Hardware Divider (HDIV)**

<b>HDIV</b>	Show how to use divider API and how to use hardware divider by control registers.
-------------	---

### **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.*