ARM Cortex®-M

32位微控制器

NuMicro™ Family
NUC230/240 系列
技术参考手册

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

www.nuvoton.com

目录

图集

表集

概述

NuMicro™ NUC200系列32位微控制器内嵌ARM® Cortex™-M0内核，拥有与传统8051单片机相匹敌的价格，适用于工业控制和需要多种通信接口的应用领域。NuMicro™ NUC200系列包括NUC200，NUC220，NUC230及NUC240产品线

NuMicro™ NUC230 CAN产品线内嵌Cortex™-M0内核，最高可运行72MHz，拥有32K/64K/128K字节flash存储器，8K/16K字节 SRAM，以及用来存储升级代码的8K LDROM，另外还有丰富的外设接口，例如：定时器，看门狗，窗口式看门狗，RTC，具有CRC计算单元的PDMA，UART，SPI， I^2C， I^2S，PWM 定时器，GPIO，LIN总线， CAN总线，PS/2，智能卡，12位ADC，模拟比较器，低压复位，掉电侦测

NuMicro™ NUC240 Connectivity产品线内建USB2.0全速设备控制器，CAN总线控制器，最高可运行72M的Cortex™-M0内核，32K/64K/128K字节flash存储器，8K/16K字节SRAM, 用来存储升级代码的8K LDROM，同样也集成了丰富的外设接口例如：定时器，看门狗，窗口式看门狗，RTC，具有CRC计算单元的PDMA，UART，SPI， I^2C， I^2S，PWM 定时器，GPIO，LIN总线， CAN总线，PS/2，智能卡，12位ADC，模拟比较器，低压复位，掉电侦测

产品线	UART	SPI	I^2C	USB	LIN	CAN	PS/2	I^2S	SC
NUC230	•	•	•		•	•	•	•	•
NUC240	•	•	•	•	•	•	•	•	•

表 0-1 NuMicro™ NUC200 系列支持的接口列表

1 特性

器件的功能依赖于产品线及该产品线下的具体型号

1.1 NuMicro™ NUC230 特性 – Automotive Line

- ARM® Cortex™-M0 内核
  - 最高可运行到 72 MHz
  - 一个 24位系统时钟
  - 支持低功耗掉电模式
  - 单周期32位硬件乘法器
  - 可嵌套向量中断控制器（NVIC）用于控制32个中断源，每个中断有4种优先级
  - 串行调试接口支持2个观察点/4个中断点
- 内建LDO,支持从2.5V到5.5V的宽电压操作
- Flash存储器
  - 32K/64K/128K 字节flash存储器用来存储程序代码
  - 8 KB flash 存储器用来存储ISP升级引导代码
  - 支持在系统编程(ISP)和在应用编程(IAP)升级代码
  - 支持512字节页擦除
  - 在128K字节系统中可配置数据flash的起始地址和大小，32K/64K字节系统中固定4K字节数据falsh
  - 通过SWD/ICE接口，支持2线ICP升级
  - 支持外部编程器并行高速编程模式
- SRAM 存储器
  - 8K/16K 字节内嵌SRAM
  - 支持PDMA 模式
- PDMA (外设 DMA)
  - 9通道PDMA支持外设和内存间的自动数据传输
  - 支持4种通用多项式的CRC计算， CRC-CCITT, CRC-8, CRC-16 and CRC-32
- 时钟控制
  - 针对不同应用可灵活选择时钟
  - 内置22.1184 MH高速振荡器可用于系统运行
    - 精度范围 ±1 % （ +25 ℃ ， V_{DD} = 5 V）
    - 精度范围 ±3 % （ -40 ℃ ~ +105 ℃ ， V_{DD} = 2.5 V ~ 5.5 V）
  - 内置 10 kHz 低速振荡器用于看门狗及掉电唤醒等功能
  - 支持一组 PLL, 高至 72 MHz, 用于高性能的系统运行
  - 外部 4~24 MHz 高速晶振用于精准的时序操作
  - 外部 32.768 kHz 低速晶振用于RTC及低功耗操作
- GPIO
  - 四种I/O模式:
    - 准双向模式
    - 推挽输出模式
    - 开漏输出模式
    - 高阻输入模式
  - 可配置TTL/Schmitt 触发输入
  - I/O管脚可配置为边沿/电平触发模式的中断源
- 定时器
  - 支持4组32位定时器，每个定时器包括一个24位向上计数器和一个8位预分频器
  - 每个定时器都有独立的时钟源
  - 提供 one-shot, periodic, toggle 和 continuous counting 操作模式
  - 支持事件计数功能

- 支持输入捕获功能

- 看门狗定时器
  - 多个时钟源选择
  - 8个可选的时间溢出周期，从1.6毫秒~26秒（取决于时钟源的选择）
  - 可用作掉电模式的唤醒
  - 看门狗溢出事件可以触发中断或者复位芯片
  - 支持4种看门狗复位延时 (1026, 130, 18 或 3 个看门狗时钟周期)
- 窗口看门狗
  - 6-位向下计数器搭配11位预分频器，用作宽范围的窗口选择
- RTC
  - 支持软件补偿功能寄存器FCR
  - 支持RTC计数 (秒, 分, 时) 及万年历 (日, 月, 年)
  - 支持闹铃寄存器（秒，分，时，日，月，年）
  - 可选12小时或24小时制
  - 自动闰年识别
  - 支持定时滴答中断，有8个可选周期1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 及1 秒
  - 支持电池单独供电脚 (VBAT)
  - 支持唤醒功能
- PWM/捕捉
  - 内建4个16位的PWM产生器，提供8路PWM输出或组成4对互补的PWM输出
  - 每个PWM产生器都配有一个时钟源选择器，一个除频器，一个8位时钟预分频器以及一个死区发生器
  - 支持单次触发模式或者自动重载模式
  - 8路16位捕捉定时器（共享PWM定时器），提供8路上升/下降沿的捕捉输入
  - 支持捕捉中断
- UART
  - 最多6路UART控制器（3路UART复用SC接口）
  - UART支持流量控制(TXD, RXD, nCTS 和 nRTS)
  - UART0 有64字节的FIFO,用于高速传输
  - UART1/2有16字节FIFO用于标准传输
  - 支持 IrDA (SIR) 及 LIN 总线功能
  - 支持9位RS-485及方向控制功能
  - 可编程波特率产生器，高至1/16系统时钟
  - 支持CST唤醒功能(UART0 ，UART1 支持)
  - 支持 PDMA 模式
- 智能卡主机 (SC)
  - 支持最多3路ISO-7816-3 接口
    - 兼容 ISO-7816-3 T=0, T=1
    - 收/发都有4字节FIFO
    - 可编程传输时钟频率
    - 可编程设置接收FIFO触发门槛
    - 可编程保证时间选择(11 ETU ~ 266 ETU)
    - 1个24位，2个8位超时计数器用于回复请求（ATR）及等待时间处理
    - 支持自动逆转换功能
    - 支持收发器错误重试及错误限制功能
    - 支持硬件激活时序处理
    - 支持硬件热复位时序处理
    - 支持硬件释放时序处理
    - 检测到卡移除时，支持硬件自动释放时序
  - 支持3路UART功能端口

- 全双工异步通信
- 支持4字节收/发FIFO
- 可编程设定波特率产生器
- 可编程设置奇偶校验/无校验产生及检测
- 可编程设置1位或2位停止位设定

- SPI
  - 多至4路SPI 控制器
  - SPI主时钟频率高至36MHz（5V工作电压）
  - SPI从时钟频率高至18MHz（5V工作电压）
  - 支持SPI主/从机模式
  - 全双工同步串行数据传输
  - 传输数据长度8到32位可编程设定
  - 可设定MSB 或 LSB 在前
  - 在时钟上升沿还是下降沿收/发数据可独立配置
  - 主机模式有两条从机/设备选择线，从机模式有一条从机/设备选择线
  - 支持32位传输模式下字节睡眠
  - 支持PDMA功能
  - 支持三线，无从机信号，双向接口
- I^2C
  - 最多2组I^2C 控制器
  - 支持主/从机模式
  - 主从机间双向数据传输
  - 多主机总线（无中心主机）
  - 总线仲裁，可避免主机同时传输数据时的冲突
  - 串行时钟的同步机制，用一条总线来实现设备间各种速度下的通讯
  - 串行时钟同步可作为握手机制，控制总线上数据的传输及暂停
  - 可编程时钟适用于各种波特率控制
  - 支持多地址识别（4个带屏蔽功能的从机地址）
  - 支持唤醒功能
- I^2S
  - 与外部音频编解码器的接口
  - 支持主/从机模式
  - 可处理8,16,24和32位长度数据
  - 支持单声道和立体声音频数据
  - 支持I^2S 和MSB对齐数据格式
  - 提供两个8字长度的FIFO缓冲区，分别给收发数据使用
  - 缓冲区数据超过设定的门槛时会产生中断
  - 支持两路DMA分别用于收发数据
- PS/2 设备
  - 支持主机通信禁止和请求发送检测
  - 接收帧错误检测
  - 可编程设定1-16字节发送缓冲以降低CPU负担
  - 数据接收支持双缓冲
  - 软件重写总线
- CAN 2.0
  - 支持CAN协议2.0 A B部分
  - 比特率高至1M
  - 32个报文对象
  - 每个信息对象有自己的标识掩码
  - 可编程FIFO模式（报文级联）
  - 可屏蔽中断

  - 时间触发的CAN应用中禁用自动重传模式
  - 支持唤醒功能
- ADC
  - 12位SAR ADC快至760KSPS(工作电压5V)
  - 多至8通道单端输入或4通道差分输入
  - 支持单次/单周期扫描/连续扫描模式
  - 每个通道都有独立的结果寄存器
  - 只对使能的通道扫描
  - 阈电压检测
  - 软件编程，外部引脚以及PWM中央对齐可以触发ADC开始转换
  - 支持PDMA模式
- 模拟比较器
  - 多至两个模拟比较器
  - 负管脚可以外部输入信号也可以选择内部Band-gap电压
  - 比较结果改变产生中断
  - 支持掉电唤醒
- EBI(外部总线接口)
  - 可访问的空间: 8位模式为64KB或16位模式为128KB
  - 支持8bit/16bit数据宽度
  - 在16-位数据宽度模式下支持字节写入
- 96 位唯一ID(UID)
- 128 位唯一客户ID (UCID)
- 一个内置温度传感器，精度1℃
- 掉电检测
  - 有 4 个等级: 4.4 V/3.7 V/2.7 V/2.2 V
  - 支持掉电中断或复位功能
- 低压复位
  - 复位门槛电压: 2.0 V
- 操作温度: -40℃ ~ 105℃
- 封装:
  - 无铅封装(RoHS)
  - LQFP 100-pin / 64-pin / 48-pin

1.2 NuMicro™ NUC240 特性 – Connectivity Line

- ARM® Cortex™-M0 内核
  - 最高可运行到 72 MHz
  - 一个 24位系统时钟
  - 支持低功耗掉电模式
  - 单周期32位硬件乘法器
  - 可嵌套向量中断控制器（NVIC）用于控制32个中断源，每个中断有4种优先级
  - 串行调试接口支持2个观察点/4个中断点
- 内建LDO,支持从2.5V到5.5V的宽电压操作
- Flash存储器
  - 32K/64K/128K 字节flash存储器用来存储程序代码
  - 8 KB flash 存储器用来存储ISP升级引导代码
  - 支持在系统编程(ISP)和在应用编程(IAP)升级代码
  - 支持512字节页擦除
  - 在128K字节系统中可配置数据flash的起始地址和大小，32K/64K字节系统中固定4K字节数据falsh
  - 通过SWD/ICE接口，支持2线ICP升级
  - 支持外部编程器并行高速编程模式
- SRAM 存储器
  - 8K/16K 字节内嵌SRAM
  - 支持PDMA 模式
- PDMA (外设 DMA)
  - 9通道PDMA支持外设和内存间的自动数据传输
  - 支持4种通用多项式的CRC计算， CRC-CCITT, CRC-8, CRC-16 and CRC-32
- 时钟控制
  - 针对不同应用可灵活选择时钟
  - 内置22.1184 MH高速振荡器可用于系统运行
    - 精度范围 ±1 % （ +25 ℃ ， V_{DD} = 5 V）
    - 精度范围 ±3 % （ -40 ℃ ~ +105 ℃ ，V_{DD} = 2.5 V ~ 5.5 V）
  - 内置 10 kHz 低速振荡器用于看门狗及掉电唤醒等功能
  - 支持一组 PLL, 高至 72 MHz, 用于高性能的系统运行
  - 外部 4~24 MHz 高速晶振用于精准的时序操作
  - 外部 32.768 kHz 低速晶振用于RTC及低功耗操作
- GPIO
  - 四种I/O模式:
    - 准双向模式
    - 推挽输出模式
    - 开漏输出模式
    - 高阻输入模式
  - 可配置TTL/Schmitt 触发输入
  - I/O管脚可配置为边沿/电平触发模式的中断源
- 定时器
  - 支持4组32位定时器，每个定时器包括一个24位向上计数器和一个8位预分频器
  - 每个定时器都有独立的时钟源
  - 提供 one-shot, periodic, toggle 和 continuous counting 操作模式
  - 支持事件计数功能
  - 支持输入捕获功能
- 看门狗定时器
  - 多个时钟源选择
  - 8个可选的时间溢出周期，从1.6毫秒~26秒（取决于时钟源的选择）

- 可用作掉电模式的唤醒
- 看门狗溢出事件可以触发中断或者复位芯片
- 支持4种看门狗复位延时 (1026, 130, 18 或 3 个看门狗时钟周期)

- 窗口看门狗
  - 6-位向下计数器搭配11位预分频器，构成宽范围的窗口选择
- RTC
  - 支持软件补偿功能寄存器FCR
  - 支持RTC计数 (秒，分，时) 及万年历 (日，月，年)
  - 支持闹铃寄存器（秒，分，时，日，月，年）
  - 可选12小时或24小时制
  - 自动闰年识别
  - 支持定时滴答中断，有8个可选周期1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 及1 秒
  - 支持电池单独供电脚 (VBAT)
  - 支持唤醒功能
- PWM/捕捉
  - 内建4个16位的PWM产生器，提供8路PWM输出或组成4对互补的PWM输出
  - 每个PWM产生器都配有一个时钟源选择器，一个除频器，一个8位时钟预分频器以及一个死区发生器
  - 支持单次触发模式或者自动重载模式
  - 8路16位捕捉定时器（共享PWM定时器），提供8路上升/下降沿的捕捉输入
  - 支持捕捉中断
- UART
  - 最多6路UART控制器（3路UART复用SC接口）
  - UART支持流量控制(TXD, RXD, nCTS 和 nRTS)
  - UART0 有64字节的FIFO,用于高速传输
  - UART1/2有16字节FIFO用于标准传输
  - 支持 IrDA (SIR) 及 LIN 总线功能
  - 支持9位RS-485及方向控制功能
  - 可编程波特率产生器，高至1/16系统时钟
  - 支持CST唤醒功能(UART0 ，UART1 支持)
  - 支持 PDMA 模式
- 智能卡主机 (SC)
  - 支持最多3路ISO-7816-3 接口
    - 兼容 ISO-7816-3 T=0, T=1
    - 收/发都有4字节FIFO
    - 可编程传输时钟频率
    - 可编程设置接收FIFO触发门槛
    - 可编程保证时间选择(11 ETU ~ 266 ETU)
    - 1个24位，2个8位超时计数器用于回复请求（ATR）及等待时间处理
    - 支持自动逆转换功能
    - 支持收发器错误重试及错误限制功能
    - 支持硬件激活时序处理
    - 支持硬件热复位时序处理
    - 支持硬件释放时序处理
    - 检测到卡移除时，支持硬件自动释放时序
  - 支持3路UART功能端口
    - 全双工异步通信
    - 支持4字节收/发FIFO
    - 可编程设定波特率产生器
    - 可编程设置奇偶校验/无校验产生及检测

- 可编程设置1位或2位停止位设定

- SPI
  - 多至4路SPI 控制器
  - SPI主时钟频率高至36MHz（5V工作电压）
  - SPI从时钟频率高至18MHz（5V工作电压）
  - 支持SPI主/从机模式
  - 全双工同步串行数据传输
  - 传输数据长度8到32位可编程设定
  - 可设定MSB 或 LSB 在前
  - 在时钟上升沿还是下降沿收/发数据可独立配置
  - 主机模式有两条从机/设备选择线，从机模式有一条从机/设备选择线
  - 支持32位传输模式下字节睡眠
  - 支持PDMA功能
  - 支持三线，无从机信号，双向接口
- I^2C
  - 最多2组I^2C 控制器
  - 支持主/从机模式
  - 主从机间双向数据传输
  - 多主机总线（无中心主机）
  - 总线仲裁，可避免主机同时传输数据时的冲突
  - 串行时钟的同步机制，可实现总线上不同波特率设备间用同一条总线通信
  - 串行时钟同步可作为握手机制，控制总线上数据的传输及暂停
  - 可编程时钟适用于各种波特率控制
  - 支持多地址识别（4个带屏蔽功能的从机地址）
  - 支持唤醒功能
- I2S
  - 与外部音频编解码器的接口
  - 支持主/从机模式
  - 可处理8,16,24和32位长度数据
  - 支持单声道和立体声音频数据
  - 支持I^2S 和MSB对齐数据格式
  - 提供两个8字长度的FIFO缓冲区，分别给收发数据使用
  - 缓冲区数据超过设定的门槛时会产生中断
  - 支持两路DMA分别用于收发数据
- PS/2 设备
  - 支持主机通信禁止和请求发送检测
  - 接收帧错误检测
  - 可编程设定1-16字节发送缓冲以降低CPU负担
  - 数据接收支持双缓冲
  - 软件重写总线
- CAN 2.0
  - 支持CAN协议2.0 A B部分
  - 比特率高至1M
  - 32个报文对象
  - 每个信息对象有自己的标识掩码
  - 可编程FIFO模式（报文级联）
  - 可屏蔽中断
  - 时间触发的CAN应用中禁用自动重传模式
  - 支持唤醒功能
  -
- USB 2.0 全速设备

  - USB2.0全速设备控制器，最高速率12M
  - 内置USB收发器
  - 提供1个中断源4个中断事件
  - 支持控制，批量，中断，同步传输模式
  - 总线空闲3ms自动挂起
  - 提供8个可编程端点
  - 内置512字节SRAM用于USB数据传输缓存
  - 支持远程唤醒功能
- ADC
  - 12位SAR ADC快至760KSPS(工作电压5V)
  - 多至8通道单端输入或4通道差分输入
  - 支持单次/单周期扫描/连续扫描模式
  - 每个通道都有独立的结果寄存器
  - 只对使能的通道扫描
  - 阈电压检测
  - 软件编程，外部引脚以及PWM中央对齐可以触发ADC开始转换
  - 支持PDMA模式
- 模拟比较器
  - 多至两个模拟比较器
  - 负管脚可以外部输入信号也可以选择内部Band-gap电压
  - 比较结果改变产生中断
  - 支持掉电唤醒
- EBI(外部总线接口)
  - 可访问的空间: 8位模式为64KB或16位模式为128KB
  - 支持8bit/16bit数据宽度
  - 在16-位数据宽度模式下支持字节写入
- 96 位唯一ID(UID)
- 128 位唯一客户ID (UCID)
- 一个内置温度传感器，精度1℃
- 掉电检测
  - 有 4 个等级: 4.4 V/3.7 V/2.7 V/2.2 V
  - 支持掉电中断或复位功能
- 低压复位
  - 复位门槛电压: 2.0 V
- 操作温度: -40℃ ~ 105℃
- 封装:
  - 无铅封装(RoHS)
  - LQFP 100-pin / 64-pin / 48-pin

2 缩写表

缩写	描述
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
DAP	Debug Access Port
DES	Data Encryption Standard
EBI	External Bus Interface
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	22.1184 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
QEI	Quadrature Encoder Interface
SDIO	Secure Digital Input/Output
SPI	Serial Peripheral Interface

SPS	Samples per Second
TDES	Triple Data Encryption Standard
TMR	Timer Controller
UART	Universal Asynchronous Receiver/Transmitter
UCID	Unique Customer ID
USB	Universal Serial Bus
WDT	Watchdog Timer
WWDT	Window Watchdog Timer

表 2-1 缩写表

3 编号信息列表与管脚定义

3.1 NuMicro™ NUC230/240xxxAE 产品选型指南

3.1.1 NuMicro™ NUC230 Automotive Line 选型指南

Part Number	APROM (KB)	RAM (KB)	Data Flash (KB)	ISP ROM (KB)	I/O	Timer (32-Bit)	Connectivity						I²S	SC	Comp.	PWM	ADC (12-Bit)	RTC	EBI	ISP/ICP/IAP	Package
							UART	SPI	I²C	USB	LIN	CAN									
NUC230LC2AE	32	8	4	8	35	4	5	1	2	-	3	2	1	2	1	4	7	v	-	v	LQFP48
NUC230LD2AE	64	8	4	8	35	4	5	1	2	-	3	2	1	2	1	4	7	v	-	v	LQFP48
NUC230LE3AE	128	16	Config.	8	35	4	5	1	2	-	3	2	1	2	1	4	7	v	-	v	LQFP48
NUC230SC2AE	32	8	4	8	49	4	5	2	2	-	3	2	1	2	2	6	7	v	v	v	LQFP64
NUC230SD2AE	64	8	4	8	49	4	5	2	2	-	3	2	1	2	2	6	7	v	v	v	LQFP64
NUC230SE3AE	128	16	Config.	8	49	4	5	2	2	-	3	2	1	2	2	6	7	v	v	v	LQFP64
NUC230VE3AE	128	16	Config.	8	83	4	6	4	2	-	3	2	1	3	2	8	8	v	v	v	LQFP100

3.1.2 NuMicro™ NUC240 Connectivity Line 选型指南

Part Number	APROM (KB)	RAM (KB)	Data Flash (KB)	ISP ROM (KB)	I/O	Timer (32-Bit)	Connectivity						I²S	SC	Comp.	PWM	ADC (12-Bit)	RTC	EBI	ISP/ICP/IAP	Package
							UART	SPI	I²C	USB	LIN	CAN									
NUC240LC2AE	32	8	4	8	31	4	4	1	2	1	2	2	1	1	1	4	7	v	-	v	LQFP48
NUC240LD2AE	64	8	4	8	31	4	4	1	2	1	2	2	1	1	1	4	7	v	-	v	LQFP48
NUC240LE3AE	128	16	Config.	8	31	4	4	1	2	1	2	2	1	1	1	4	7	v	-	v	LQFP48
NUC240SC2AE	32	8	4	8	45	4	5	2	2	1	3	2	1	2	2	4	7	v	v	v	LQFP64
NUC240SD2AE	64	8	4	8	45	4	5	2	2	1	3	2	1	2	2	4	7	v	v	v	LQFP64
NUC240SE3AE	128	16	Config.	8	45	4	5	2	2	1	3	2	1	2	2	4	7	v	v	v	LQFP64
NUC240VE3AE	128	16	Config.	8	79	4	6	4	2	1	3	2	1	3	2	8	8	v	v	v	LQFP100

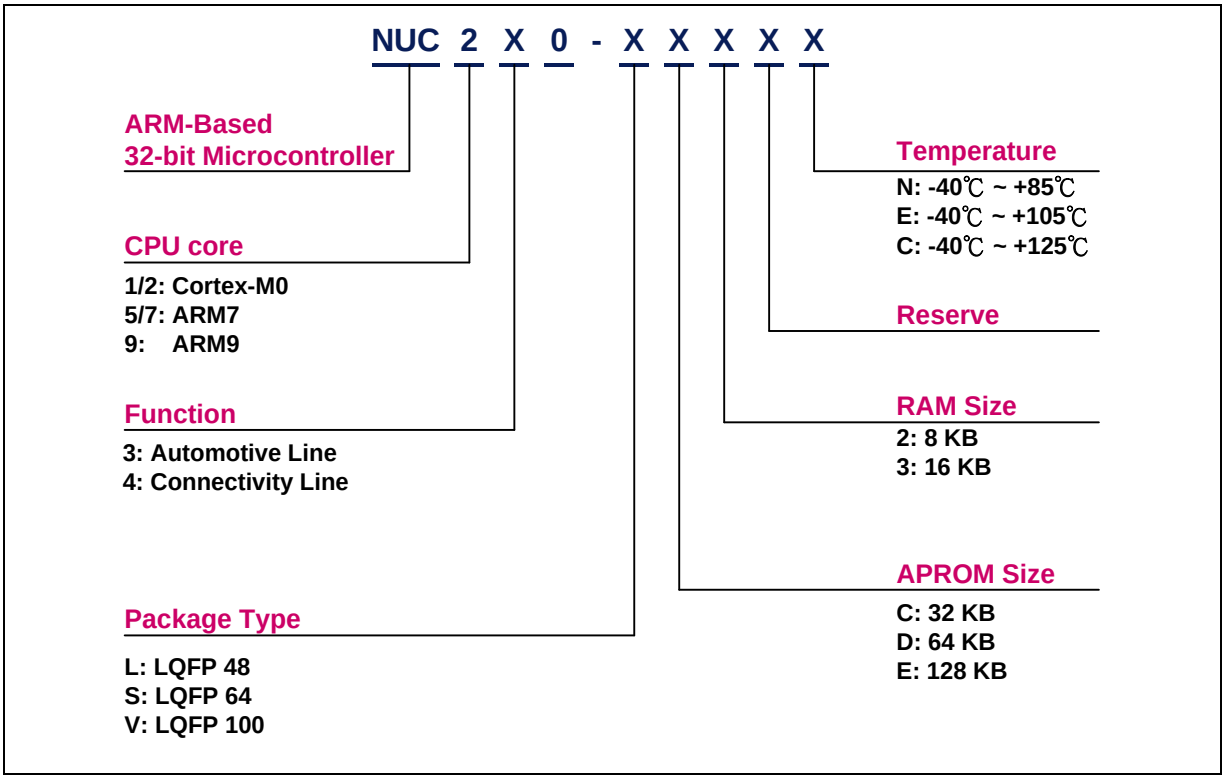


图 3-1 NuMicro™ NUC200 系列选型代码

3.2 管脚配置

3.2.1 NuMicro™ NUC230 管脚图

3.2.1.1 *NuMicro™ NUC230VxxAE LQFP 100 pin*

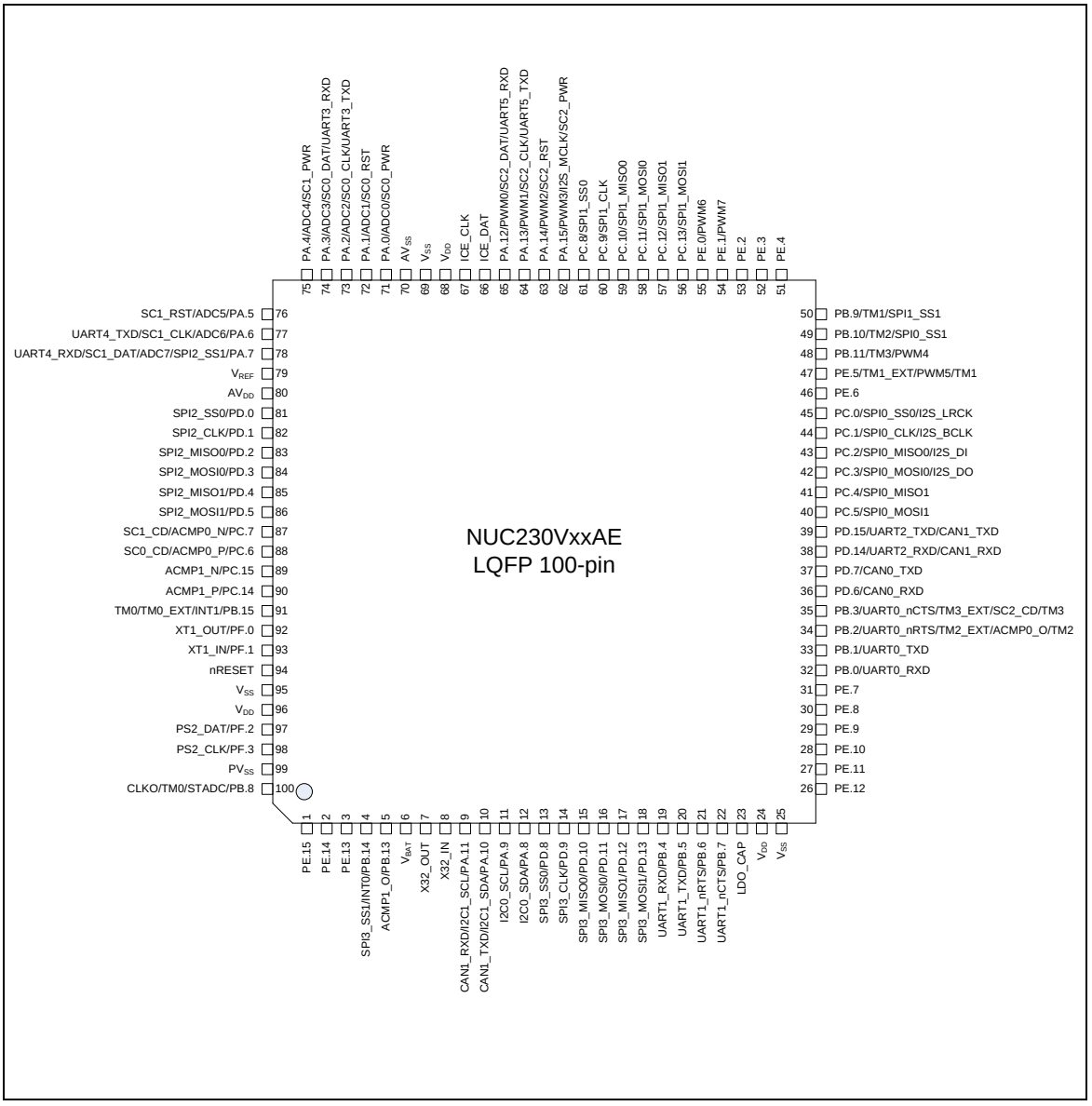


图 3-2 NuMicro™ NUC230VxxAE LQFP 100-pin 管脚图

3.2.1.2 *NuMicro™ NUC230SxxAE LQFP 64 pin*

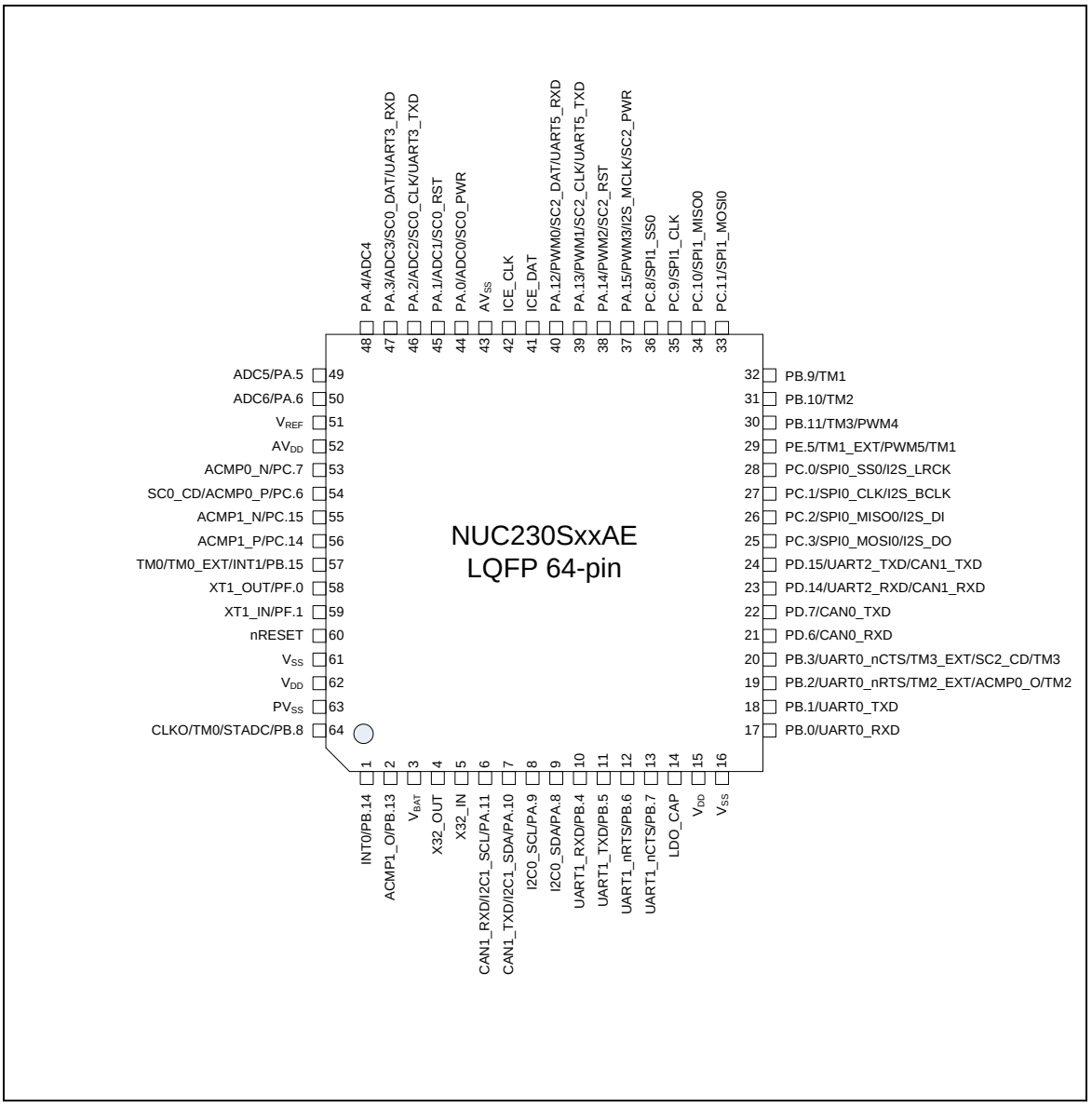


图 3-3 NuMicro™ NUC230SxxAE LQFP 64-pin 管脚图

3.2.1.3 *NuMicro™ NUC230LxxAE LQFP 48 pin*

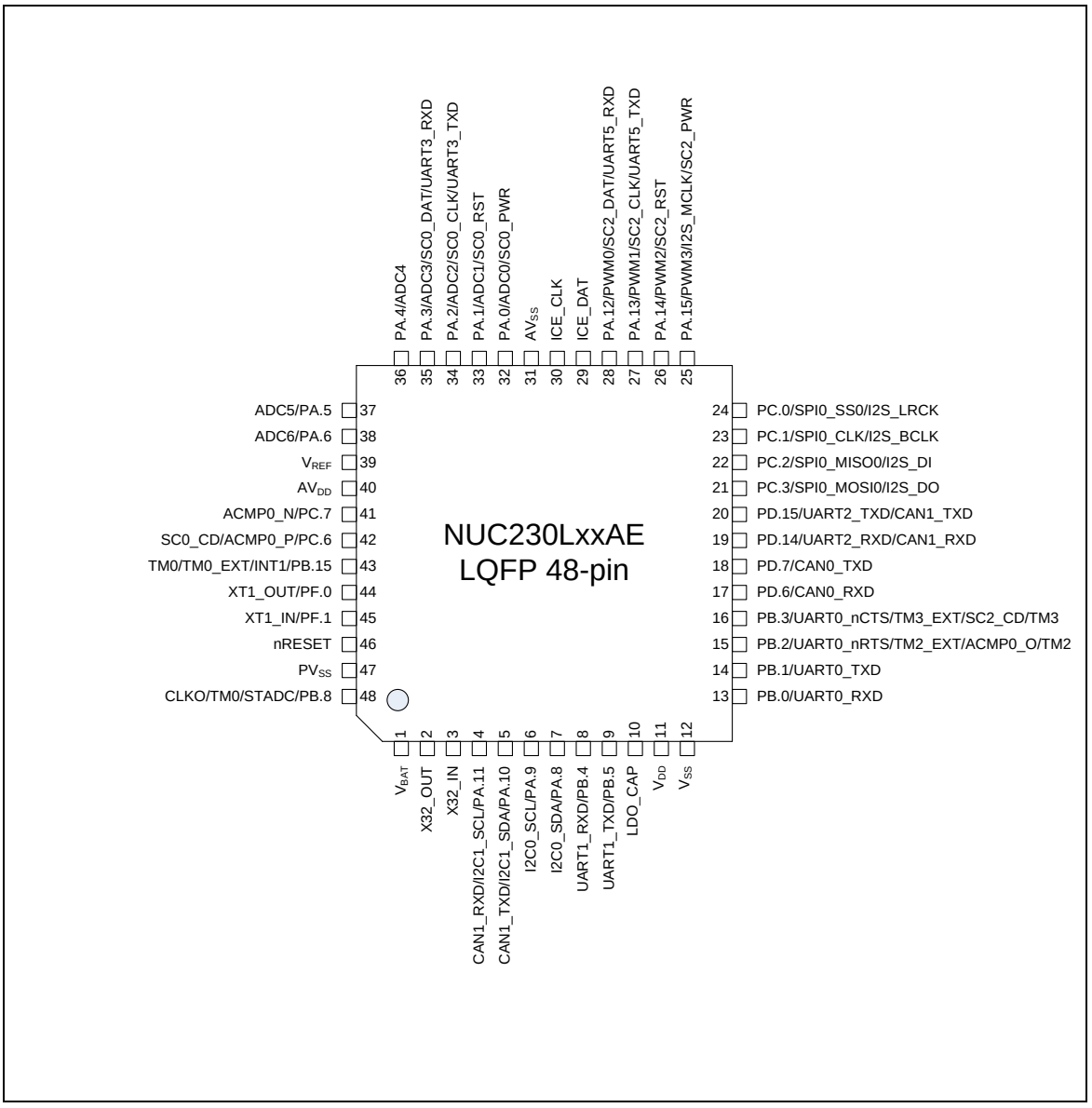


图 3-4 NuMicro™ NUC230LxxAE LQFP 48-pin 管脚图

3.2.2 NuMicro™ NUC240 管脚图

3.2.2.1 *NuMicro™ NUC240VxxAE LQFP 100 pin*

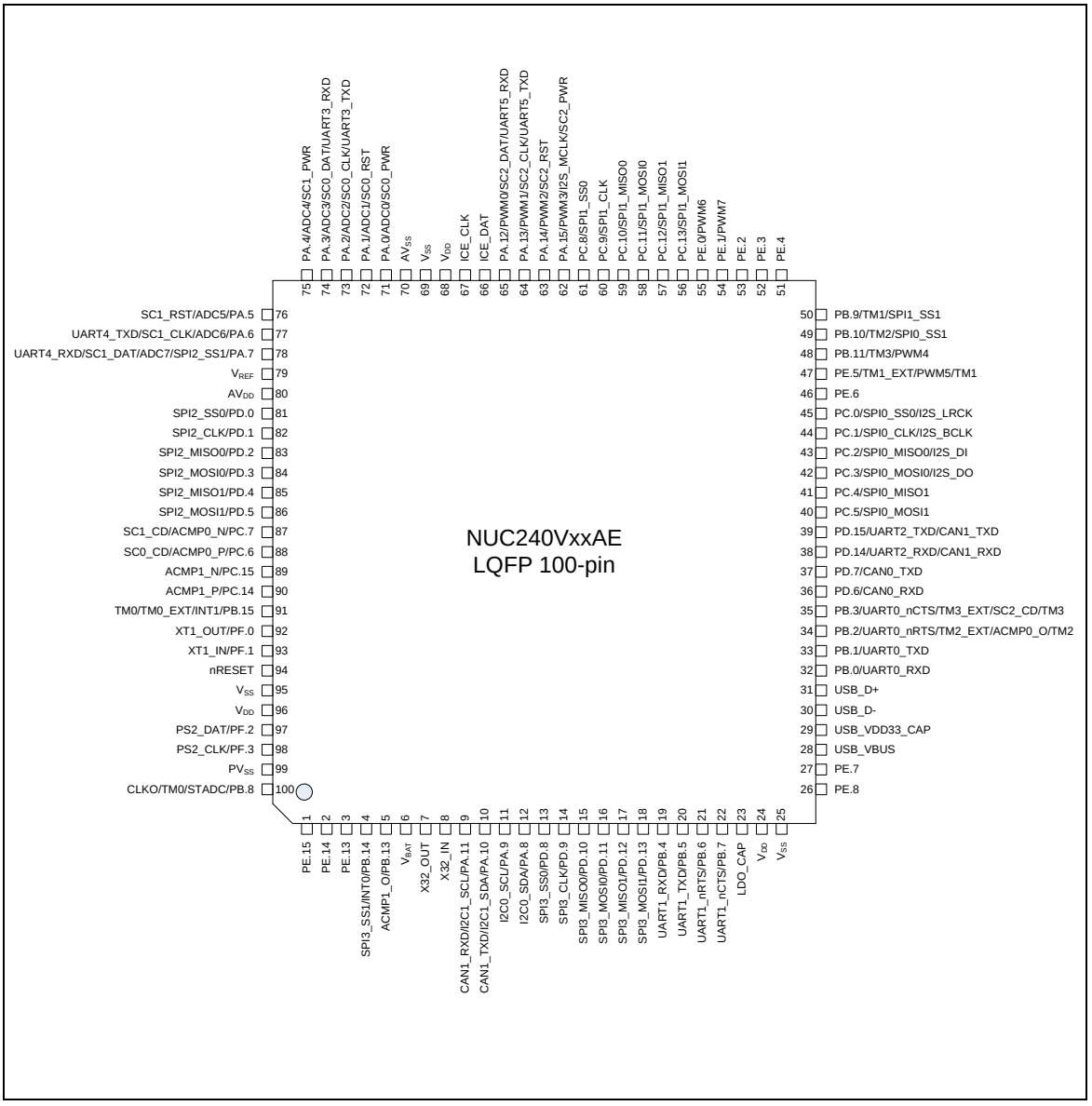


图 3-5 NuMicro™ NUC240VxxAE LQFP 100-pin 管脚图

3.2.2.2 *NuMicro™ NUC240SxxAE LQFP 64 pin*

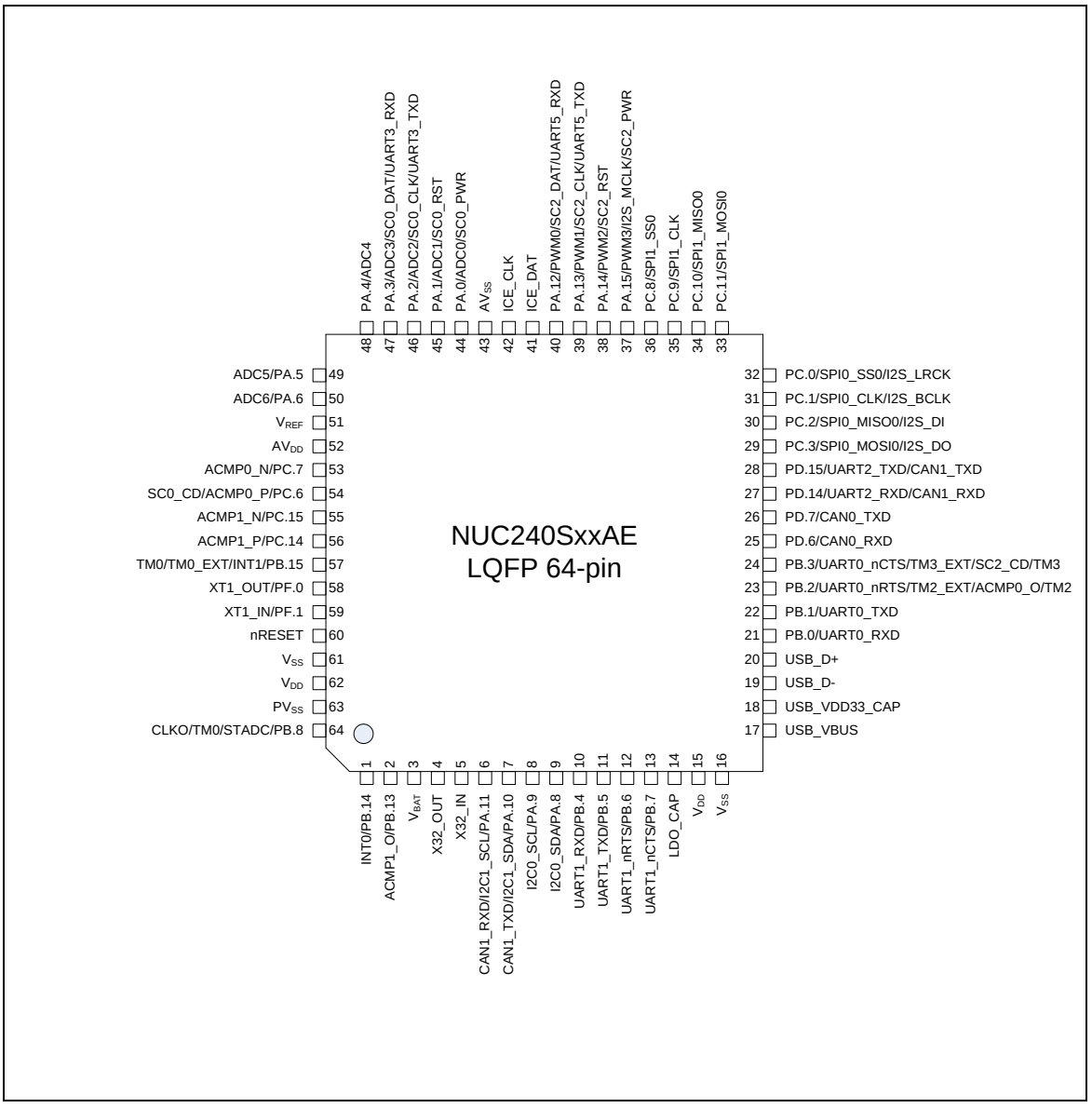


图 3-6 NuMicro™ NUC240SxxAE LQFP 64-pin 管脚图

3.2.2.3 *NuMicro™ NUC240LxxAE LQFP 48 pin*

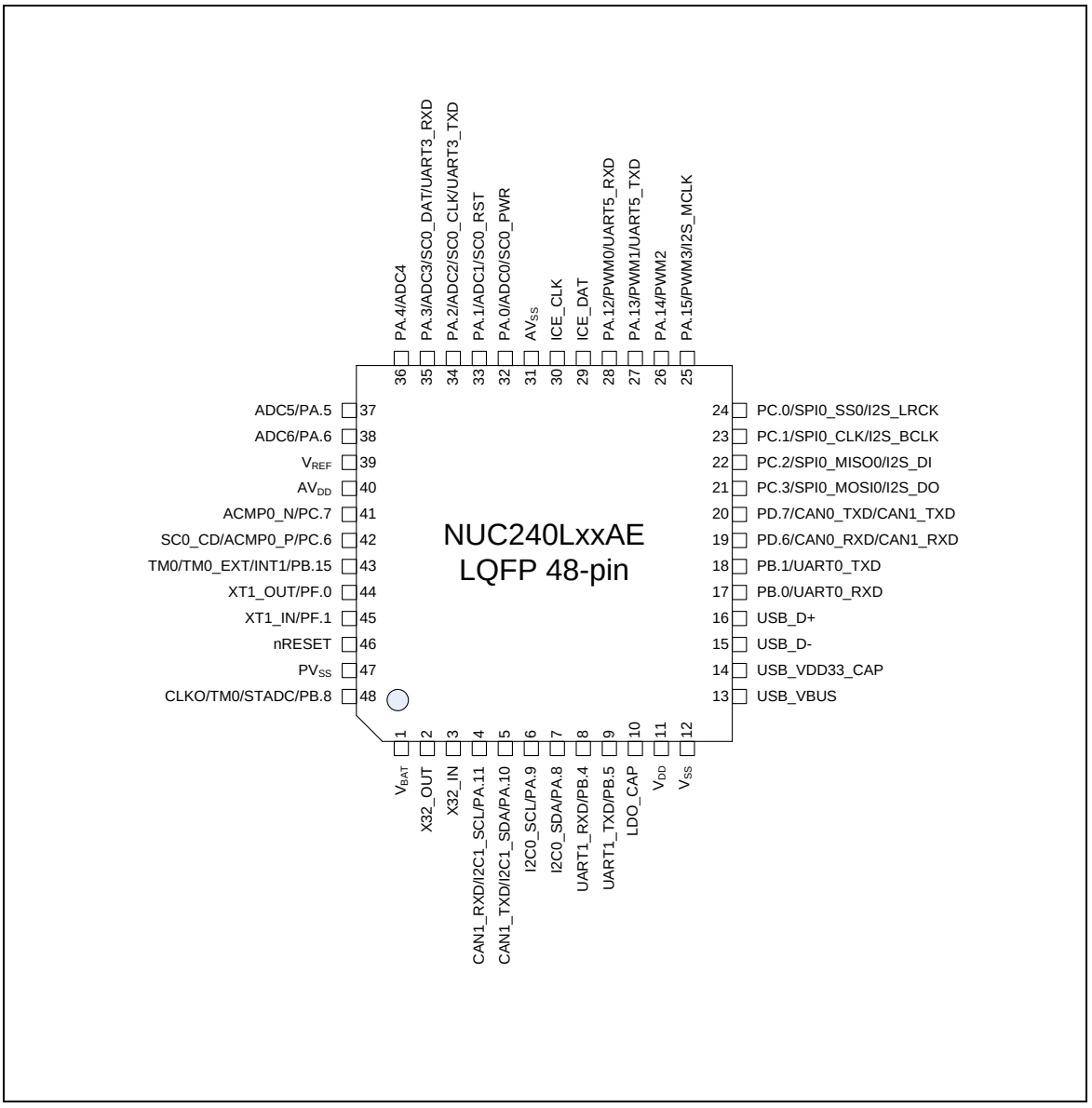


图 3-7 NuMicro™ NUC240LxxAE LQFP 48-pin 管脚图

3.3 管脚描述

3.3.1 NuMicro™ NUC230 管脚描述

管脚号			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
1			PE.15	**I/O**	通用数字输入/输出管脚
2			PE.14	**I/O**	通用数字输入/输出管脚
3			PE.13	**I/O**	通用数字输入/输出管脚
4	1		PB.14	**I/O**	通用数字输入/输出管脚
			AD0	I/O	EBI 地址/数据总线位0
			INT0	**I**	外部中断0输入管脚
			SPI3_SS1	**I/O**	SPI3 2nd 从选择管脚.
5	2		PB.13	**I/O**	通用数字输入/输出管脚
			AD1	I/O	EBI 地址/数据总线位1
			ACMP1_O	**O**	Comparator1输出管脚
6	3	1	V_{BAT}	**P**	RTC电池供电电源管脚.
7	4	2	X32_OUT	**O**	外部32.768 kHz (低速) 晶体输出管脚
8	5	3	X32_IN	**I**	外部32.768 kHz (低速) 晶体输入管脚
9	6	4	PA.11	**I/O**	通用数字输入/输出管脚.
			I2C1_SCL	**I/O**	I^2C1 时钟管脚.
			CAN1_RXD	**I**	CAN1数据接收器输入管脚
			nRD	O	EBI 读使能输出管脚
10	7	5	PA.10	**I/O**	通用数字输入/输出管脚.
			I2C1_SDA	**I/O**	I^2C1 数据输入输出管脚.
			CAN1_TXD	**O**	CAN1数据输出管脚
			nWR	**O**	EBI 写使能输出管脚
11	8	6	PA.9	**I/O**	通用数字输入/输出管脚.
			I2C0_SCL	**I/O**	I^2C0 时钟管脚.
12	9	7	PA.8	**I/O**	通用数字输入/输出管脚.
			I2C0_SDA	**I/O**	I^2C0 数据输入/输出管脚.
13			PD.8	**I/O**	通用数字输入/输出管脚.
			SPI3_SS0	**I/O**	SPI3 1st 从选择管脚.
14			PD.9	**I/O**	通用数字输入/输出管脚.

管脚号			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SPI3_CLK	I/O	SPI3 串行时钟管脚.
15			PD.10	I/O	通用数字输入/输出管脚.
			SPI3_MISO0	I/O	SPI3 1st MISO (主入，从出) 管脚.
16			PD.11	I/O	通用数字输入/输出管脚.
			SPI3_MOSI0	I/O	SPI3 1st MOSI (主出，从入) 管脚.
17			PD.12	I/O	通用数字输入/输出管脚.
			SPI3_MISO1	I/O	SPI3 2nd MISO (主入，从出) 管脚.
18			PD.13	I/O	通用数字输入/输出管脚.
			SPI3_MOSI1	I/O	SPI3 2nd MOSI (主出，从入) 管脚.
19	10	8	PB.4	I/O	通用数字输入/输出管脚.
			UART1_RXD	I	UART1数据接收器输入管脚
20	11	9	PB.5	I/O	通用数字输入/输出管脚.
			UART1_TXD	O	UART1数据输出管脚.
21	12		PB.6	I/O	通用数字输入/输出管脚
			ALE	O	EBI 地址锁存使能输出管脚
			UART1_nRTS	O	UART1请求发送输出管脚
22	13		PB.7	I/O	通用数字输入/输出管脚
			nCS	O	EBI 片选择使能输出管脚
			UART1_nCTS	I	UART1清零发送输入管脚.
23	14	10	LDO_CAP	P	LDO 输出管脚.
24	15	11	V_{DD}	P	电源供应管脚，为IO端口、内部PLL电路LDO源和数字电路提供电源
25	16	12	V_{SS}	P	数字电路地
26			PE.12	I/O	通用数字输入/输出管脚.
27			PE.11	I/O	通用数字输入/输出管脚.
28			PE.10	I/O	通用数字输入/输出管脚.
29			PE.9	I/O	通用数字输入/输出管脚.
30			PE.8	I/O	通用数字输入/输出管脚.
31			PE.7	I/O	通用数字输入/输出管脚.
32	17	13	PB.0	I/O	通用数字输入/输出管脚.
			UART0_RXD	I	UART0数据接收器输入管脚
33	18	14	PB.1	I/O	通用数字输入/输出管脚.

管脚号			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			UART0_TXD	O	UART0数据发送输出管脚
34	19	15	PB.2	I/O	通用数字输入/输出管脚.
			UART0_nRTS	O	UART0请求发送输出管脚
			TM2_EXT	I	Timer2 外部捕捉输入管脚.
			TM2	O	Timer2 toggle 输出管脚.
			ACMP0_O	O	Comparator0 输出管脚.
			nWRL	O	EBI 低字节写使能输出管脚
35	20	16	PB.3	I/O	通用数字输入/输出管脚.
			UART0_nCTS	I	UART0清零发送输入管脚
			TM3_EXT	I	Timer3 外部捕捉输入管脚.
			TM3	O	Timer3 toggle 输出管脚.
			SC2_CD	I	SmartCard2 卡检测输入管脚.
			nWRH	O	EBI高字节写使能输出管脚
36	21	17	PD.6	I/O	通用数字输入/输出管脚
			CAN0_RXD	I	CAN0数据接收器输入管脚.
37	22	18	PD.7	I/O	通用数字输入/输出管脚
			CAN0_TXD	O	CAN0数据发送输出管脚
38	23	19	PD.14	I/O	通用数字输入/输出管脚
			UART2_RXD	I	UART2数据接收器输入管脚.
			CAN1_RXD	I	CAN1数据接收输入管脚
39	24	20	PD.15	I/O	通用数字输入/输出管脚
			UART2_TXD	O	UART2数据发送器输出管脚
			CAN1_TXD	O	CAN1数据发送器输出管脚
40			PC.5	I/O	通用数字输入/输出管脚
			SPI0_MOSI1	I/O	SPI0 2nd MOSI (主出，从入) 管脚.
41			PC.4	I/O	通用数字输入/输出管脚.
			SPI0_MISO1	I/O	SPI0 2nd MISO (主入，从出) 管脚.
42	25	21	PC.3	I/O	通用数字输入/输出管脚.
			SPI0_MOSI0	I/O	SPI0 1st MOSI (主出,从入) 管脚.
			I2S_DO	O	I2S数据输出管脚.

管脚号			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
43	26	22	PC.2	I/O	通用数字输入/输出管脚.
			SPI0_MISO0	I/O	SPI0 1st MISO (主入,从出) 管脚.
			I2S_DI	I	I2S 数据输入管脚
44	27	23	PC.1	I/O	通用数字输入/输出管脚.
			SPI0_CLK	I/O	SPI0 串行时钟输入管脚.
			I2S_BCLK	I/O	I2S 位时钟管脚.
45	28	24	PC.0	I/O	通用数字输入/输出管脚.
			SPI0_SS0	I/O	SPI0 1st 从选择管脚.
			I2S_LRCK	I/O	I2S 左通道时钟管脚.
46			PE.6	I/O	通用数字输入/输出管脚.
47	29		PE.5	I/O	通用数字输入/输出管脚.
			PWM5	I/O	PWM5 输出/捕捉输入管脚
			TM1_EXT	I	Timer1 外部捕捉输入管脚.
			TM1	O	Timer1 toggle 输出管脚.
48	30		PB.11	I/O	通用数字输入/输出管脚.
			TM3	I/O	Timer3 事件计数器输入/ toggle 输出管脚.
			PWM4	I/O	PWM4 输出/捕捉输入管脚.
49	31		PB.10	I/O	通用数字输入/输出管脚.
			TM2	I/O	Timer2事件计数器输入/ toggle 输出管脚
			SPI0_SS1	I/O	SPI0 2nd 从选择管脚.
50	32		PB.9	I/O	通用数字输入/输出管脚.
			TM1	I/O	Timer1事件计数器输入/ toggle 输出管脚
			SPI1_SS1	I/O	SPI1 2nd从选择管脚.
51			PE.4	I/O	通用数字输入/输出管脚.
52			PE.3	I/O	通用数字输入/输出管脚.
53			PE.2	I/O	通用数字输入/输出管脚.
54			PE.1	I/O	通用数字输入/输出管脚.
			PWM7	I/O	PWM7输出/捕捉输入管脚
55			PE.0	I/O	通用数字输入/输出管脚.
			PWM6	I/O	PWM6输出/捕捉输入管脚

管脚号			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
56			PC.13	I/O	通用数字输入/输出管脚.
			SPI1_MOSI1	I/O	SPI1 2nd MOSI (主出,从入) 管脚
57			PC.12	I/O	通用数字输入/输出管脚
			SPI1_MISO1	I/O	SPI1 2nd MISO (主入, 从出) 管脚.
58	33		PC.11	I/O	通用数字输入/输出管脚.
			SPI1_MOSI0	I/O	SPI1 1st MOSI (主出,从入) 管脚.
59	34		PC.10	I/O	通用数字输入/输出管脚.
			SPI1_MISO0	I/O	SPI1 1st MISO (主入, 从出) 管脚.
60	35		PC.9	I/O	通用数字输入/输出管脚.
			SPI1_CLK	I/O	SPI1 串行时钟管脚.
61	36		PC.8	I/O	通用数字输入/输出管脚.
			MCLK	O	EBI 时钟输出
			SPI1_SS0	I/O	SPI1 1st 从选择管脚
62	37	25	PA.15	I/O	通用数字输入/输出管脚.
			PWM3	I/O	PWM 输出/捕捉输入管脚
			I2S_MCLK	O	I2S 主时钟输出管脚.
			SC2_PWR	O	SmartCard2 电源管脚.
63	38	26	PA.14	I/O	通用数字输入/输出管脚.
			PWM2	I/O	PWM2输出/捕捉输入管脚
			SC2_RST	O	SmartCard2 复位管脚
			AD15	I/O	EBI地址/数据总线位15
64	39	27	PA.13	I/O	通用数字输入/输出管脚.
			PWM1	I/O	PWM1输出/捕捉输入管脚
			SC2_CLK	O	SmartCard2 时钟管脚.
			UART5_TXD	O	UART5数据发送器输出管脚.
			AD14	I/O	EBI地址/数据总线位14
65	40	28	PA.12	I/O	通用数字输入/输出管脚.
			PWM0	I/O	PWM0输出/捕捉输入管脚
			SC2_DAT	O	SmartCard2 数据管脚.
			UART5_RXD	I	UART5数据接收器输入管脚

管脚号			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			AD13	I/O	EBI地址/数据总线位13
66	41	29	ICE_DAT	I/O	SWD调试接口数据管脚.
67	42	30	ICE_CLK	I	SWD调试接口时钟管脚.
68			V_{DD}	P	电源供应管脚，为IO端口、内部PLL电路LDO源和数字电路提供电源
69			V_{SS}	P	数字电路地管脚
70	43	31	AV_{SS}	AP	模拟电路地管脚
71	44	32	PA.0	I/O	通用数字输入/输出管脚.
			ADC0	AI	ADC0 模拟输入管脚
			SC0_PWR	O	SmartCard0 电源管脚
72	45	33	PA.1	I/O	通用数字输入/输出管脚.
			ADC1	AI	ADC1模拟输入管脚
			SC0_RST	O	SmartCard0 复位管脚
			AD12	I/O	EBI地址/数据总线位12
73	46	34	PA.2	I/O	通用数字输入/输出管脚.
			ADC2	AI	ADC2模拟输入管脚
			SC0_CLK	O	SmartCard0时钟管脚
			UART3_TXD	O	UART3数据发送器输出管脚
			AD11	I/O	EBI地址/数据总线位11
74	47	35	PA.3	I/O	通用数字输入/输出管脚
			ADC3	AI	ADC3模拟输入管脚
			SC0_DAT	O	SmartCard0 数据管脚
			UART3_RXD	I	UART3数据接收器输入管脚
			AD10	I/O	EBI地址/数据总线位10
75	48	36	PA.4	I/O	通用数字输入/输出管脚
			ADC4	AI	ADC4模拟输入管脚
			SC1_PWR	O	SmartCard1 电源管脚
			AD9	I/O	EBI地址/数据总线位9
76	49	37	PA.5	I/O	通用数字输入/输出管脚
			ADC5	AI	ADC5模拟输入管脚
			SC1_RST	O	SmartCard1 复位管脚
			AD8	I/O	EBI地址/数据总线位8

管脚号			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
77	50	38	PA.6	I/O	通用数字输入/输出管脚
			ADC6	AI	ADC6模拟输入管脚
			SC1_CLK	I/O	SmartCard1 时钟管脚.
			UART4_TXD	O	UART4数据发送器输出管脚
			AD7	I/O	EBI地址/数据总线位7
78			PA.7	I/O	通用数字输入/输出管脚.
			ADC7	AI	ADC7模拟输入管脚
			SC1_DAT	O	SmartCard1 数据管脚
			UART4_RXD	I	UART4数据接收器输入管脚
			SPI2_SS1	I/O	SPI2 2nd 从选择管脚
			AD6	I/O	EBI地址/数据总线位6
79	51	39	V_{REF}	AP	ADC 参考电压输入管脚.
80	52	40	AV_{DD}	AP	内部模拟电路电源管脚
81			PD.0	I/O	通用数字输入/输出管脚
			SPI2_SS0	I/O	SPI2 1st从选择管脚
82			PD.1	I/O	通用数字输入/输出管脚
			SPI2_CLK	I/O	SPI2 串行时钟管脚
83			PD.2	I/O	通用数字输入/输出管脚.
			SPI2_MISO0	I/O	SPI2 1st MISO (主入，从出) 管脚.
84			PD.3	I/O	通用数字输入/输出管脚.
			SPI2_MOSI0	I/O	SPI2 1st MOSI (主出，从入) 管脚.
85			PD.4	I/O	通用数字输入/输出管脚
			SPI2_MISO1	I/O	SPI2 2nd MISO (主入，从出) 管脚.
86			PD.5	I/O	通用数字输入/输出管脚
			SPI2_MOSI1	I/O	SPI2 2nd MOSI (主出，从入) 管脚.
87	53	41	PC.7	I/O	通用数字输入/输出管脚
			ACMP0_N	AI	Comparator0 负输入端管脚.
			SC1_CD	I	SmartCard1卡检测管脚
			AD5	I/O	EBI地址/数据总线位5
88	54	42	PC.6	I/O	通用数字输入/输出管脚.
			ACMP0_P	AI	Comparator0正输入端管脚.

管脚号			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SC0_CD	I	SmartCard0 卡检测管脚
			AD4	I/O	EBI地址/数据总线位4
89	55		PC.15	I/O	通用数字输入/输出管脚.
			AD3	I/O	EBI地址/数据总线位3
			ACMP1_N	AI	Comparator1负输入端管脚.
90	56		PC.14	I/O	通用数字输入/输出管脚
			AD2	I/O	EBI地址/数据总线位2
			ACMP1_P	AI	Comparator1正输入端管脚
91	57	43	PB.15	I/O	通用数字输入/输出管脚.
			INT1	I	外部中断1输入管脚
			TM0_EXT	I	Timer0 外部捕捉输入管脚
			TM0	O	Timer0 toggle 输出管脚
			AD6	I/O	EBI地址/数据总线位6
92	58	44	PF.0	I/O	通用数字输入/输出管脚.
			XT1_OUT	O	外部 4~24 MHz (高速) 晶体输出管脚.
93	59	45	PF.1	I/O	通用数字输入/输出管脚.
			XT1_IN	I	外部 4~24 MHz (高速) 晶体输入管脚.
94	60	46	nRESET	I	外部复位输入: 低电平有效, 带一个内部上拉. 设置该脚为低电平可复位芯片到初始状态
95	61		V_{SS}	P	数字电路地
96	62		V_{DD}	P	电源供应管脚，为IO端口、内部PLL电路LDO源和数字电路提供电源
97			PF.2	I/O	通用数字输入/输出管脚.
			PS/2_DAT	I/O	PS/2 数据管脚.
98			PF.3	I/O	通用数字输入/输出管脚.
			PS/2_CLK	I/O	PS/2 时钟管脚
99	63	47	PV_{SS}	P	PLL 地管脚.
100	64	48	PB.8	I/O	通用数字输入/输出管脚.
			STADC	I	ADC外部触发输入管脚
			TM0	I/O	Timer0 事件计数器输入 / toggle 输出管脚.
			CLKO	O	时钟频率分频输出管脚

注意: 管脚类型 I = 数字输入, O = 数字输出; AI = 模拟输入; P = 电源; AP = 模拟电源

3.3.2 NuMicro™ NUC240 管脚描述

管脚号.			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
1			PE.15	I/O	通用数字输入/输出管脚.
2			PE.14	I/O	通用数字输入/输出管脚.
3			PE.13	I/O	通用数字输入/输出管脚.
4	1		PB.14	I/O	通用数字输入/输出管脚.
			AD0	I/O	EBI地址/数据总线位0
			INT0	I	外部中断0输入管脚.
			SPI3_SS1	I/O	SPI3 2nd 从选择管脚
5	2		PB.13	I/O	通用数字输入/输出管脚.
			AD1	I/O	EBI地址/数据总线位1
			ACMP1_O	O	Comparator1输出管脚.
6	3	1	V_{BAT}	P	RTC 电池供电管脚.
7	4	2	X32_OUT	O	外部32.768 kHz (低速) 晶体输出管脚.
8	5	3	X32_IN	I	外部32.768 kHz (低速) 晶体输入管脚.
9	6	4	PA.11	I/O	通用数字输入/输出管脚.
			I2C1_SCL	I/O	I^2C1 时钟管脚
			CAN1_RXD	I	CAN1数据接收器输入管脚
			nRD	I/O	EBI读使能输出管脚
10	7	5	PA.10	I/O	通用数字输入/输出管脚.
			I2C1_SDA	I/O	I^2C1数据输入/输出管脚.
			CAN1_TXD	O	CAN1数据发送器输出管脚
			nWR	O	EBI 写使能输出管脚
11	8	6	PA.9	I/O	通用数字输入/输出管脚.
			I2C0_SCL	I/O	I^2C0 时钟管脚
12	9	7	PA.8	I/O	通用数字输入/输出管脚.
			I2C0_SDA	I/O	I^2C0数据输入/输出管脚.
13			PD.8	I/O	通用数字输入/输出管脚.
			SPI3_SS0	I/O	SPI3 1st 从选择管脚
14			PD.9	I/O	通用数字输入/输出管脚
			SPI3_CLK	I/O	SPI3 串行时钟管脚
15			PD.10	I/O	通用数字输入/输出管脚.

管脚号.			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SPI3_MISO0	I/O	1st SPI3 MISO(主入，从出) 管脚..
16			PD.11	I/O	通用数字输入/输出管脚.
			SPI3_MOSI0	I/O	SPI3 1st MOSI (主出，从入) 管脚.
17			PD.12	I/O	通用数字输入/输出管脚.
			SPI3_MISO1	I/O	SPI3 2nd MISO (主入，从出) 管脚.
18			PD.13	I/O	通用数字输入/输出管脚.
			SPI3_MOSI1	I/O	SPI3 2nd MOSI (主出,从入) 管脚
19	10	8	PB.4	I/O	通用数字输入/输出管脚.
			UART1_RXD	I	UART1数据接收器输入管脚
20	11	9	PB.5	I/O	通用数字输入/输出管脚.
			UART1_TXD	O	UART1数据发送器输出管脚
21	12		PB.6	I/O	通用数字输入/输出管脚.
			ALE	O	EBI 地址锁存使能输出管脚
			UART1_nRTS	O	UART1 请求发送输出管脚
22	13		PB.7	I/O	通用数字输入/输出管脚.
			nCS	O	EBI 片选使能输出管脚
			UART1_nCTS	I	UART1清零发送输入管脚
23	14	10	LDO_CAP	P	LDO 输出管脚
24	15	11	V_{DD}	P	电源供应管脚，为IO端口、内部PLL电路LDO源和数字电路提供电源
25	16	12	V_{SS}	P	数字电路地管脚.
26			PE.8	I/O	通用数字输入/输出管脚
27			PE.7	I/O	通用数字输入/输出管脚.
28	17	13	USB_VBUS	USB	来自USB 主机或HUB的电源
29	18	14	USB_V_{DD}33_CAP	USB	内部电源调节器3.3V输出去耦管脚
30	19	15	USB_D-	USB	USB差分信号D-.
31	20	16	USB_D+	USB	USB差分信号D+.
32	21	17	PB.0	I/O	通用数字输入/输出管脚.
			UART0_RXD	I	UART0数据接收器输入管脚.
33	22	18	PB.1	I/O	通用数字输入/输出管脚.
			UART0_TXD	O	UART0数据发送器输出管脚

管脚号.			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
34	23		PB.2	I/O	通用数字输入/输出管脚.
			nWRL	O	EBI低字节写使能输出管脚
			UART0_nRTS	O	UART0请求发送输出管脚
			TM2_EXT	I	Timer2 外部捕捉输入管脚
			TM2	O	Timer2 toggle 输出管脚
			ACMP0_O	O	Comparator0 输出管脚
35	24		PB.3	I/O	通用数字输入/输出管脚
			nWRH	O	EBI高字节写使能输出管脚
			UART0_nCTS	I	UART0清零发送输入管脚
			TM3_EXT	I	Timer3 外部捕捉输入管脚
			TM3	O	Timer3 toggle 输出管脚
			SC2_CD	I	SmartCard2 卡检测管脚
36	25	19	PD.6	I/O	通用数字输入/输出管脚.
			CAN0_RXD	I	CAN0数据接收器输入管脚
37	26	20	PD.7	I/O	通用数字输入/输出管脚
			CAN0_TXD	O	CAN0数据发送器输出管脚
38	27		PD.14	I/O	通用数字输入/输出管脚
			UART2_RXD	I	UART2数据接收器输入管脚
			CAN1_RXD	I	CAN1数据接收器输入管脚
39	28		PD.15	I/O	通用数字输入/输出管脚.
			UART2_TXD	O	UART2数据发送器输出管脚
			CAN1_TXD	O	CAN1数据发送器输出管脚
40			PC.5	I/O	通用数字输入/输出管脚
			SPI0_MOSI1	I/O	SPI0 2nd MOSI (主出,从入) 管脚
41			PC.4	I/O	通用数字输入/输出管脚
			SPI0_MISO1	I/O	SPI0 2nd MISO (主入, 从出) 管脚.
42	29	21	PC.3	I/O	通用数字输入/输出管脚
			SPI0_MOSI0	I/O	SPI0 1st MOSI (主出,从入) 管脚.
			I2S_DO	O	I2S 数据输出管脚
43	30	22	PC.2	I/O	通用数字输入/输出管脚.

管脚号.			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SPI0_MISO0	I/O	SPI0 1st MISO(主入，从出) 管脚.
			I2S_DI	I	I2S 数据输入管脚
44	31	23	PC.1	I/O	通用数字输入/输出管脚
			SPI0_CLK	I/O	SPI0 串行时钟管脚
			I2S_BCLK	I/O	I2S位时钟管脚
45	32	24	PC.0	I/O	通用数字输入/输出管脚.
			SPI0_SS0	I/O	SPI0 1st 从选择管脚
			I2S_LRCK	I/O	I2S 左右通道时钟
46			PE.6	I/O	通用数字输入/输出管脚.
47			PE.5	I/O	通用数字输入/输出管脚.
			PWM5	I/O	PWM5输出/捕捉输入管脚
			TM1_EXT	I	Timer1 外部捕捉输入管脚
			TM1	O	Timer1 toggle 输出管脚
48			PB.11	I/O	通用数字输入/输出管脚
			TM3	I/O	Timer3 事件计数器输入 / toggle 输出管脚.
			PWM4	I/O	PWM4输出/捕捉输入管脚
49			PB.10	I/O	通用数字输入/输出管脚
			TM2	I/O	Timer2事件计数器输入 / toggle 输出管脚.
			SPI0_SS1	I/O	SPI0 2nd从选择管脚
50			PB.9	I/O	通用数字输入/输出管脚
			TM1	I/O	Timer1事件计数器输入 / toggle 输出管脚.
			SPI1_SS1	I/O	SPI1 2nd 从选择管脚
51			PE.4	I/O	通用数字输入/输出管脚
52			PE.3	I/O	通用数字输入/输出管脚
53			PE.2	I/O	通用数字输入/输出管脚
54			PE.1	I/O	通用数字输入/输出管脚
			PWM7	I/O	PWM7输出/捕捉输入管脚
55			PE.0	I/O	通用数字输入/输出管脚
			PWM6	I/O	PWM6输出/捕捉输入管脚
56			PC.13	I/O	通用数字输入/输出管脚

管脚号.			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SPI1_MOSI1	I/O	SPI1 2nd MOSI (主出,从入) 管脚
57			PC.12	I/O	通用数字输入/输出管脚.
			SPI1_MISO1	I/O	SPI1 2nd MISO (主入, 从出) 管脚.
58	33		PC.11	I/O	通用数字输入/输出管脚.
			SPI1_MOSI0	I/O	SPI1 1st MOSI (主出,从入) 管脚
59	34		PC.10	I/O	通用数字输入/输出管脚.
			SPI1_MISO0	I/O	SPI1 1st MISO (主入, 从出) 管脚..
60	35		PC.9	I/O	通用数字输入/输出管脚.
			SPI1_CLK	I/O	SPI1 串行时钟管脚
61	36		PC.8	I/O	通用数字输入/输出管脚.
			MCLK	O	EBI 时钟输出
			SPI1_SS0	I/O	SPI1 1st从选择管脚
62	37	25	PA.15	I/O	通用数字输入/输出管脚.
			PWM3	I/O	PWM3输出/捕捉输入管脚
			I2S_MCLK	O	I2S 主时钟输出管脚
			SC2_PWR	O	SmartCard2 电源管脚
63	38	26	PA.14	I/O	通用数字输入/输出管脚.
			PWM2	I/O	PWM2输出/捕捉输入管脚
			AD15	I/O	EBI地址/数据总线位15
			SC2_RST	O	SmartCard2 复位管脚
64	39	27	PA.13	I/O	通用数字输入/输出管脚.
			PWM1	I/O	PWM1输出/捕捉输入管脚
			SC2_CLK	O	SmartCard2 时钟管脚
			AD14	I/O	EBI地址/数据总线位14
		27	UART5_TXD	O	UART5数据发送器输出管脚
65	40	28	PA.12	I/O	通用数字输入/输出管脚.
			PWM0	I/O	PWM0输出/捕捉输入管脚
			SC2_DAT	O	SmartCard2 数据管脚
			AD13	I/O	EBI地址/数据总线位13
		28	UART5_RXD	I	UART5数据接收器输入管脚
66	41	29	ICE_DAT	I/O	SDW调试接口数据管脚

管脚号.			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
67	42	30	ICE_CLK	I	SDW调试接口时钟管脚
68			V_{DD}	P	电源供应管脚，为IO端口、内部PLL电路LDO源和数字电路提供电源
69			V_{SS}	P	数字电路地
70	43	31	AV_{SS}	AP	模拟电路地
71	44	32	PA.0	I/O	通用数字输入/输出管脚
			ADC0	AI	ADC0模拟输入管脚
			SC0_PWR	O	SmartCard0 电源管脚.
72	45	33	PA.1	I/O	通用数字输入/输出管脚
			ADC1	AI	ADC1模拟输入管脚
			AD12	I/O	EBI地址/数据总线位12
			SC0_RST	O	SmartCard0 复位管脚
73	46	34	PA.2	I/O	通用数字输入/输出管脚
			ADC2	AI	ADC2模拟输入管脚
			SC0_CLK	O	SmartCard0 时钟管脚
			AD11	I/O	EBI地址/数据总线位11
			UART3_TXD	O	UART3数据发送器输出管脚
74	47	35	PA.3	I/O	通用数字输入/输出管脚.
			ADC3	AI	ADC3模拟输入管脚
			SC0_DAT	O	SmartCard0 数据管脚
			AD10	I/O	EBI地址/数据总线位10
			UART3_RXD	I	UART3数据接收器输入管脚
75	48	36	PA.4	I/O	通用数字输入/输出管脚.
			ADC4	AI	ADC4模拟输入管脚
			AD9	I/O	EBI地址/数据总线位9
			SC1_PWR	O	SmartCard1 电源管脚
76	49	37	PA.5	I/O	通用数字输入/输出管脚.
			ADC5	AI	ADC5模拟输入管脚
			AD8	I/O	EBI地址/数据总线位8
			SC1_RST	O	SmartCard1 复位管脚
77	50	38	PA.6	I/O	通用数字输入/输出管脚.
			ADC6	AI	ADC6模拟输入管脚

管脚号.			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SC1_CLK	I/O	SmartCard1 时钟管脚.
			AD7	I/O	EBI地址/数据总线位7
			UART4_TXD	O	UART4数据发送器输出管脚
78			PA.7	I/O	通用数字输入/输出管脚.
			ADC7	AI	ADC7模拟输入管脚
			AD6	I/O	EBI地址/数据总线位6
			SC1_DAT	O	SmartCard1 数据管脚
			UART4_RXD	I	UART4数据接收器输入管脚
			SPI2_SS1	I/O	SPI2 2nd 从选择管脚
79	51	39	V_{REF}	AP	ADC 参考电压输入管脚
80	52	40	AV_{DD}	AP	内部模拟电路电源输入管脚
81			PD.0	I/O	通用数字输入/输出管脚
			SPI2_SS0	I/O	SPI2 1st 从选择管脚
82			PD.1	I/O	通用数字输入/输出管脚.
			SPI2_CLK	I/O	SPI2串行时钟管脚
83			PD.2	I/O	通用数字输入/输出管脚.
			SPI2_MISO0	I/O	SPI2 1st MISO(主入，从出) 管脚.
84			PD.3	I/O	通用数字输入/输出管脚.
			SPI2_MOSI0	I/O	SPI2 1st MOSI (主出,从入) 管脚
85			PD.4	I/O	通用数字输入/输出管脚.
			SPI2_MISO1	I/O	SPI2 2nd MISO (主入，从出) 管脚.
86			PD.5	I/O	通用数字输入/输出管脚.
			SPI2_MOSI1	I/O	SPI2 2nd MOSI (主出,从入) 管脚
87	53	41	PC.7	I/O	通用数字输入/输出管脚.
			ACMP0_N	AI	Comparator0 负输入端管脚.
			AD5	I/O	EBI地址/数据总线位5
			SC1_CD	I	SmartCard1 卡检测管脚
88	54	42	PC.6	I/O	通用数字输入/输出管脚.
			ACMP0_P	AI	Comparator0正输入端管脚
			AD4	I/O	EBI地址/数据总线位4
			SC0_CD	I	SmartCard0 卡检测管脚.

管脚号.			管脚名称	管脚类型	管脚描述
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
89	55		PC.15	I/O	通用数字输入/输出管脚.
			AD3	I/O	EBI地址/数据总线位3
			ACMP1_N	AI	Comparator1负输入端管脚.
90	56		PC.14	I/O	通用数字输入/输出管脚.
			AD2	I/O	EBI地址/数据总线位2
			ACMP1_P	AI	Comparator1正输入端管脚.
91	57	43	PB.15	I/O	通用数字输入/输出管脚.
			INT1	I	外部中断1输入脚
			TM0_EXT	I	Timer 0 外部捕捉输入管脚
			TM0	O	Timer0 toggle 输出管脚
			AD6	I/O	EBI地址/数据总线位6
92	58	44	PF.0	I/O	通用数字输入/输出管脚.
			XT1_OUT	O	外部 4~24 MHz (高速) 晶体输出管脚.
93	59	45	PF.1	I/O	通用数字输入/输出管脚.
			XT1_IN	I	外部 4~24 MHz (高速) 晶体输入管脚.
94	60	46	nRESET	I	外部复位输入: 低电平有效, 带一个内部上拉. 设置该脚为低电平可复位芯片到初始状态
95	61		V_{SS}	P	数字电路地管脚
96	62		V_{DD}	P	电源供应管脚，为IO端口、内部PLL电路LDO源和数字电路提供电源
97			PF.2	I/O	通用数字输入/输出管脚.
			PS/2_DAT	I/O	PS/2数据管脚
98			PF.3	I/O	通用数字输入/输出管脚.
			PS/2_CLK	I/O	PS/2 时钟管脚
99	63	47	PV_{SS}	P	PLL 地管脚
100	64	48	PB.8	I/O	通用数字输入/输出管脚
			STADC	I	ADC 外部触发输入管脚
			TM0	I/O	Timer0 事件计数器输入 / toggle 输出管脚.
			CLKO	O	时钟频率分频输出管脚

注意: 管脚类型 I = 数字输入, O = 数字输出; AI = 模拟输入; P = 电源; AP = 模拟电源

4 方块图

4.1 NuMicro™ NUC230 模块框图

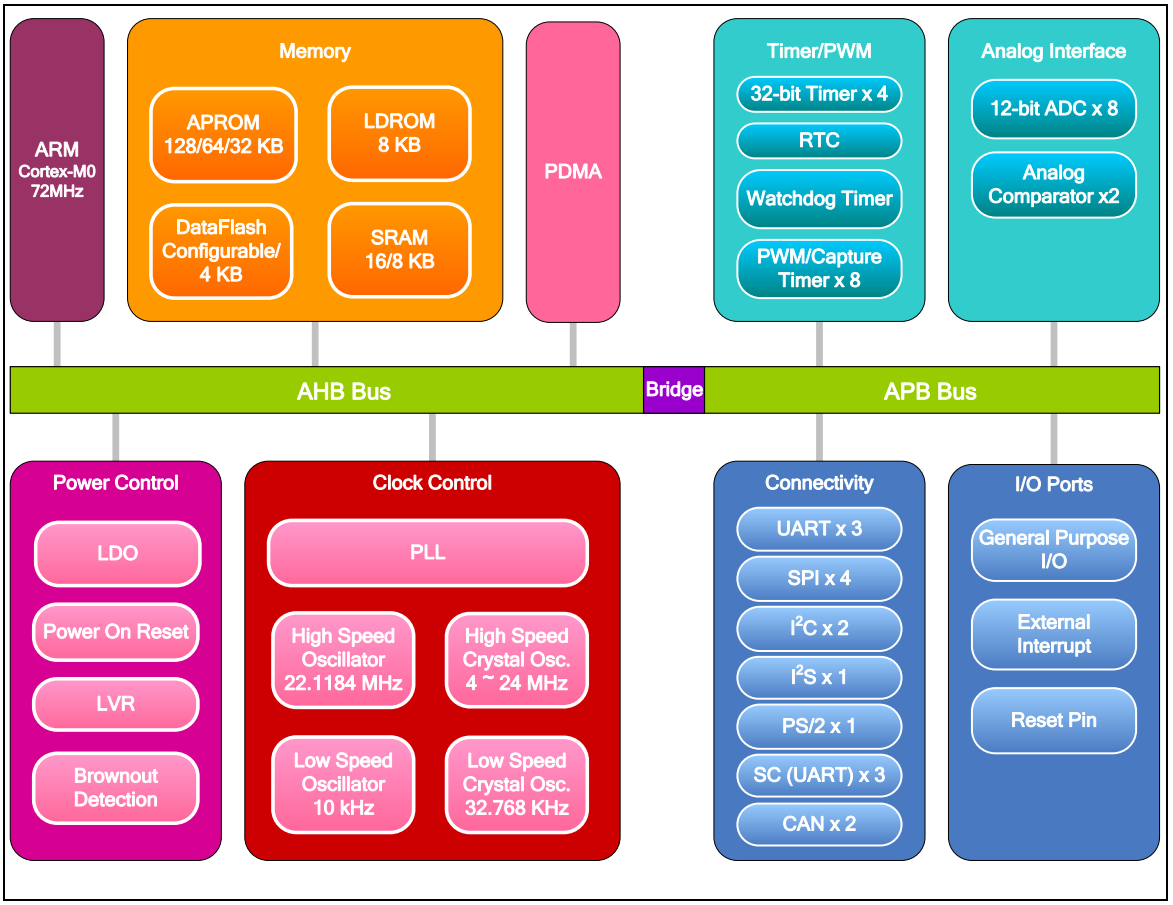


图 4-1 NuMicro™ NUC230 模块框图

4.2 NuMicro™ NUC240 模块框图

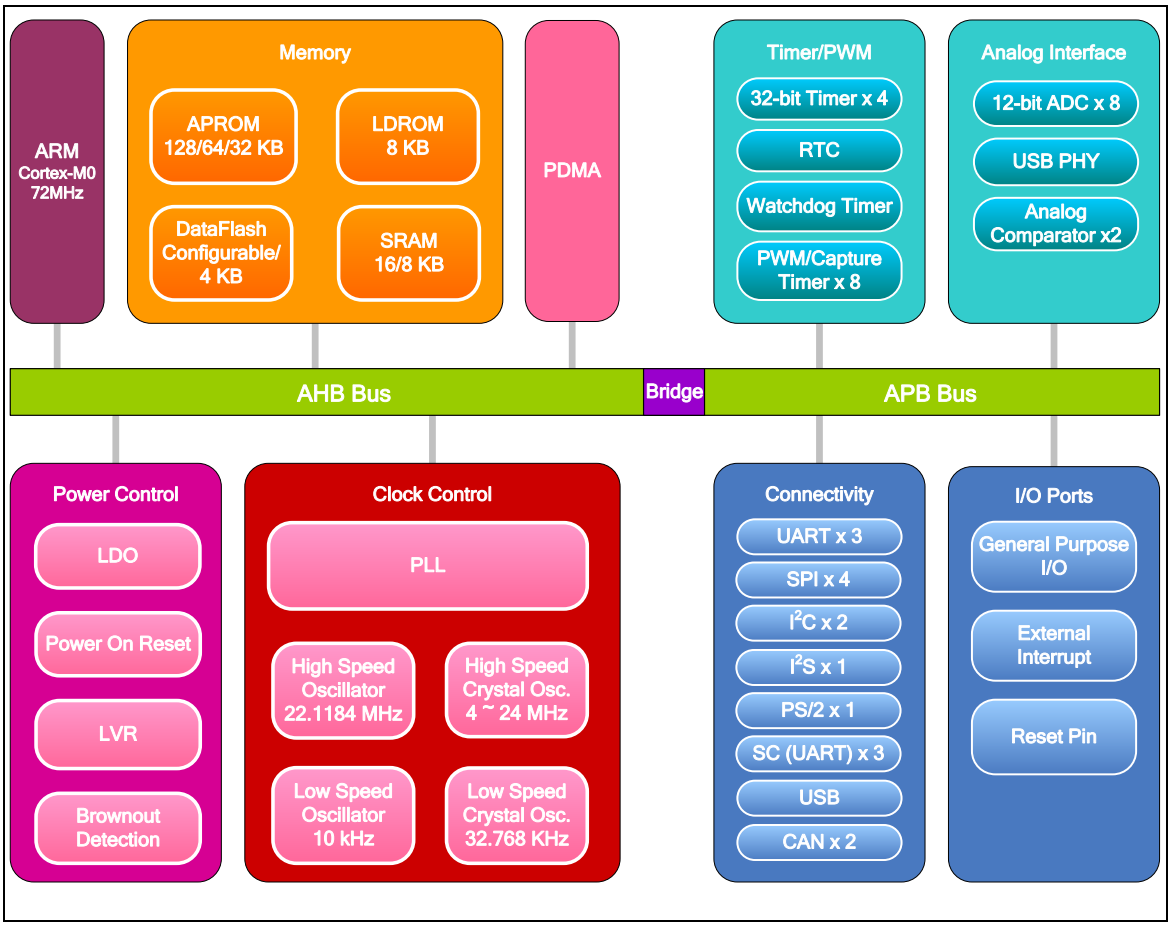


图 4-2 NuMicro™ NUC240 模块框图

5 功能描述

5.1 ARM® Cortex™-M0 内核

Cortex™-M0处理器是一个可配置，多级流水线的32位精简指令集处理器。它有 AMBA、AHB-Lite 接口和嵌套向量中断控制器（NVIC），具有可选的硬件调试功能，可以执行Thumb指令，并与其它Cortex-M系列兼容。支持两种模式-Thread 模式与 Handler 模式。异常时系统进入 Handler 模式。从Handler 模式返回时，执行异常返回。复位时系统进入Thread 模式。Thread 模式也可由异常返回时进入。

图 5-1为处理器的功能图

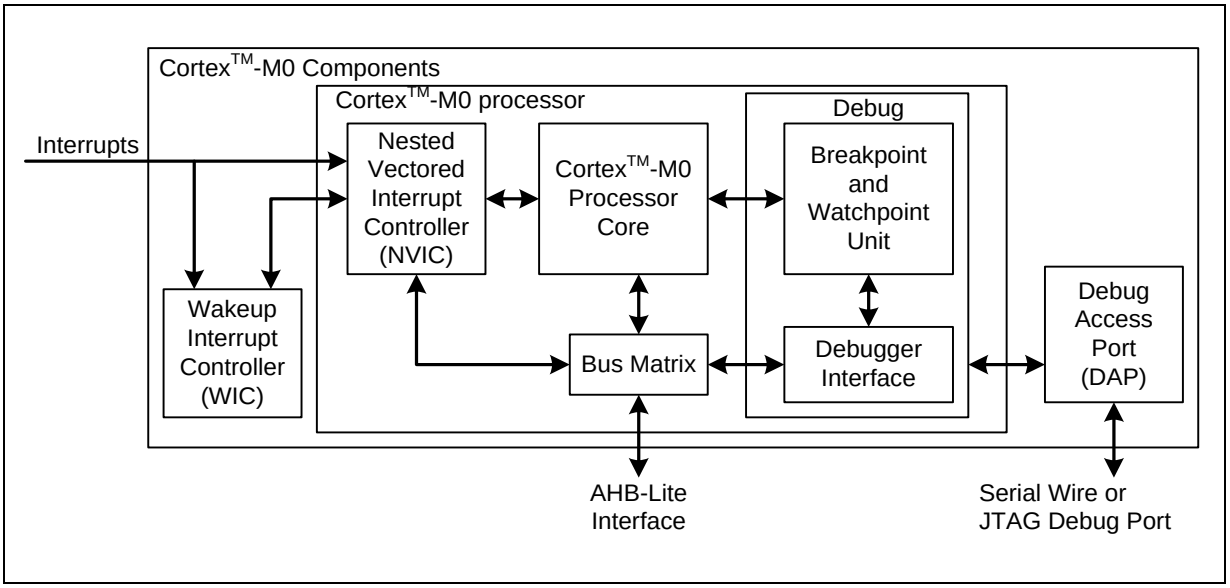


图 5-1 功能控制器框图

设备提供了以下组件及特性

- 低门数处理器:
  - ARMv6-M Thumb® 指令集
  - Thumb-2 技术
  - ARMv6-M 兼容 24-位 系统定时器
  - 一个32-位 硬件乘法器
  - 系统接口支持小端（little-endian）数据访问
  - 准确而及时的中断处理能力
  - 加载/存储多个数据和多周期乘法指令可被终止然后重新开始从而实现快速中断处理
  - C 应用程序二进制接口的异常兼容模式（C-ABI）。这个 ARMv6-M 的模式允许用户使用纯C函数实现中断处理。
  - 使用中断唤醒（WFI）进入低功耗的休眠模式，事件唤醒（WFE）指令或者从中断退出休眠模式

-

- NVIC 特性:
  - 32 个外部中断，每个中断有4个优先级
  - 专用的不可屏蔽中断（NMI）
  - 同时支持电平和脉冲中断触发
  - 中断唤醒控制器（WIC），支持低功耗睡眠模式
- 调试
  - 四个硬件断点
  - 两个观察点
  - 用于非侵入式代码分析的程序计数采样寄存器（PCSR）
  - 单步和向量捕获能力
- 总线接口:
  - 提供简单的集成到所有系统外设和存储器的单一32位 AMBA-3 ABH-Lite 系统接口
  - 支持DAP (Debug Access Port) 的单一32位的从机端口

5.2 系统管理器

5.2.1 概述

系统管理包括如下功能：

- 系统复位
- 系统内存映射
- 产品 ID、芯片复位、模块功能复位和多功能管脚控制的系统管理寄存器
- 系统定时器 (SysTick)
- 嵌套中断向量控制器（NVIC）
- 系统控制寄存器

5.2.2 系统复位

系统复位可以由如下的任何一种中断实现，这些复位中断标志可以通过寄存器RSTSRC读取。

- 上电复位
- nRESET引脚低电平复位
- 看门狗复位
- 低压复位
- 欠压检测器复位
- CPU 复位
- 系统复位

系统复位和上电复位可以复位整个芯片，包含外围设备。系统复位和上电复位的区别在于外部晶振电路和BS(ISPCON[1]) 位。系统复位不复位外部晶振电路和BS(ISPCON[1]) 位，但上电复位可以。

5.2.3 系统电源分配

该器件的电源分配包括三个部分：

- 由AV_{DD} 和 AV_{SS}提供的模拟电源，为芯片模拟部分工作提供电压。
- 由V_{DD} 和 V_{SS} 提供的数字电源，提供一个固定的1.8V数字电源，用于数字部分和I/O 引脚工作
- V_{BUS} 提供给USB的电源，用于USB模块传输操作。
- V_{BAT} 提供给电池的电源， 用于RTC和外部的32.768 kHz晶振。

内部的电压调节器LDO和V_{DD33}. 要求在相应的引脚上外接电容，并尽量靠近引脚摆放。模拟电源(AV_{DD})要与数字电源(V_{DD})是同一个电压准位。图6-2说明了NuMicro™ NUC230的电源分布，图表6-4说明了NuMicro™ NUC240的电源分布。

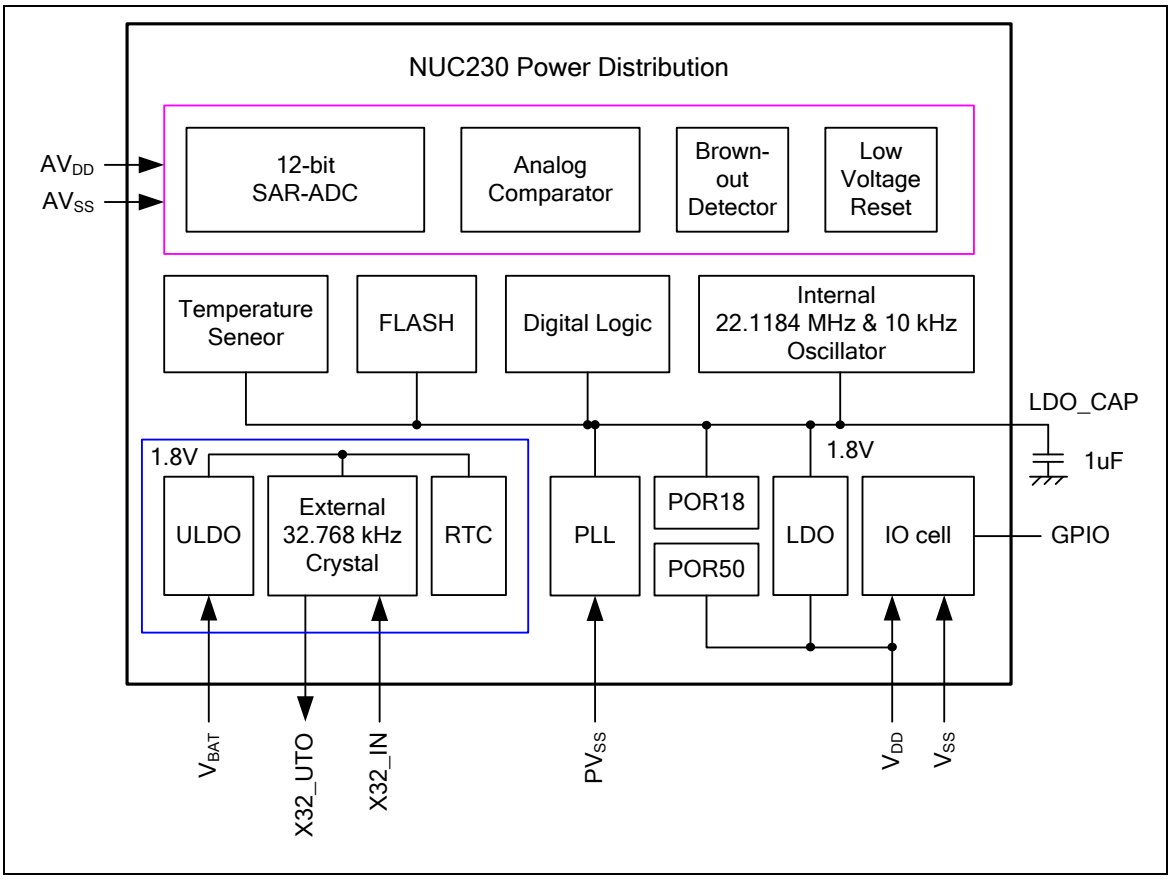


图5-2 NuMicro™ NUC230电源分布

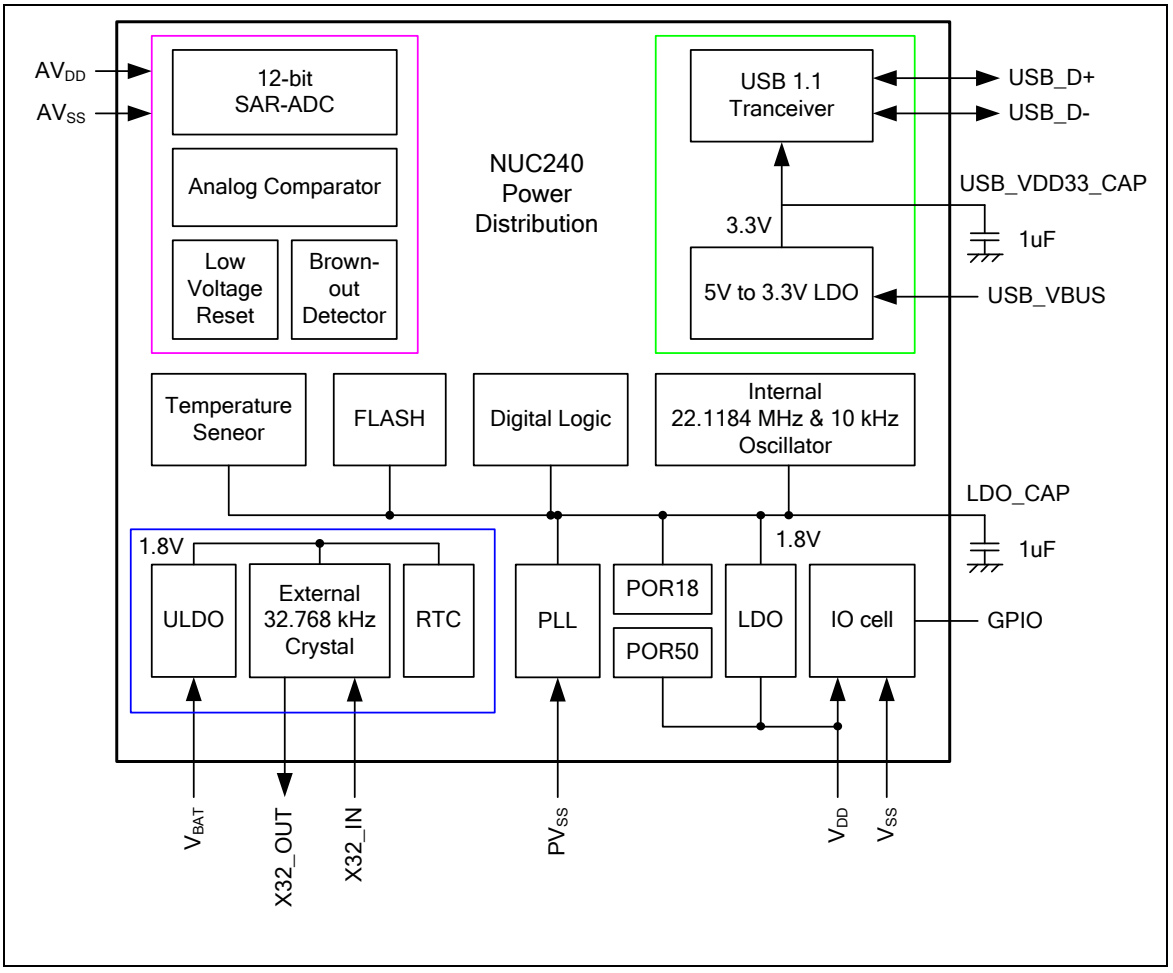


图5-3NuMicro™ NUC240电源分布

5.2.4 系统内存映射

NuMicro™ NUC200 系列提供了4G字节寻址空间。片内控制器的内存地址分配如下表所示。对片上外设的详细寄存器定义，内存空间，和编程指南，将在每个章节中详细描述。 NuMicro™ NUC200 系列只支持小端数据格式。

地址空间	标志	控制器
Flash 和 SRAM 内存空间		
0x0000_0000 – 0x0001_FFFF	FLASH_BA	Flash存储空间(128 KB)
0x2000_0000 – 0x2000_3FFF	SRAM_BA	SRAM存储空间(16K)
AHB Controllers Space (0x5000_0000 – 0x501F_FFFF)		
0x5000_0000 – 0x5000_01FF	GCR_BA	系统全局控制寄存器
0x5000_0200 – 0x5000_02FF	CLK_BA	时钟控制寄存器
0x5000_0300 – 0x5000_03FF	INT_BA	多路中断控制寄存器
0x5000_4000 – 0x5000_7FFF	GPIO_BA	GPIO 控制寄存器
0x5000_8000 – 0x5000_BFFF	PDMA_BA	外围DMA 控制寄存器
0x5000_C000 – 0x5000_FFFF	FMC_BA	Flash 内存控制寄存器
0x5001_0000 – 0x5001_03FF	EBI_BA	EB外部总线接口控制寄存器
APB1 Controllers Space (0x4000_0000 ~ 0x400F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	看门狗控制寄存器
0x4000_8000 – 0x4000_BFFF	RTC_BA	实时时钟（RTC）控制寄存器
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 控制寄存器
0x4002_0000 – 0x4002_3FFF	I2C0_BA	I^2C0接口控制寄存器
0x4003_0000 – 0x4003_3FFF	SPI0_BA	带主/从功能的SPI0控制寄存器
0x4003_4000 – 0x4003_7FFF	SPI1_BA	带主/从功能的SPI1控制寄存器
0x4004_0000 – 0x4004_3FFF	PWMA_BA	PWM0/1/2/3 控制寄存器
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0 控制寄存器
0x4006_0000 – 0x4006_3FFF	USBD_BA	USB 2.0 FS 设备控制寄存器
0x400D_0000 – 0x400D_3FFF	ACMP_BA	模拟比较控制寄存器
0x400E_0000 – 0x400E_FFFF	ADC_BA	模拟数字转换(ADC) 控制寄存器
APB2 Controllers Space (0x4010_0000 ~ 0x401F_FFFF)		
0x4010_0000 – 0x4010_3FFF	PS2_BA	PS/2 接口控制寄存器
0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3 控制寄存器
0x4012_0000 – 0x4012_3FFF	I2C1_BA	I^2C1接口控制寄存器
0x4013_0000 – 0x4013_3FFF	SPI2_BA	带主/从功能的SPI2控制寄存器

0x4013_4000 – 0x4013_7FFF	SPI3_BA	带主/从功能的SPI3控制寄存器
0x4014_0000 – 0x4014_3FFF	PWMB_BA	PWM4/5/6/7 控制寄存器
0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 控制寄存器
0x4015_4000 – 0x4015_7FFF	UART2_BA	UART2 控制寄存器
0x4018_0000 – 0x4018_3FFF	CAN0_BA	CAN0 总线控制寄存器
0x4018_4000 – 0x4018_7FFF	CAN1_BA	CAN1 总线控制寄存器
0x4019_0000 – 0x4019_3FFF	SC0_BA	SC0 控制寄存器
0x4019_4000 – 0x4019_7FFF	SC1_BA	SC1 控制寄存器
0x4019_8000 – 0x4019_BFFF	SC2_BA	SC2 控制寄存器
0x401A_0000 – 0x401A_3FFF	I2S_BA	I^2S 接口控制寄存器
System Controllers Space (0xE000_E000 ~ 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SCS_BA	系统定时器控制寄存器
0xE000_E100 – 0xE000_ECFF	SCS_BA	外围中断控制器控制寄存器
0xE000_ED00 – 0xE000_ED8F	SCS_BA	系统控制寄存器

表 5-1 片上控制器地址空间分配

5.2.5 寄存器映射

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
GCR 基地址: **GCR_BA = 0x5000_0000**				
PDID	GCR_BA+0x00	R	器件ID 寄存器	0x2014_0018[1]
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX
IPRSTC1	GCR_BA+0x08	R/W	外围复位控制寄存器1	0x0000_0000
IPRSTC2	GCR_BA+0x0C	R/W	外围复位控制寄存器2	0x0000_0000
IPRSTC3	GCR_BA+0x10	R/W	外围复位控制寄存器3	0x0000_0000
BODCR	GCR_BA+0x18	R/W	欠压检测控制寄存器	0x0000_008X
TEMPCR	GCR_BA+0x1C	R/W	温度传感器控制寄存器	0x0000_0000
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_XXXX
GPA_MFP	GCR_BA+0x30	R/W	GPIOA 复用功能和输入类型控制寄存器	0x0000_0000
GPB_MFP	GCR_BA+0x34	R/W	GPIOB 复用功能和输入类型控制寄存器	0x0000_0000
GPC_MFP	GCR_BA+0x38	R/W	GPIOC 复用功能和输入类型控制寄存器	0x0000_0000
GPD_MFP	GCR_BA+0x3C	R/W	GPIOD 复用功能和输入类型控制寄存器	0x0000_0000
GPE_MFP	GCR_BA+0x40	R/W	GPIOE 复用功能和输入类型控制寄存器	0x0000_0000
GPF_MFP	GCR_BA+0x44	R/W	GPIOF 复用功能和输入类型控制寄存器	0x0000_000X
ALT_MFP	GCR_BA+0x50	R/W	复用多功能引脚控制寄存器	0x0000_0000
ALT_MFP1	GCR_BA+0x58	R/W	复用多功能引脚控制寄存器1	0x0000_0000
ALT_MFP2	GCR_BA+0x5C	R/W	复用多功能引脚控制寄存器2	0x0000_0000
IRCTRIMCTL	GCR_BA+0x80	R/W	IRC Trim 控制寄存器	0x0000_0000
IRCTRIMIEN	GCR_BA+0x84	R/W	IRC Trim 中断使能寄存器	0x0000_0000
IRCTRIMINT	GCR_BA+0x88	R/W	IRC Trim 中断状态寄存器	0x0000_0000
REGWRPROT	GCR_BA+0x100	R/W	寄存器写保护寄存器	0x0000_0000

注: [1] 依据每个产品型号而定

5.2.6 寄存器描述

器件ID寄存器(PDID)

寄存器	偏移地址	R/W	描述	复位值
PDID	GCR_BA+0x00	R	器件ID寄存器	0x2014_0018[1]

[1] 每个器件具有一个独一无二的默认复位值

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

位	描述	
[31:0]	PDID	**产品器件识别码** 该寄存器反应设备的器件号码. 软件可以读取该寄存器来识别所使用的器件。

系统复位源寄存器(RSTSRC)

该寄存器提供一些信息用于识别引起芯片上次复位操作的复位源。

寄存器	偏移地址	R/W	描述	复位值
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	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
RSTS_CPU	Reserved	RSTS_SYS	RSTS_BOD	RSTS_LVR	RSTS_WDT	RSTS_RESET	RSTS_POR

位	描述	
[31:8]	**Reserved**	保留
[7]	**RSTS_CPU**	**CPU 复位标志** 如果软件写1到CPU_RST(IPRSTC1[1]) 1，复位Cortex™-M0内核和Flash内存控制器(FMC). 硬件会把 RSTS_CPU 标志位置起. 0 = CPU无复位 1 = Cortex™-M0 CPU 内核与FMC因为软件置CPU_RST(IPRSTC1[1]) 为 1而复位。 注：向该位写1清零。
[6]	**Reserved**	保留
[5]	**RSTS_SYS**	**系统复位标志** RSTS_SYS 标志位由来自Cortex™-M0核的“复位信号”置位，用于表示导致之前复位的复位源。 0 = Cortex™-M0无复位 1 = Cortex™-M0 因为软件向SYSRESETREQ (AIRCR[2])写1，发出复位信号而复位系统(AIRCR[2]寄存器的地址是0xE000ED0C). 注：向该位写1清零。
[4]	**RSTS_BOD**	**欠压检测复位标志** RSTS_BOD 标志位由欠压检测模块的“复位信号”置位，用于表示导致之前复位的复位源。 0 = BOD 无复位。 1= 欠压检查模块发出复位信号使系统复位。 注：向该位写1清零。
[3]	**RSTS_LVR**	**低电压复位标志** RSTS_LVR标志位由低压复位模块的“复位信号”置位，用于表示导致之前复位的复位源。 0 =LVR 无复位

		1 =LVR 模块发出复位信号使系统复位 注：向该位写1清零。
[2]	**RSTS_WDT**	**看门狗复位标志** RSTS_WDT标志位由看门狗模块或窗口看门狗的“复位信号”置起，用于表示导致之前复位的复位源。 0 = 看门狗或窗口看门狗无复位 1= 看门狗或窗口看门狗发出复位信号来复位系统 **注1:** 向该位写1清零。 **注2:** 系统发生WDT看门狗复位，WTRF(WTCR[2])置1。系统发生WWDT看门狗复位，WWDTRF(WWDTSR)位置1。
[1]	**RSTS_RESET**	**复位引脚复位标志** RSTS_RESET标志由nRESET引脚的“复位信号”置起，用于表示导致之前复位的复位源。 0= 复位引脚无复位 1= 复位引脚发出复位信号使系统复位 注：向该位写1清零。
[0]	**RSTS_POR**	**上电复位标志** RSTS_POR 标志由上电复位（POR）控制器置起或CHIP_RST (IPRSTC1[0])位置起 用来表示导致之前复位的复位源。 0= 上电复位(POR)或CHIP_RST (IPRSTC1[0])无复位 1= 上电复位(POR)或CHIP_RST (IPRSTC1[0])发出复位信号使系统复位 注：向该位写1清零。

外设复位控制寄存器1(IPRSTC1)

寄存器	偏移地址	R/W	描述	复位值
IPRSTC1	GCR_BA+0x08	R/W	外设复位控制寄存器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				EBI_RST	PDMA_RST	CPU_RST	CHIP_RST

位	描述	
[31:3]	Reserved	保留
[3]	EBI_RST	**EBI 控制器复位（写保护）** 该位置1，产生复位信号到 EBI, 用户需要置0才能释放复位状态。 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h”来关闭寄存器保护。该位参考寄存器REGWRPROT（地址GCR_BA+0x100） 0 = EBI 控制器正常工作 1 = EBI 控制器复位
[2]	PDMA_RST	**PDMA 控制器复位（写保护）** 该位置1，产生复位信号到PDMA, 用户需要置0才能释放复位状态。 0 = PDMA 控制器正常工作 1 = PDMA 控制器复位 注1：该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该位参考寄存器REGWRPROT（地址GCR_BA+0x100） 注2：设置PDMA_RST位为1，将向PDMA模组产生异步的复位信号。用户需要设置PDMA_RST为0来解除PDMA的复位状态。
[1]	CPU_RST	**CPU内核复位（写保护）** 设置该位仅复位CPU内核和Flash存储控制器(FMC),该位将在2个时钟周期后自动清零。 0= CPU 正常工作 1= CPU 复位 注：该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[0]	CHIP_RST	**CHIP 复位(写保护)** 设置该位复位整个芯片，包括 CPU内核和所有外设，该位将在2个时钟周期后自动清零。 CHIP_RST与上电复位(POR)一样，所有芯片控制器都复位，芯片设置从flash重新加载。 CHIP_RST和SYSRESETREQ的区别，请参考章节5.2.2

		0= CHIP正常工作 1= CHIP复位 注：该位受保护，编程该位时，需要一次向地址0x5000_0100写入“59h”, “16h”, “88h”，操作参考寄存器REGWRPROT(地址GCR_BA+0x100)

外设复位控制寄存器2(IPRSTC2)

置1会产生异步复位信号给相应的控制器。用户需要将该位置0才能将相应的控制器从复位状态恢复。

寄存器	偏移地址	R/W	描述	复位值
IPRSTC2	GCR_BA+0x0C	R/W	外设复位控制寄存器2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		I2S_RST	ADC_RST	USBD_RST	Reserved	CAN1_RST	CAN0_RST
23	22	21	20	19	18	17	16
PS2_RST	ACMP_RST	PWM47_RST	PWM03_RST	Reserved	UART2_RST	UART1_RST	UART0_RST
15	14	13	12	11	10	9	8
SPI3_RST	SPI2_RST	SPI1_RST	SPI0_RST	Reserved		I2C1_RST	I2C0_RST
7	6	5	4	3	2	1	0
Reserved		TMR3_RST	TMR2_RST	TMR1_RST	TMR0_RST	GPIO_RST	Reserved

位	描述	
[31:30]	**Reserved**	保留
[29]	**I2S_RST**	**I^2S控制器复位** 0 = I^2S 控制器正常工作 1 = I^2S 控制器复位
[28]	**ADC_RST**	**ADC 控制器复位** 0 = ADC 控制器正常工作 1 = ADC 控制器复位
[27]	**USBD_RST**	**USB 设备控制器复位** 0 = USB 设备控制器正常工作 1 = USB 设备控制器复位
[26]	**Reserved**	保留
[25]	**CAN1_RST**	**CAN1 控制器复位** 0 = CAN1 控制器正常工作 1 = CAN1 控制器复位
[24]	**CAN0_RST**	**CAN0控制器复位** 0 = CAN0 控制器正常工作 1 = CAN0 控制器复位
[23]	**PS2_RST**	**PS/2 控制器复位** 0 = PS/2 控制器正常工作 1 = PS/2 控制器复位
[22]	**ACMP_RST**	**模拟比较控制器复位** 0 = 模拟比较控制器正常工作

		1 = 模拟比较控制器复位
[21]	**PWM47_RST**	**PWM47 控制器复位** 0 = PWM47 控制器正常工作 1 = PWM47 控制器复位
[20]	**PWM03_RST**	**PWM03 控制器复位** 0 = PWM03 控制器正常工作 1 = PWM03 控制器复位
[19]	**Reserved**	保留
[18]	**UART2_RST**	**UART2 控制器复位** 0 = UART2 控制器正常工作 1 = UART2 控制器复位
[17]	**UART1_RST**	**UART1控制器复位** 0 = UART1 控制器正常工作 1 = UART1 控制器复位
[16]	**UART0_RST**	**UART0 控制器复位** 0 = UART0 控制器正常工作 1 = UART0 控制器复位
[15]	**SPI3_ RST**	**SPI3 控制器复位** 0 = SPI3 控制器正常工作 1 = SPI3 控制器复位
[14]	**SPI2_ RST**	**SPI2 控制器复位** 0 = SPI2 控制器正常工作 1 = SPI2 控制器复位
[13]	**SPI1_ RST**	**SPI1 控制器复位** 0 = SPI1 控制器正常工作 1 = SPI1 控制器复位
[12]	**SPI0_ RST**	**SPI0 控制器复位** 0 = SPI0 控制器正常工作 1 = SPI0 控制器复位
[11:10]	**Reserved**	保留
[9]	**I2C1_RST**	**I^2C1 控制器复位** 0 = I^2C1 控制器正常工作 1 = I^2C1 控制器复位
[8]	**I2C0_RST**	**I^2C0 控制器复位** 0 = I^2C0 控制器正常工作 1 = I^2C0 控制器复位
[7:6]	**Reserved**	保留
[5]	**TMR3_RST**	**Timer3 控制器复位** 0 = Timer3 控制器正常工作 1 = 控制器复位

[4]	**TMR2_RST**	**Timer2 控制器复位** 0 = Timer2 控制器正常工作 1 = Timer2 控制器复位
[3]	**TMR1_RST**	**Timer1 控制器复位** 0 = Timer1 控制器正常工作 1 = Timer1 控制器复位
[2]	**TMR0_RST**	**Timer0 控制器复位** 0 = Timer0 控制器正常工作 1 = Timer0 控制器复位
[1]	**GPIO_RST**	**GPIO 控制器复位** 0 = GPIO 控制器正常工作 1 = GPIO 控制器复位
[0]	**Reserved**	保留

外设复位控制寄存器3 (IPRSTC3)

置1会产生异步复位信号给相应的控制器。用户需要将该位置0才能将相应的控制器从复位状态恢复。

寄存器	偏移地址	R/W	描述	复位值
IPRSTC3	GCR_BA+0x10	R/W	外设复位控制寄存器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					SC2_RST	SC1_RST	SC0_RST

位	描述	
[31:3]	Reserved	保留
[2]	SC2_RST	SC2 控制器复位 0 = SC2 控制器正常工作 1 = SC2 控制器复位
[1]	SC1_RST	SC1 控制器复位 0 = SC1控制器正常工作 1 = SC1控制器复位
[0]	SC0_RST	SC0 控制器复位 0 = SC0 控制器正常工作 1 = SC0 控制器复位

欠压检测控制寄存器 (BODCR)

BODCR控制寄存器的部分位在flash配置时，已经被初始化，部分位是受保护的位。编程这些写保护的位时，需要向地址0x5000_0100依次写入“59h”, “16h”, “88h”。该位操作请参考REGWRPROT（地址GCR_BA+0x100）

寄存器	偏移地址	R/W	描述	复位值
BODCR	GCR_BA+0x18	R/W	欠压检测控制寄存器	0x0000_008X

31	30	29	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**
LVR_EN	**BOD_OUT**	**BOD_LPM**	**BOD_INTF**	**BOD_RSTEN**	**BOD_VL**		**BOD_EN**

位	描述	
[31:8]	**Reserved**	保留
[7]	**LVR_EN**	**低电压复位使能位（写保护位）** 当输入电源电压低于LVR电路设置时，LVR复位芯片。低电压复位功能是默认使能的。 0 = 禁用低电压复位功能 1 = 使能低电压复位功能，使能该位100us后，低电压复位输出稳定，LVR功能生效(默认) **注:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h”。该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[6]	**BOD_OUT**	**欠压检测器输出状态位** 0 = 欠压检测器输出状态为0。表示检测到电压高于BOD_VL的设置或BOD_EN为0。 1 = 欠压检测器输出状态为1。表示检测到电压低于BOD_VL的设置. 如果 BOD_EN 是 0, BOD 功能禁用, 该位通常响应必定为0。
[5]	**BOD_LPM**	**欠压检测器低功耗模式（写保护位）** **0 = BOD 工作在正常模式（默认）** **1 = BOD工作于低功耗模式** **注1:** BOD 在正常模式下消耗电流约为100 uA ，低功耗模式下能减小到当前的约1/10, 但BOD响应速度变慢。 **注2:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h”，该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[4]	**BOD_INTF**	**欠压检测中断标志** 0 = 欠压检测没有检测到任何V_{DD} 下降或者上升到BOD_VL设定值。 1 = 当欠压检测到V_{DD}下降或者上升到BOD_VL设定电压，该位置为1，如果欠压电压中断被使能，则发生欠压中断。 注：向该位写1清零。

[3]	**BOD_RSTEN**	**欠压复位使能（写保护位）** 0 = 使能欠压“中断”功能 1 = 使能欠压“复位”功能 当同时使能欠压检测功能 (BOD_EN 为高) 和 BOD 复位功能 (BOD_RSTEN 为高), 如果检测到电压低于阈电压(BOD_OUT 为高)，BOD将发送信号复位芯片。 **注1:** 当同时使能 BOD 功能（BOD_EN 为高）和 BOD 中断功能（BOD_RSTEN 为低），如果 BOD_OUT 为高，则 BOD 将产生中断。BOD 中断将保持直到 BOD_EN 被设置为 0。可以通过禁用 NVIC BOD 中断或者禁用 BOD 功能（设置 BOD_EN 为低）封锁 BOD 中断。 **注2:** 默认值由用户配置flash 控制寄存器CBORST(CONFIG0[20]) 时设置。 **注3:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h”，该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[2:1]	**BOD_VL**	**欠压检测阈电压选择（写保护位）** 默认值由用户在配置flash控制器CBOV(CONFIG0[22:21]) 时设置. 00 = 欠压是 4.4V. 01 = 欠压是 3.7V. 10 = 欠压是 2.7V. 11 = 欠压是 2.2V. **注:**该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h”，该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[0]	**BOD_EN**	**欠压检测使能位（写保护位）** 默认值由用户在配置flash控制器CBODEN(CONFIG0[23])时设置. 0 = 禁用欠压检测功能 1 = 使能欠压检测功能 **注:**该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h”，该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)

温度传感器控制寄存器(TEMPCR)

寄存器	偏移地址	R/W	描述	复位值
TEMPCR	GCR_BA+0x1C	R/W	温度传感器控制寄存器	0x0000_0000

31	30	29	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							**VTEMP_EN**

位	描述	
[31:1]	**Reserved**	保留
[0]	**VTEMP_EN**	**温度传感器使能位** 该位用于使能/禁用温度传感器功能 0 = 禁用温度传感器功能(默认) 1 = 使能温度传感器功能 **注:** 该位置1后，温度值可以从ADC转换结果获得，这之前需要把ADC 通道选择为7，多功能复用通道选择到温度传感器选项。具体的ADC转换功能请参考ADC功能章节。

上电复位控制寄存器 (PORCR)

寄存器	偏移地址	R/W	描述	复位值
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	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**
POR_DIS_CODE[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POR_DIS_CODE[7:0]							

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**POR_DIS_CODE**	**上电复位使能位（写保护）** 上电时，POR电路产生复位信号使整个芯片复位，但是电源部分的干扰可能引起POR重新有效。用户可以将POR_DIS_CODE设置为0x5AA5，禁用POR内部电路，以免造成不可预知的干扰。 当设置POR_DIS_CODE为其他值时，或者由芯片的其他复位功能引起复位时，POR功能重新有效。这些复位功能包括： nRESET引脚复位，看门狗复位，窗口看门狗复位，LVR复位，BOD复位，ICE复位命令和软件复位。 **注:**该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h”，该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)

GPIOA 多功能引脚和输入类型控制寄存器 (GPA_MFP)

寄存器	偏移地址	R/W	描述	复位值
GPA_MFP	GCR_BA+0x30	R/W	GPIOA 多功能引脚和输入类型控制寄存器	0x0000_0000

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

位	描述	
[31:16]	**GPA_TYPEn**	**触发功能选择** 0 = 禁用 GPIOA[15:0] I/O Schmitt 触发输入 1 = 使能 GPIOA[15:0] I/O Schmitt 触发输入
[15]	**GPA_MFP15**	**PA.15 引脚功能选择** 该引脚功能取决于PA15_SC2PWR (ALT_MFP1[12]), PA15_I2SMCLK (ALT_MFP[9]) 和 GPA_MFP[15]. (PA15_SC2PWR, PA15_I2SMCLK, GPA_MFP15) 数值和功能对应如下： (0, 0, 0) = 选用GPIOA功能 (0, 0, 1) = 选用PWM3功能 (0, 1, 1) = 选用I2S_MCLK功能 (1, 0, 1) = 选用SC2_PWR功能
[14]	**GPA_MFP14**	**PA.14 引脚功能选择** 该引脚功能取决于EBI_HB_EN[7] (ALT_MFP[23]), EBI_EN (ALT_MFP[11]), PA14_SC2RST (ALT_MFP1[13]) 和 GPA_MFP[14] (EBI_HB_EN , EBI_EN, PA14_SC2RST, GPA_MFP14) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIOA功能 (0, 0, 0, 1) = 选用PWM2功能 (0, 0, 1, 1) = 选用SC2_RST功能 (1, 1, 0, 1) = 选用AD15功能

[13]	**GPA_MFP13**	**PA.13 引脚功能选择** 该引脚功能取决于EBI_HB_EN[6] (ALT_MFP[22]), EBI_EN (ALT_MFP[11]), PA13_SC2CLK (ALT_MFP1[10]) 和 GPA_MFP[13]. (EBI_HB_EN, EBI_EN, PA13_SC2CLK, GPA_MFP13) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用PWM1功能 (0, 0, 1, 1) = 选用SC2_CLK/UART5_TXD功能 (1, 1, 0, 1) = 选用AD14功能
[12]	**GPA_MFP12**	**PA.12 引脚功能选择** 该引脚功能取决于EBI_HB_EN[5] (ALT_MFP[21]), EBI_EN (ALT_MFP[11]), PA12_SC2DAT (ALT_MFP1[11]) 和GPA_MFP[12]. (EBI_HB_EN, EBI_EN, PA12_SC2DAT, GPA_MFP12) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用PWM0功能 (0, 0, 1, 1) = 选用SC2_DAT/UART5_RXD功能 (1, 1, 0, 1) = 选用AD13功能
[11]	**GPA_MFP11**	**PA.11引脚功能选择** 该引脚功能取决于EBI_EN (ALT_MFP[11]), PA10_11_CAN1 (ALT_MFP[28]) 和 GPA_MFP[11] . (EBI_EN, PA10_11_CAN1, GPA_MFP11) 数值和功能对应如下： (0,0, 0) = 选用GPIO 功能 (0,0, 1) = 选用I2C1_SCL 功能 (0, 1, 1) = 选用CAN1_RXD 功能 (1, 0, 1) = 选用nRD(EBI) 功能
[10]	**GPA_MFP10**	**PA.10引脚功能选择** 引脚功能取决于EBI_EN (ALT_MFP[11]), PA10_11_CAN1 (ALT_MFP[28]) 和 GPA_MFP[10] (EBI_EN, PA10_11_CAN1, GPA_MFP10) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能. (0, 0, 1) = 选用I2C1_SDA功能 (0, 1, 1) = 选用CAN1_TXD功能 (1, 0, 1) = 选用nWR(EBI)功能
[9]	**GPA_MFP9**	**PA.9 Pin引脚功能选择** 该引脚功能取决于Bit GPA_MFP[9] 0 = 选用GPIO 功能 1 = 选用I2C0_SCL 功能
[8]	**GPA_MFP8**	**PA.8 引脚功能选择** 该引脚功能取决于Bit GPA_MFP[8] 0 = 选用GPIO功能 1 = 选用I2C0_SDA功能

[7]	**GPA_MFP7**	**PA.7 引脚功能选择** 引脚功能取决于EBI_EN (ALT_MFP[11]), PA7_SC1DAT (ALT_MFP1[6]), PA7_S21 (ALT_MFP[2]) 和 GPA_MFP[7] . 数值和功能对应如下: (0, 0, 0, 0) = 选用GPIO功能. (0, 0, 0, 1) = 选用ADC7功能. (0, 0,1, 1) = 选用SPI2_SS1. (0, 1, 0, 1) = 选用SC1_DAT\UART4_RXD 功能. (1, 0, 0, 1) = 选用AD6 功能.
[6]	**GPA_MFP6**	**PA. 6 引脚功能选择** 引脚功能取决于EBI_EN (ALT_MFP[11]), PA6_SC1CLK (ALT_MFP1[5]) 和GPA_MFP[6]. 数值和功能对应如下: (0, 0, 0) = 选用GPIO功能. (0, 0, 1) = 选用ADC6功能. (0, 1, 1) = 选用SC1_CLK/UART4_TXD 功能. (1, 0, 1) = 选用AD7功能.
[5]	**GPA_MFP5**	**PA.5引脚功能选择** 引脚功能取决于EBI_HB_EN[0] (ALT_MFP[16]), EBI_EN (ALT_MFP[11]),PA5_SC1RST (ALT_MFP1[8]) 和GPA_MFP[5] . 数值和功能对应如下: (0, 0, 0, 0) = 选用GPIO功能. (0, 0, 0, 1) = 选用ADC5功能. (0, 0,1, 1) = 选用SC1_RST功能. (1, 1, 0, 1) = 选用AD8 功能.
[4]	**GPA_MFP4**	**PA.4 引脚功能选择** 该引脚功能取决于EBI_HB_EN[1] (ALT_MFP[17]), EBI_EN (ALT_MFP[11]), PA4_SC1PWR (ALT_MFP1[7]) 和 GPA_MFP[4] . 数值和功能对应如下: (0, 0, 0, 0) = 选用GPIO 功能. (0, 0, 0, 1) = 选用ADC4 功能. (0, 0, 1, 1) = 选用SC1_PWR 功能. (1, 1, 0, 1) = 选用AD9功能.
[3]	**GPA_MFP3**	**PA.3 引脚功能选择** 该引脚功能取决于EBI_HB_EN[2] (ALT_MFP[18]), EBI_EN (ALT_MFP[11]), PA3_SC0DAT (ALT_MFP1[1]) 和GPA_MFP[3] . 数值和功能对应如下: (0, 0, 0, 0) = 选用GPIO功能. (0, 0, 0, 1) = 选用ADC3功能. (0, 0, 1, 1) = 选用SC0_DAT/UART3_RXD 功能. (1, 1, 0, 1) = 选用AD10功能.

[2]	**GPA_MFP2**	**PA.2引脚功能选择** 该引脚功能取决于EBI_HB_EN[3] (ALT_MFP[19]), EBI_EN (ALT_MFP[11]), PA2_SC0CLK (ALT_MFP1[0]) 和GPA_MFP[2]. 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能. (0, 0, 0, 1) = 选用ADC2功能. (0, 0, 1, 1) = 选用SC0_CLK/UART3_TXD功能. (1, 1, 0, 1) = 选用AD11功能.
[1]	**GPA_MFP1**	**PA.1引脚功能选择** 该引脚功能取决于EBI_HB_EN[4] (ALT_MFP[20]), EBI_EN (ALT_MFP[11]), PA1_SC0RST (ALT_MFP1[3]) 和GPA_MFP[1]. 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能. (0, 0, 0, 1) = 选用ADC1功能. (0, 0, 1, 1) = 选用SC0_RST功能. (1, 1, 0, 1) = 选用AD12功能.
[0]	**GPA_MFP0**	**PA.0引脚功能选择** 该引脚功能取决于PA0_SC0PWR (ALT_MFP1[2]) 和GPA_MFP[0] (PA0_SC0PWR, GPA_MFP0) 数值和功能对应如下： (0, 0) =选用GPIO功能 (0, 1) =选用ADC0功能 (1, 1) =选用SC0_PWR功能

GPIOB多功能引脚和输入类型控制寄存器(GPB_MFP)

寄存器	偏移地址	R/W	描述	复位值
GPB_MFP	GCR_BA+0x34	R/W	GPIOB多功能引脚和输入类型控制寄存器	0x0000_0000

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

位	描述	
[31:16]	GPB_TYPEn	**触发功能选择** 0 = 禁用GPIOB[15:0] I/O Schmitt 触发输入 1 = 使能GPIOB[15:0] I/O Schmitt 触发输入
[15]	GPB_MFP15	**PB.15引脚功能选择** 该引脚功能取决于PB14_15_EBI (ALT_MFP2[1]), PB15_T0EX (ALT_MFP[24]), PB15_TM0 (ALT_MFP2[2]) 和GPB_MFP[15] (PB14_15_EBI, PB15_T0EX, PB15_TM0, GPB_MFP15) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用INT1功能 (0 ,0, 1, 1) = 选用TM0功能 (0, 1, 0, 1) = 选用TM0_EXT功能 (1, 0, 0, 1) = 选用AD6功能
[14]	GPB_MFP14	**PB.14引脚功能选择** 该引脚功能取决于PB14_15_EBI (ALT_MFP2[1]), PB14_S31 (ALT_MFP[3]) 和GPB_MFP[14] (PB14_15_EBI , PB14_S31, GPB_MFP14) 数值和功能对应如下： (0, 0, 0) = 选用GPIO 功能 (0, 0, 1) = 选用INT0 功能 (0, 1, 1) = 选用SPI3_SS1 功能. (1, 0, 1) = 选用AD0 功能
[13]	GPB_MFP13	**PB.13引脚功能选择** 该引脚功能取决于EBI_EN (ALT_MFP[11]), GPB_MFP[13] (EBI_EN, GPB_MFP13) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用ACMP1_O功能 (1, 1) = 选用AD1功能

[12]	**GPB_MFP12**	保留
[11]	**GPB_MFP11**	**PB.11引脚功能选择** 该引脚功能取决于PB11_PWM4 (ALT_MFP[4]) 和GPB_MFP[11] (PB11_PWM4, GPB_MFP11) 数值和功能对应如下： (0, 0) = 选用GPIO 功能 (0, 1) = 选用TM3 功能 (1, 1) = 选用PWM4 功能
[10]	**GPB_MFP10**	**PB.10引脚功能选择** 该引脚功能取决于PB10_S01 (ALT_MFP[0]) 和 GPB_MFP[10] (PB10_S01, GPB_MFP10) 数值和功能对应如下： (0, 0) = 选用GPIO 功能 (0, 1) = 选用TM2 功能 (1, 1) = 选用SPI0_SS1 功能
[9]	**GPB_MFP9**	**PB.9 引脚功能选择** 该引脚功能取决于PB9_S11 (ALT_MFP[1]) 和GPB_MFP[9] (PB9_S11, GPB_MFP9) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用TM1功能 (1, 1) = 选用SPI1_SS1功能
[8]	**GPB_MFP8**	**PB.8 引脚功能选择** 该引脚功能取决于 PB8_CLKO (ALT_MFP[29]) 和 GPB_MFP[8] (PB8_CLKO, GPB_MFP8) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用TM0功能 (1, 1) = 选用CLKO功能
[7]	**GPB_MFP7**	**PB.7引脚功能选择** 该引脚功能取决于EBI_EN (ALT_MFP[11]), GPB_MFP[7] (EBI_EN, GPB_MFP7) 数值和功能对应如下： (0, 0) =选用GPIO 功能 (0, 1) =选用UART1_nCTS 功能 (1, 1) =选用nCS(EBI) 功能
[6]	**GPB_MFP6**	**PB.6引脚功能选择** 该引脚功能取决于 EBI_EN (ALT_MFP[11]), GPB_MFP[6] (EBI_EN, GPB_MFP6) 数值和功能对应如下： (0, 0) =选用GPIO功能 (0, 1) =选用UART1_nRTS功能 (1, 1) = 选用ALE(EBI) 功能
[5]	**GPB_MFP5**	**PB 5引脚功能选择** 该引脚功能取决于GPB_MFP[5] 0 = 选用GPIO 功能 1 = 选用UART1_TXD 功能
[4]	**GPB_MFP4**	**PB.4引脚功能选择**

		该引脚功能取决于GPB_MFP[4] 0 = 选用GPIO功能 1 = 选用UART1_RXD功能
[3]	**GPB_MFP3**	**PB.3引脚功能选择** 该引脚功能取决于EBI_nWRH_EN (ALT_MFP[14]), EBI_EN (ALT_MFP[11]), PB3_TM3 (ALT_MFP2[5]), PB3_SC2CD (ALT_MFP1[14]), PB3_T3EX (ALT_MFP[27]) 和GPB_MFP[3] (EBI_nWRH_EN, EBI_EN, PB3_TM3, PB3_SC2CD, PB3_T3EX, GPB_MFP3) 数值和功能对应如下： (0, 0, 0, 0, 0, 0) = 选用GPIO 功能 (0, 0, 0, 0, 0, 1) = 选用UART0_nCTS功能. (0, 0, 0, 0, 1, 1) = 选用TM3_EXT 功能 (0, 0, 0, 1, 0, 1) = 选用SC2_CD 功能 (0, 0, 1, 0, 0, 1) = 选用TM3 功能 (1, 1, 0, 0, 0, 1) = 选用WRH(EBI)功能
[2]	**GPB_MFP2**	**PB.2 引脚功能选择** 该引脚功能取决于EBI_nWRL_EN (ALT_MFP[13]), EBI_EN (ALT_MFP[11]), PB2_TM2 (ALT_MFP2[4]), PB2_CPO0 (ALT_MFP[30]), PB2_T2EX (ALT_MFP[26]) 和GPB_MFP[2] (EBI_nWRL_EN, EBI_EN, PB2_TM2, PB2_CPO0, PB2_T2EX, GPB_MFP2) 数值和功能对应如下： (0, 0, 0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 0, 0, 1) = 选用UART0_nRTS功能 (0, 0, 0, 0, 1, 1) = 选用TM2_EXT功能 (0, 0, 0, 1, 0, 1) = 选用ACMP0_O功能 (0, 0, 1, 0, 0, 1) = 选用TM2功能 (1, 1, 0, 0, 0, 1) = 选用nWRL(EBI)功能
[1]	**GPB_MFP1**	**PB.1引脚功能选择** 该引脚功能取决于GPB_MFP[1] 0 = 选用GPIO 功能 1 = 选用UART0_TXD 功能
[0]	**GPB_MFP0**	**PB.0引脚功能选择** 该引脚功能取决于GPB_MFP[0] 0 = 选用GPIO功能 1 = 选用UART0_RXD功能

GPIOC 多功能引脚和输入类型控制寄存器 (GPC_MFP)

寄存器	偏移地址	R/W	描述	复位值
GPC_MFP	GCR_BA+0x38	R/W	GPIOC 多功能引脚和输入类型控制寄存器	0x0000_0000

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

位	描述	
[31:16]	GPC_TYPEn	**触发功能选择** 0 = 禁用GPIOC[15:0] I/O Schmitt 触发输入 1 = 使能GPIOC[15:0] I/O Schmitt 触发输入
[15]	GPC_MFP15	**PC.15引脚功能选择** 该引脚功能取决于EBI_EN (ALT_MFP[11]) 和GPC_MFP[15] (EBI_EN, GPC_MFP15) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用ACMP1_N功能 (1, 1) = 选用AD3功能
[14]	GPC_MFP14	**PC.14引脚功能选择** 该引脚功能取决于EBI_EN (ALT_MFP[11]) 和GPC_MFP[14] (EBI_EN, GPC_MFP14) 数值和功能对应如下： (0, 0) = 选用GPIO 功能 (0, 1) = 选用ACMP1_P 功能 (1, 1) = 选用AD2功能
[13]	GPC_MFP13	**PC.13引脚功能选择** 该引脚功能取决于GPC_MFP[13] 0 = 选用GPIO 功能 1 = 选用SPI1_MOSI1 功能
[12]	GPC_MFP12	**PC.12引脚功能选择** 该引脚功能取决于 GPC_MFP[12] 0 = 选用GPIO 功能 1 = 选用SPI1_MISO1 功能

[11]	**GPC_MFP11**	**PC.11引脚功能选择** 该引脚功能取决于GPC_MFP[11] 0 = 选用GPIO 功能 1 = 选用SPI1_MOSI0 功能
[10]	**GPC_MFP10**	**PC.10引脚功能选择** 该引脚功能取决于GPC_MFP[10] 0 = 选用GPIO 功能 1 = 选用SPI1_MISO0 功能
[9]	**GPC_MFP9**	**PC.9引脚功能选择** 该引脚功能取决于GPC_MFP[9] 0 = 选用GPIO 功能 1 = 选用SPI1_CLK 功能
[8]	**GPC_MFP8**	**PC.8引脚功能选择** 该引脚功能取决于EBI_MCLK_EN (ALT_MFP[12]), EBI_EN (ALT_MFP[11]), GPC_MFP[8] (EBI_MCLK_EN, EBI_EN, GPC_MFP8) 数值和功能对应如下： (0, 0, 0) = 选用GPIO 功能. (0, 0 ,1) = 选用SPI1_SS0 功能 (1, 1, 1) = 选用MCLK(EBI) 功能
[7]	**GPC_MFP7**	**PC.7引脚功能选择** 该引脚功能取决于EBI_EN (ALT_MFP[11]), PC7_SC1CD (ALT_MFP1[9]) 和GPC_MFP[7]. (EBI_EN, PC7_SC1CD, GPC_MFP7) 数值和功能对应如下： (0, 0, 0) = 选用GPIO 功能 (0, 0, 1) = 选用ACMP0_N 功能 (0, 1, 1) = 选用SC1_CD 功能 (1, 0, 1) = 选用AD5功能
[6]	**GPC_MFP6**	**PC.6引脚功能选择** 该引脚功能取决于EBI_EN (ALT_MFP[11]). PC6_SC0CD (ALT_MFP1[4]) (EBI_EN, PC6_SC0CD, GPB_MFP6) 数值和功能对应如下： (0, 0, 0) =选用GPIO功能 (0, 0, 1) =选用ACMP0_P功能 (0, 1, 1) =选用SC0_CD功能 (1, 0, 1) =选用AD4功能
[5]	**GPC_MFP5**	**PC.5引脚功能选择** 该引脚功能取决于GPC_MFP[9] 0 = 选用GPIO 功能 1 = 选用SPI0_MOSI1 功能
[4]	**GPC_MFP4**	**PC.4引脚功能选择** 该引脚功能取决于 GPC_MFP[9] 0 = 选用GPIO 功能 1 = 选用SPI0_MISO1 功能

[3]	**GPC_MFP3**	**PC.3引脚功能选择** 该引脚功能取决于 PC3_I2SDO (ALT_MFP[8]) 和 GPC_MFP[3] (PC3_I2SDO, GPC_MFP3) 数值和功能对应如下： (0, 0) = 选用GPIO 功能 (0, 1) = 选用SPI0_MOSI0 功能 (1, 1) = 选用I2S_DO 功能
[2]	**GPC_MFP2**	**PC.2引脚功能选择** 该引脚功能取决于 PC2_I2SDI (ALT_MFP[7]) and GPC_MFP[2] (PC2_I2SDI, GPC_MFP2) 数值和功能对应如下： (0, 0) = 选用GPIO 功能 (0, 1) = 选用SPI0_MISO0 功能 (1, 1) = 选用I2S_DI 功能
[1]	**GPC_MFP1**	**PC.1引脚功能选择** 该引脚功能取决于 PC1_I2SBCLK (ALT_MFP[6]) 和GPC_MFP[1] (PC1_I2SBCLK, GPC_MFP1) 数值和功能对应如下： (0, 0) = 选用GPIO 功能 (0, 1) = 选用SPI0_CLK 功能 (1, 1) = 选用I2S_BCLK 功能
[0]	**GPC_MFP0**	**PC.0引脚功能选择** 该引脚功能取决于 PC0_I2SLRCLK (ALT_MFP[5]) 和GPC_MFP[0] (PC0_I2SLRCLK, GPC_MFP0) 数值和功能对应如下： (0, 0) = 选用GPIO 功能 (0, 1) = 选用SPI0_SS0 功能 (1, 1) = 选用I2S_LRCK 功能

GPIOD多功能引脚和输入类型控制寄存器(GPD_MFP)

寄存器	偏移地址	R/W	描述	复位值
GPD_MFP	GCR_BA+0x3C	R/W	GPIOD多功能引脚和输入类型控制寄存器	0x0000_0000

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

位	描述	
[31:16]	**GPD_TYPEn**	**触发功能选择** 0 = 禁用GPIOD[15:0] I/O Schmitt 触发输入 1 = 使能GPIOD[15:0] I/O Schmitt 触发输入
[15]	**GPD_MFP15**	**PD.15引脚功能选择** 该引脚功能取决于PD14_15_CAN1 (ALT_MFP2[0]) 和 GPD_MFP[15] (PD14_15_CAN1, GPD_MFP15) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用UART2_TXD功能 (1, 1) = 选用CAN1_TXD功能
[14]	**GPD_MFP14**	**PD.14引脚功能选择** 该引脚功能取决于PD14_15_CAN1 (ALT_MFP2[0]) 和GPD_MFP[14] (PD14_15_CAN1, GPD_MFP14) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用UART2_RXD功能 (1, 1) = 选用CAN1_RXD功能
[13]	**GPD_MFP13**	**PD.13引脚功能选择** 该引脚功能取决于GPD_MFP[13] 0 = 选用GPIO功能 1 = 选用SPI3_MOSI1功能
[12]	**GPD_MFP12**	**PD.12引脚功能选择** 该引脚功能取决于GPD_MFP[12] 0 = 选用GPIO功能 1 = 选用SPI3_MISO1功能
[11]	**GPD_MFP11**	**PD.11引脚功能选择**

		该引脚功能取决于GPD_MFP[11] 0 = 选用GPIO功能 1 = 选用SPI3_MOSI0功能
[10]	**GPD_MFP10**	**PD.10引脚功能选择** 该引脚功能取决于GPD_MFP[10] 0 = 选用GPIO功能 1 = 选用SPI3_MISO0功能
[9]	**GPD_MFP9**	**PD.9引脚功能选择** 该引脚功能取决于GPD_MFP[9] 0 = 选用GPIO功能 1 = 选用SPI3_CLK功能
[8]	**GPD_MFP8**	**PD.8引脚功能选择** 该引脚功能取决于GPD_MFP[8] 0 = 选用GPIO功能 1 = 选用SPI3_SS0功能
[7]	**GPD_MFP7**	**PD.7引脚功能选择** 该引脚功能取决于GPD_MFP[7] 0 = 选用 GPIO功能 1 = 选用 CAN0_TXD功能
[6]	**GPD_MFP6**	**PD.6引脚功能选择** 该引脚功能取决于GPD_MFP[9] 0 = 选用 GPIO功能 1 = 选用 CAN0_RXD功能
[5]	**GPD_MFP5**	**PD.5引脚功能选择** 该引脚功能取决于GPD_MFP[9] 0 = 选用GPIO功能 1 = 选用SPI2_MOSI1功能
[4]	**GPD_MFP4**	**PD.4引脚功能选择** 该引脚功能取决于GPD_MFP[4] 0 = 选用GPIO 功能 1 = 选用SPI2_MISO1 功能
[3]	**GPD_MFP3**	**PD.3引脚功能选择** 该引脚功能取决于GPD_MFP[3] 0 = 选用GPIO功能 1 = 选用SPI2_MOSI0功能
[2]	**GPD_MFP2**	**PD.2引脚功能选择** 该引脚功能取决于GPD_MFP[2] 0 = 选用GPIO功能 1 = 选用SPI2_MISO0功能
[1]	**GPD_MFP1**	**PD.1引脚功能选择** 该引脚功能取决于GPD_MFP[1]

		0 = 选用GPIO功能 1 = 选用SPI2_CLK功能
[0]	**GPD_MFP0**	**PD.0引脚功能选择** 该引脚功能取决于GPD_MFP[0] 0 = 选用GPIO功能 1 = 选用SPI2_SS0功能

GPIOE 多功能引脚和输入类型控制寄存器(GPE_MFP)

寄存器	偏移地址	R/W	描述	复位值
GPE_MFP	GCR_BA+0x40	R/W	GPIOE 多功能引脚和输入类型控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
GPE_TYPE[15:8]							
23	22	21	20	19	18	17	16
GPE_TYPE[7:0]							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		GPE_MFP5	Reserved			GPE_MFP1	GPE_MFP0

位	描述	
[31:16]	GPE_TYPEn	**触发功能选择** 0 = 禁用GPIOE[15:0] I/O Schmitt 触发输入 1 = 使能GPIOE [15:0] I/O Schmitt 触发输入
[15:6]	Reserved	保留
[5]	GPE_MFP5	**PE.5引脚功能选择** 该引脚功能取决于 PE5_T1EX (ALT_MFP[25]), PE5_TM1 (ALT_MFP2[3]) 和 GPE_MFP5 (PE5_T1EX, GPE_MFP5) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用PWM5功能 (1, 0, 1) = 选用TM1_EXT功能 (0, 1, 1) = 选用TM1功能
[4:2]	Reserved	保留
[1]	GPE_MFP1	**PE.1引脚功能选择** 该引脚功能取决于GPE_MFP[1] 0 = 选用GPIO功能 1 = 选用PWM7功能
[0]	GPE_MFP0	**PE.0引脚功能选择** 该引脚功能取决于GPE_MFP[0] 0 = 选用GPIO功能 1 = 选用PWM6功能

GPIOF多功能引脚和输入类型控制寄存器(GPF_MFP)

寄存器	偏移地址	R/W	描述	复位值
GPF_MFP	GCR_BA+0x44	R/W	GPIOF多功能引脚和输入类型控制寄存器	0x0000_000X

注：GPF_MFP[3]/GPF_MFP[2]默认值为1. GPF_MFP[1]/GPF_MFP[0]默认值由用户在CGPFMFP(CONFIG0[27]配置。

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				GPF_TYPE[3:0]			
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				GPF_MFP3	GPF_MFP2	GPF_MFP1	GPF_MFP0

位	描述	
[31:20]	**Reserved**	保留
[19:16]	**GPF_TYPEn**	**触发功能选择** 0 = 禁用GPIOF[3:0] I/O Schmitt 触发输入 1 = 使能GPIOF [3:0] I/O Schmitt 触发输入
[15:4]	**Reserved**	保留
[3]	**GPF_MFP3**	**PF.3引脚功能选择** 该引脚功能取决于GPF_MFP[3] 0 = 选用GPIO 功能 1 = 选用PS/2_CLK 功能
[2]	**GPF_MFP2**	**PF.2引脚功能选择** 该引脚功能取决于GPF_MFP[2] 0 = 选用GPIO 功能 1 = 选用PS/2_DAT 功能
[1]	**GPF_MFP1**	**PF.1引脚功能选择** 该引脚功能取决于GPF_MFP[1] 0 = 选用GPIO 功能 1 = 选用XT1_IN 功能 **注:**该位只读，用户在CGPFMFP (CONFIG0[27])配置
[0]	**GPF_MFP0**	**PF.0引脚功能选择** 该引脚功能取决于GPF_MFP[0] 0 = 选用GPIO 功能 1 = 选用XT1_OUT 功能 **注:**该位只读，用户在CGPFMFP (CONFIG0[27])配置

复用多功能引脚控制寄存器 (ALT_MFP)

寄存器	偏移地址	R/W	描述	复位值
ALT_MFP	GCR_BA+0x50	R/W	复用多功能引脚控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	PB2_CPO0	PB8_CLKO	PA10_11_CAN1	PB3_T3EX	PB2_T2EX	PE5_T1EX	PB15_T0EX
23	22	21	20	19	18	17	16
EBI_HB_EN							
15	14	13	12	11	10	9	8
Reserved	EBI_nWRH_EN	EBI_nWRL_EN	EBI_MCLK_EN	EBI_EN	Reserved	PA15_I2SMCLK	PC3_I2SDO
7	6	5	4	3	2	1	0
PC2_I2SDI	PC1_I2SBCLK	PC0_I2SLRCLK	PB11_PWM4	PB14_S31	PA7_S21	PB9_S11	PB10_S01

位	描述	
[31]	Reserved	保留
[30]	PB2_CPO0	**PB.2 复用多功能引脚功能选择** PB2_TM2 (ALT_MFP2[4]), PB2_CPO0 (ALT_MFP[30]), PB2_T2EX (ALT_MFP[26]) 和 GPB_MFP[2] 决定了PB.2 的功能。 (PB2_TM2, PB2_CPO0, PB2_T2EX, GPB_MFP2) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用UART0_nRTS功能 (0, 0, 1, 1) = 选用TM2_EXT功能 (0, 1, 0, 1) = 选用ACMP0_O功能 (1, 0, 0, 1) = 选用TM2功能
[29]	PB8_CLKO	**PB.8 Pin复用多功能引脚功能选择** PB8_CLKO (ALT_MFP[29]) 和GPB_MFP[8] 决定了PB.8 的功能。 (PB8_CLKO, GPB_MFP8) 数值和功能对应如下： (0, 0) = 选用GPIO 功能 (0, 1) = 选用TM0 function 功能 (1, 1) = 选用CLKO 功能

[28]	**PA10_11_CAN1**	**PA.10 和 PA.11复用多功能引脚功能选择** PA10_11_CAN1 (ALT_MFP[28])和GPA_MFP[11] 决定了PA.11的功能。 (PA10_11_CAN1, GPA_MFP11) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用I2C1_SCL功能 (1, 1) = 选用CAN1_RXD功能 PA10_11_CAN1 (ALT_MFP[28]) 和GPA_MFP[10] 决定了PA.10的功能。 (PA10_11_CAN1, GPA_MFP10) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用I2C1_SDA功能 (1, 1) = 选用CAN1_TXD功能
[27]	**PB3_T3EX**	**PB.3复用多功能引脚功能选择** PB3_TM3 (ALT_MFP2[5]), PB3_SC2CD (ALT_MFP1[14]), PB3_T3EX (ALT_MFP[27]) 和 GPB_MFP[3] 决定了PB.3的功能。 (PB3_TM3, PB3_SC2CD, PB3_T3EX, GPB_MFP3) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO 功能 (0, 0, 0, 1) = 选用UART0_nCTS 功能 (0, 0, 1, 1) = 选用TM3_EXT 功能 (0, 1, 0, 1) = 选用SC2_CD 功能 (1, 0, 0, 1) = 选用TM3 功能
[26]	**PB2_T2EX**	**PB.2复用多功能引脚功能选择** PB2_TM2 (ALT_MFP2[4])**,** PB2_CPO0 (ALT_MFP[30]), PB2_T2EX (ALT_MFP[26]) 和 GPB_MFP[2] 决定了PB.2 的功能。 (PB2_TM2, PB2_CPO0, PB2_T2EX, GPB_MFP2) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用UART0_nRTS 功能 (0, 0, 1, 1) = 选用TM2_EXT 功能 (0, 1, 0, 1) = 选用ACMP0_O 功能 (1, 0, 0, 1) = 选用TM2 功能
[25]	**PE5_T1EX**	**PE.5复用多功能引脚功能选择** PE5_T1EX (ALT_MFP[25]), PE5_TM1 (ALT_MFP2[3]) 和 GPE_MFP5 决定了PE.5 的功能。 (PE5_T1EX, GPE_MFP5) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用PWM5功能 (1, 0, 1) = 选用TM1_EXT功能 (0, 1, 1) = 选用TM1功能

[24]	**PB15_T0EX**	**PB.15复用多功能引脚功能选择** PB14_15_EBI (ALT_MFP2[1]), PB15_T0EX (ALT_MFP[24]), PB15_TM0 (ALT_MFP2[2]) 和 GPB_MFP[15] 决定了 PB.15 的功能。 (PB14_15_EBI, PB15_T0EX, PB15_TM0, GPB_MFP15) 数值和功能对应如下： (0, 0, 0, 0) =选用GPIO功能 (0, 0, 0, 1) =选用INT1 功能 (0 ,0, 1, 1) =选用TM0 功能 (0, 1, 0, 1) =选用TM0_EXT 功能 (1, 0, 0, 1) =选用AD6 功能
[23]	**EBI_HB_EN[7]**	EBI_HB_EN[7] (ALT_MFP[23]), EBI_EN (ALT_MFP[11]), PA14_SC2RST (ALT_MFP1[13]) 和 GPA_MFP[14] 决定了 PA.14 的功能。 (EBI_HB_EN, EBI_EN, PA14_SC2RST, GPA_MFP14) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用PWM2功能 (0, 0, 1, 1) = 选用SC2_RST功能 (1, 1, 0, 1) = 选用AD15功能
[22]	**EBI_HB_EN[6]**	EBI_HB_EN[6] (ALT_MFP[22]), EBI_EN (ALT_MFP[11]), PA13_SC2CLK (ALT_MFP1[10]) 和 GPA_MFP[13] 决定了PA.13的功能。 (EBI_HB_EN, EBI_EN, PA13_SC2CLK, GPA_MFP13) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用PWM1功能 (0, 0, 1, 1) = 选用SC2_CLK/UART5_TXD功能 (1, 1, 0, 1) = 选用AD14功能
[21]	**EBI_HB_EN[5]**	EBI_HB_EN[5] (ALT_MFP[21]), EBI_EN (ALT_MFP[11]), PA12_SC2DAT (ALT_MFP1[11]) 和 GPA_MFP[12] 决定了 PA.12 的功能。 (EBI_HB_EN, EBI_EN, PA12_SC2DAT, GPA_MFP12) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用PWM0功能 (0, 0, 1, 1) = 选用SC2_DAT/UART5_RXD功能 (1, 1, 0, 1) = 选用AD13功能
[20]	**EBI_HB_EN[4]**	EBI_HB_EN[4] (ALT_MFP[20]), EBI_EN (ALT_MFP[11]), PA1_SC0RST (ALT_MFP1[3]) 和 GPA_MFP[1] 决定了 PA.1 的功能。 (EBI_HB_EN, EBI_EN, PA1_SC0RST, GPA_MFP1) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC1功能 (0, 0, 1, 1) = 选用SC0_RST功能 (1, 1, 0, 1) = 选用AD12功能
[19]	**EBI_HB_EN[3]**	EBI_HB_EN[3] (ALT_MFP[19]), EBI_EN (ALT_MFP[11]), PA2_SC0CLK (ALT_MFP1[0]) 和 GPA_MFP[2] 决定了 PA.2 的功能。 (EBI_HB_EN, EBI_EN, PA2_SC0CLK, GPA_MFP2) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC2功能 (0, 0, 1, 1) = 选用SC0_CLK/UART3_TXD 功能 (1, 1, 0, 1) = 选用AD11功能

[18]	**EBI_HB_EN[2]**	EBI_HB_EN[2] (ALT_MFP[18]), EBI_EN (ALT_MFP[11]), PA3_SC0DAT (ALT_MFP1[1]) 和 GPA_MFP[3]决定了 PA.3 的功能。 (EBI_HB_EN, EBI_EN PA3_SC0DAT, GPA_MFP3) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC3功能 (0, 0, 1, 1) = 选用SC0_DAT/UART3_RXD功能 (1, 1, 0, 1) = 选用AD10功能
[17]	**EBI_HB_EN[1]**	EBI_HB_EN[1] (ALT_MFP[17]), EBI_EN (ALT_MFP[11]), PA4_SC1PWR (ALT_MFP1[7]) 和 GPA_MFP[4] 决定了PA.4 的功能。 (EBI_HB_EN, EBI_EN, PA4_SC1PWR, GPA_MFP4) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC4功能 (0, 0, 1, 1) = 选用SC1_PWR功能 (1, 1, 0, 1) = 选用AD9功能
[16]	**EBI_HB_EN[0]**	EBI_HB_EN[0] (ALT_MFP[16]), EBI_EN (ALT_MFP[11]),PA5_SC1RST (ALT_MFP1[8]) and GPA_MFP[5] 决定了 PA.5 的功能。 (EBI_HB_EN, EBI_EN, PA5_SC1RST, GPA_MFP5) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC5功能 (0, 0, 1, 1) = 选用SC1_RST功能 (1, 1, 0, 1) = 选用AD8功能
[15]	**Reserved**	Reserved
[14]	**EBI_nWRH_EN**	EBI_nWRH_EN (ALT_MFP[14]), EBI_EN (ALT_MFP[11]), PB3_TM3 (ALT_MFP2[5]), PB3_SC2CD (ALT_MFP1[14]), PB3_T3EX (ALT_MFP[27]) 和 GPB_MFP[3] 决定了PB.3 的功能。 (EBI_nWRH_EN, EBI_EN, PB3_TM3, PB3_SC2CD, PB3_T3EX, GPB_MFP3) 数值和功能对应如下： (0, 0, 0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 0, 0, 1) = 选用UART0_nCTS功能 (0, 0, 0, 0, 1, 1) = 选用TM3_EXT功能 (0, 0, 0, 1, 0, 1) = 选用SC2_CD功能 (0, 0, 1, 0, 0, 1) = 选用TM3功能 (1, 1, 0, 0, 0, 1) = 选用nWRH(EBI)功能
[13]	**EBI_nWRL_EN**	EBI_nWRL_EN (ALT_MFP[13]), EBI_EN (ALT_MFP[11]), PB2_TM2 (ALT_MFP2[4]), PB2_CPO0 (ALT_MFP[30]), PB2_T2EX (ALT_MFP[26]) 和 GPB_MFP[2] 决定了 PB.2 的功能。 (EBI_nWRL_EN, EBI_EN, PB2_TM2, PB2_CPO0, PB2_T2EX, GPB_MFP2) 数值和功能对应如下： (0, 0, 0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 0, 0, 1) = 选用UART0_nRTS功能 (0, 0, 0, 0, 1, 1) = 选用TM2_EXT功能 (0, 0, 0, 1, 0, 1) = 选用ACMP0_O功能 (0, 0, 1, 0, 0, 1) = 选用TM2 功能 (1, 1, 0, 0, 0, 1) = 选用nWRL(EBI)功能

[12]	**EBI_MCLK_EN**	EBI_MCLK_EN (ALT_MFP[12]), EBI_EN (ALT_MFP[11]) 和 GPC_MFP[8]决定了PC.8 的功能。 (EBI_MCLK_EN, EBI_EN, GPC_MFP8) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0 ,1) = 选用SPI1_SS0功能 (1, 1, 1) = 选用MCLK(EBI)功能
[11]	**EBI_EN**	EBI_EN是用来切换GPIO功能到EBI功能的(AD[15:0], ALE, RE, WE, CS, MCLK)。它需要额外的寄存器EBI_EN[7:0]和EBI_MCLK_EN来为某些GPIO实现切换到EBI功能(AD[15:8], MCLK)。
[10]	**Reserved**	Reserved
[9]	**PA15_I2SMCLK**	**PA.15复用多功能引脚功能选择** PA15_SC2PWR (ALT_MFP1[12]), PA15_I2SMCLK (ALT_MFP[9]) 和GPA_MFP[15] 决定了 The PA.15 的功能。 (PA15_SC2PWR, PA15_I2SMCLK, GPA_MFP15) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用PWM3功能 (0, 1, 1) = 选用I2S_MCLK功能 (1, 0, 1) = 选用SC2_PWR功能
[8]	**PC3_I2SDO**	**PC.3复用多功能引脚功能选择** PC3_I2SDO (ALT_MFP[8]) 和 GPC_MFP[3] 决定了 PC.3 的功能。 (PC3_I2SDO, GPC_MFP3) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用SPI0_MOSI0功能 (1, 1) = 选用I2S_DO功能
[7]	**PC2_I2SDI**	**PC.2复用多功能引脚功能选择** PC2_I2SDI (ALT_MFP[7]) 和GPC_MFP[2] 决定了PC.2的功能。 (PC2_I2SDI, GPC_MFP2) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用SPI0_MISO0功能 (1, 1) = 选用I2S_DI功能
[6]	**PC1_I2SBCLK**	**PC.1复用多功能引脚功能选择** PC1_I2SBCLK (ALT_MFP[6]) 和 GPC_MFP[1] 决定了PC.1 的功能。 (PC1_I2SBCLK, GPC_MFP1) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用SPI0_CLK功能 (1, 1) = 选用I2S_BCLK功能
[5]	**PC0_I2SLRCLK**	**PC.0复用多功能引脚功能选择** PC0_I2SLRCLK (ALT_MFP[5]) 和 GPC_MFP[0] 决定了PC.0 的功能。 (PC0_I2SLRCLK, GPC_MFP0) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用SPI0_SS0功能 (1, 1) = 选用I2S_LRCLK功能

[4]	**PB11_PWM4**	**PB.11复用多功能引脚功能选择** PB11_PWM4 (ALT_MFP[4]) 和 GPB_MFP[11] 决定了 PB.11 的功能。 (PB11_PWM4, GPB_MFP11) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用TM3功能 (1, 1) = 选用PWM4功能
[3]	**PB14_S31**	**PB.14复用多功能引脚功能选择** PB14_15_EBI (ALT_MFP2[1]), PB14_S31 (ALT_MFP[3]) 和 GPB_MFP[14] 决定了PB.14的功能。 (PB14_15_EBI , PB14_S31, GPB_MFP14) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用INT0功能 (0, 1, 1) = 选用SPI3_SS1功能 (1, 0, 1) = 选用AD0功能
[2]	**PA7_S21**	**PA.7复用多功能引脚功能选择** EBI_EN (ALT_MFP[11]), PA7_SC1DAT (ALT_MFP1[6]), PA7_S21 (ALT_MFP[2]) 和 GPA_MFP[7] 决定了 PA.7 的功能。 (EBI_EN, PA7_SC1DAT, PA7_S21, GPA_MFP7) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC7功能 (0, 0, 1, 1) = 选用SPI2_SS1功能 (0, 1, 0, 1) = 选用SC1_DAT\UART4_RXD功能 (1, 0, 0, 1) = 选用AD6功能
[1]	**PB9_S11**	**PB.9复用多功能引脚功能选择** PB9_S11 (ALT_MFP[1]) 和 GPB_MFP[9] 决定了 PB.9 的功能。 (PB9_S11, GPB_MFP9) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用TM1功能 (1, 1) = 选用SPI1_SS1功能
[0]	**PB10_S01**	**PB.10复用多功能引脚功能选择** PB10_S01 (ALT_MFP[0]) 和 GPB_MFP[10] 决定了PB.10 的功能。 (PB10_S01, GPB_MFP10) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用TM2 功能 (1, 1) = 选用SPI0_SS1功能

复用多功能引脚控制寄存器1 (ALT_MFP1)

寄存器	偏移地址	R/W	描述	复位值
ALT_MFP1	GCR_BA+0x58	R/W	复用多功能引脚控制寄存器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	PB3_ SC2CD	PA14_ SC2RST	PA15_ SC2PWR	PA12_ SC2DAT	PA13_ SC2CLK	PC7_ SC1CD	PA5_ SC1RST
7	6	5	4	3	2	1	0
PA4_ SC1PWR	PA7_ SC1DAT	PA6_ SC1CLK	PC6_ SC0CD	PA1_ SC0RST	PA0_ SC0PWR	PA3_ SC0DAT	PA2_ SC0CLK

位	描述	
[31:15]	Reserved	保留
[14]	PB3_SC2CD	**PB.3 复用多功能引脚功能选择** EBI_nWRH_EN (ALT_MFP[14]), EBI_EN (ALT_MFP[11]), PB3_TM3 (ALT_MFP2[5]), PB3_SC2CD (ALT_MFP1[14]), PB3_T3EX (ALT_MFP[27]) 和 GPB_MFP[3] 决定了 PB.3 的功能。 (EBI_nWRH_EN, EBI_EN PB3_TM3, PB3_SC2CD, PB3_T3EX, GPB_MFP3) 数值和功能对应如下： (0, 0, 0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 0, 0, 1) = 选用UART0_nCTS功能 (0, 0, 0, 0, 1, 1) = 选用TM3_EXT功能 (0, 0, 0, 1, 0, 1) = 选用SC2_CD功能 (0, 0, 1, 0, 0, 1) = 选用TM3功能 (1, 1, 0, 0, 0, 1) = 选用nWRH功能
[13]	PA14_SC2RST	**PA.14复用多功能引脚功能选择** EBI_HB_EN[7] (ALT_MFP[23]), EBI_EN (ALT_MFP[11]), PA14_SC2RST (ALT_MFP1[13]) 和 GPA_MFP[14] 决定了 PA.14 的功能。 (EBI_HB_EN, EBI_EN, PA14_SC2RST, GPA_MFP14) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用PWM2功能 (0, 0, 1, 1) = 选用SC2_RST功能 (1, 1, 0, 1) = 选用AD15功能

[12]	**PA15_SC2PWR**	**PA.15复用多功能引脚功能选择** PA15_SC2PWR (ALT_MFP1[12]), PA15_I2SMCLK (ALT_MFP[9]) 和GPA_MFP[15] 决定了 PA.15 的功能。 (PA15_SC2PWR, PA15_I2SMCLK, GPA_MFP15) 数值和功能对应如下： (0, 0, 0) = 选用GPIOA功能 (0, 0, 1) = 选用PWM3功能 (0, 1, 1) = 选用I2S_MCLK功能 (1, 0, 1) = 选用SC2_PWR功能
[11]	**PA12_SC2DAT**	**PA.12复用多功能引脚功能选择** EBI_HB_EN[5] (ALT_MFP[21]), EBI_EN (ALT_MFP[11]), PA12_SC2DAT (ALT_MFP1[11]) 和 GPA_MFP[12] 决定了 PA.12 的功能。 (EBI_HB_EN, EBI_EN, PA12_SC2DAT, GPA_MFP12) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用PWM0功能 (0, 0, 1, 1) = 选用SC2_DAT/UART5_RXD功能 (1, 1, 0, 1) = 选用AD13功能
[10]	**PA13_SC2CLK**	**PA.13复用多功能引脚功能选择** EBI_HB_EN[6] (ALT_MFP[22]), EBI_EN (ALT_MFP[11]), PA13_SC2CLK (ALT_MFP1[10]) 和 GPA_MFP[13] 决定了 PA.13 的功能。 (EBI_HB_EN, EBI_EN, PA13_SC2CLK, GPA_MFP13) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用PWM1功能 (0, 0, 1, 1) = 选用SC2_CLK/UART5_TXD功能 (1, 1, 0, 1) = 选用AD14功能
[9]	**PC7_SC1CD**	**PC.7复用多功能引脚功能选择** EBI_EN (ALT_MFP[11]), PC7_SC1CD (ALT_MFP1[9]) 和 GPC_MFP[7] 决定了 PC.7 的功能。 (EBI_EN, PC7_SC1CD, GPC_MFP7) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用ACMP0_N功能 (0, 1, 1) = 选用SC1_CD功能 (1, 0, 1) = 选用AD5功能
[8]	**PA5_SC1RST**	**PA.5复用多功能引脚功能选择** EBI_HB_EN[1] (ALT_MFP[17]), EBI_EN (ALT_MFP[11]),PA5_SC1RST (ALT_MFP1[8]) 和 GPA_MFP[5] 决定了 PA.5 的功能。 (EBI_HB_EN , EBI_EN, PA5_SC1RST, GPA_MFP5) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用ADC5功能 (0, 1, 1) = 选用SC1_RST功能 (1, 0, 1) = 选用AD8 功能

[7]	**PA4_SC1PWR**	**PA.4复用多功能引脚功能选择** EBI_HB_EN[1] (ALT_MFP[17]), EBI_EN (ALT_MFP[11]),PA5_SC1RST (ALT_MFP1[8]) 和 GPA_MFP[5] 决定了 PA.5 的功能。 (EBI_HB_EN, EBI_EN, PA5_SC1RST, GPA_MFP5) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用ADC5功能 (0, 1, 1) = 选用SC1_RST功能 (1, 0, 1) = 选用AD9功能
[6]	**PA7_SC1DAT**	**PA.7复用多功能引脚功能选择** EBI_EN (ALT_MFP[11]), PA7_SC1DAT (ALT_MFP1[6]), PA7_S21 (ALT_MFP[2]) 和 GPA_MFP[7] 决定了 PA.7 的功能。 (EBI_EN, PA7_SC1DAT, PA7_S21, GPA_MFP7) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC7功能 (0, 0,1, 1) = 选用SPI2_SS1功能 (0, 1, 0, 1) = 选用SC1_DAT\UART4_RXD功能 (1, 0, 0, 1) = 选用AD6功能
[5]	**PA6_SC1CLK**	**PA.6复用多功能引脚功能选择** EBI_EN (ALT_MFP[11]), PA6_SC1CLK (ALT_MFP1[5]) 和 GPA_MFP[6] 决定了 PA.6 的功能。 (EBI_EN , PA6_SC1CLK, GPA_MFP6) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用ADC6功能 (0, 1, 1) = 选用SC1_CLK/UART4_TXD功能 (1, 0, 1) = 选用AD7功能
[4]	**PC6_SC0CD**	**PC.6复用多功能引脚功能选择** EBI_EN (ALT_MFP[11]), PC6_SC0CD (ALT_MFP1[4]) 和 GPC_MFP[6] 决定了 PC.6 的功能。 (EBI_EN, PC6_SC0CD, GPB_MFP6) 数值和功能对应如下： (0, 0, 0) = 选用GPIO功能 (0, 0, 1) = 选用ACMP0_P功能 (0, 1, 1) = 选用SC0_CD功能 (1, 0, 1) = 选用AD4功能
[3]	**PA1_SC0RST**	**PA.1复用多功能引脚功能选择** EBI_HB_EN[4] (ALT_MFP[20]), EBI_EN (ALT_MFP[11]), PA1_SC0RST (ALT_MFP1[3]) 和 GPA_MFP[1] 决定了 PA.0 的功能。 (EBI_HB_EN, EBI_EN, PA1_SC0RST, GPA_MFP1) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC1功能 (0, 0, 1, 1) = 选用SC0_RST功能 (1, 1, 0, 1) = 选用AD12功能

[2]	**PA0_SC0PWR**	**PA.0复用多功能引脚功能选择** PA0_SC0PWR (ALT_MFP1[2]) 和 GPA_MFP[0] 决定了 PA.0 的功能。 (PA0_SC0PWR, GPA_MFP0) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用ADC0功能 (1, 1) = 选用SC0_PWR功能
[1]	**PA3_SC0DAT**	**PA.3复用多功能引脚功能选择** EBI_HB_EN[2] (ALT_MFP[18]), EBI_EN (ALT_MFP[11]), PA3_SC0DAT (ALT_MFP1[1]) 和 GPA_MFP[3] 决定了 PA.3 的功能。 (EBI_HB_EN, EBI_EN PA3_SC0DAT, GPA_MFP3) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC3功能 (0, 0, 1, 1) = 选用SC0_DAT/UART3_RXD功能 (1, 1, 0, 1) = 选用AD10功能
[0]	**PA2_SC0CLK**	**PA.2复用多功能引脚功能选择** EBI_HB_EN[3] (ALT_MFP[19]), EBI_EN (ALT_MFP[11]), PA2_SC0CLK (ALT_MFP1[0]) 和 GPA_MFP[2] 决定了 PA.2 的功能。 (EBI_HB_EN, EBI_EN, PA2_SC0CLK, GPA_MFP2) 数值和功能对应如下： (0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 1) = 选用ADC2功能 (0, 0, 1, 1) = 选用SC0_CLK/UART3_TXD功能 (1, 1, 0, 1) = 选用AD11功能

复用多功能引脚控制寄存器2 (ALT_MFP2)

寄存器	偏移地址	R/W	描述	复位值
ALT_MFP2	GCR_BA+0x5C	R/W	复用多功能引脚控制寄存器2	0x0000_0000

31	30	29	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		PB3_TM3	PB2_TM2	PE5_TM1	PB15_TM0	PB14_15_EBI	PD14_15_CAN1

位	描述	
[31:6]	**Reserved**	保留
[5]	**PB3_TM3**	**PB.3复用多功能引脚功能选择** EBI_nWRH_EN (ALT_MFP[14]), EBI_EN (ALT_MFP[11]), PB3_TM3 (ALT_MFP2[5]), PB3_SC2CD (ALT_MFP1[14]), PB3_T3EX (ALT_MFP[27]) and GPB_MFP[3] 决定了 PB.3 的功能。 (EBI_nWRH_EN, EBI_EN PB3_TM3, PB3_SC2CD, PB3_T3EX, GPB_MFP3) 数值和功能对应如下： (0, 0, 0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 0, 0, 1) = 选用UART0_nCTS功能 (0, 0, 0, 0, 1, 1) = 选用TM3_EXT功能 (0, 0, 0, 1, 0, 1) = 选用SC2_CD功能 (0, 0, 1, 0, 0, 1) = 选用TM3功能 (1, 1, 0, 0, 0, 1) = 选用nWRH(EBI)功能
[4]	**PB2_TM2**	**PB.2复用多功能引脚功能选择** EBI_nWRL_EN (ALT_MFP[13]), EBI_EN (ALT_MFP[11]), PB2_TM2 (ALT_MFP2[4]), PB2_CPO0 (ALT_MFP[30]), PB2_T2EX (ALT_MFP[26]) 和 GPB_MFP[2] 决定了 PB.2 的功能。 (EBI_nWRL_EN, EBI_EN, PB2_TM2, PB2_CPO0, PB2_T2EX, GPB_MFP2) 数值和功能对应如下： (0, 0, 0, 0, 0, 0) = 选用GPIO功能 (0, 0, 0, 0, 0, 1) = 选用UART0_nRTS功能 (0, 0, 0, 0, 1, 1) = 选用TM2_EXT功能 (0, 0, 0, 1, 0, 1) = 选用ACMP0_O功能 (0, 0, 1, 0, 0, 1) = 选用TM2功能 (1, 1, 0, 0, 0, 1) = 选用nWRL功能

<table>
<tr><td>[3]</td><td>PE5_TM1</td><td>PE.5复用多功能引脚功能选择
PE5_T1EX (ALT_MFP[25]), PE5_TM1 (ALT_MFP2[3]) 和GPE_MFP5 决定了 PE.5 的功能。
(PE5_T1EX, GPE_MFP5) 数值和功能对应如下：
(0, 0, 0) = 选用GPIO功能
(0, 0, 1) = 选用PWM5 功能
(1, 0, 1) = 选用TM1_EXT 功能
(0, 1, 1) = 选用TM1 功能</td></tr>
<tr><td>[2]</td><td>PB15_T0EX</td><td>PB.15复用多功能引脚功能选择
PB14_15_EBI (ALT_MFP2[1]), PB15_T0EX (ALT_MFP[24]), PB15_TM0 (ALT_MFP2[2]) 和 GPB_MFP[15] 决定了 PB.15 的功能。
(PB14_15_EBI, PB15_T0EX, PB15_TM0, GPB_MFP15) 数值和功能对应如下：
(0, 0, 0, 0) = 选用GPIO功能
(0, 0, 0, 1) = 选用INT1功能
(0 ,0, 1, 1) = 选用TM0功能
(0, 1, 0, 1) = 选用TM0_EXT功能
(1, 0, 0, 1) = 选用AD6功能</td></tr>
<tr><td>[1]</td><td>PB14_15_EBI</td><td>PB14_15_EBI (ALT_MFP2[1]), PB15_T0EX (ALT_MFP[24]), PB15_TM0 (ALT_MFP2[2]) 和 GPB_MFP[15] 决定了 PB.15 的功能。
<table>
<tr><th>PB15_T0EX</th><th>PB14_15_EBI</th><th>PB15_TM0</th><th>GPB_MFP[15]</th><th>PB.15 功能</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>/INT1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>T0EX (TMR0)</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>AD6</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>TM0</td></tr>
</table>
PB14_S31 (ALT_MFP[3]), PB14_15_EBI (ALT_MFP2[1]) 和 GPB_MFP[14] 决定了 PB.14 的功能。
<table>
<tr><th>PB14_S31</th><th>PB14_15_EBI</th><th>GPB_MFP[14]</th><th>PB.14 功能</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>/INT0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>SPISS31 (SPI3)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>AD0</td></tr>
</table>
</td></tr>
</table>

[0]	**PD14_15_CAN1**	**PD.14 和PD.15 Pin复用多功能引脚功能选择** PD14_15_CAN1 (ALT_MFP2[0]) 和 GPD_MFP[15] 决定了 PD.15 的功能。 (PD14_15_CAN1, GPD_MFP15) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用UART2_TXD功能 (1, 1) = 选用CAN1_TXD功能 PD14_15_CAN1 (ALT_MFP2[0]) 和 GPD_MFP[14] 决定了 PD.14 的功能。 (PD14_15_CAN1, GPD_MFP14) 数值和功能对应如下： (0, 0) = 选用GPIO功能 (0, 1) = 选用UART2_RXD功能 (1, 1) = 选用CAN1_RXD功能

HIRC校准控制寄存器 (IRCTRIMCTL)

寄存器	偏移地址	R/W	描述	复位值
IRCTRIMCTL	GCR_BA+0x80	R/W	IRC 校准控制寄存器	0x0000_0000

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

位	描述	
[31:9]	**Reserved**	保留
[8]	**CLKERR_STOP_EN**	**时钟错误停止使能位** 0 = HIRC校准正常运行 1 = HIRC校准停止运行
[7:6]	**TRIM_RETRY_CNT**	**校准数值更新极限计数** 该寄存器定义了在HIRC频率锁住之前，自动校准电路会多少次试着更新HIRC的校准值 当HIRC锁定，内部校准数值计数器会复位。 如果HIRC还没有锁定，校准数值更新计数器已经达到了极限值，自动校准操作将停止，TRIM_SEL也会清零。 00 = 校准重试极限是 64. 01 = 校准重试极限是128. 10 = 校准重试极限是256. 11 = 校准重试极限是512.
[5:4]	**TRIM_LOOP**	**校准计算循环** 该部分定义了校准数值计算是基于多少个32.768 kHz的时钟基础上的。 例如，TRIM_LOOP设置为00，自动校准电路会在平均频率4个32.768 kHz时钟的基础上计算校准值。 00 = 校准数值基于平均4个不同的时钟 01 = 校准数值基于平均8个不同的时钟 10 = 校准数值基于平均16个不同的时钟 11 = 校准数值基于平均32个不同的时钟
[3:2]	**Reserved**	保留
[1:0]	**TRIM_SEL**	**校准频率选择** 该寄存器表示内部22.1184 MHz高速时钟将自动校准为精确的22.1184MHz或24MHz。

		如果没有选择目标频率(TRIM_SEL is 00), HIRC自动校准禁用。 在自动校准期间，如果检测到由于CLKERR_STOP_EN =1或达到校准重试限制值等时钟错误， 该寄存器会自动清零， 00 = HIRC 自动校准功能禁用 01 = HIRC 自动校准功能使能，HIRC校准到22.1184 MHz 10 = HIRC 自动校准功能使能，HIRC校准到24 MHz 11 = 保留

HIRC校准中断使能寄存器(IRCTRIMIEN)

寄存器	偏移地址	R/W	描述	复位值
IRCTRIMIEN	GCR_BA+0x84	R/W	IRC 校准中断使能寄存器	0x0000_0000

31	30	29	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					**CLKERR_IEN**	**TRIM_FAIL_IEN**	**Reserved**

位	描述	
[31:3]	**Reserved**	保留
[2]	**CLKERR_IEN**	**时钟错误中断使能位** 该位控制着在自动校准期间，当时钟不准的时候，CPU是否会产生中断。 如果这位设置为1，并且在自动校准操作期间，CLKERR_INT (IRCTRIMINT[2])使能，会产生一个中断通知时钟频率不准。 0 = 禁止CLKERR_INT (IRCTRIMINT[2]) 触发CPU中断 1 = 使能CLKERR_INT (IRCTRIMINT[2]) 触发CPU中断
[1]	**TRIM_FAIL_IEN**	**校准失败中断使能位** 该位控制着当HIRC校准值达到更新极限的时候，是否产生一个中断，及HIRC的频率是否仍然锁定在TRIM_SEL (IRCTRIMCTL[1:0])设定的目标频率。 如果该位为1，并且在自动调整操作期间，TRIM_FAIL_INT (IRCTRIMINT[1])被使能，会产生一个中断通知HIRC达到了自动校准更新限制次数。 0 = 禁止TRIM_FAIL_INT (IRCTRIMINT[1]) 触发CPU中断 1 = 使能TRIM_FAIL_INT (IRCTRIMINT[1]) 触发CPU中断
[0]	**Reserved**	保留

HIRC校准中断状态寄存器(IRCTRIMINT)

寄存器	偏移地址	R/W	描述	复位值
IRCTRIMINT	GCR_BA+0x88	R/W	IRC 校准中断状态寄存器	0x0000_0000

31	30	29	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					CLKERR_INT	TRIM_FAIL_INT	FREQ_LOCK

位	描述	
[31:3]	**Reserved**	保留
[2]	**CLKERR_INT**	**时钟错误中断状态** 当外部的32.768 kHz低速晶振或内部的22.1184 MHz高速时钟不准时，该位被置起，说明时钟频率不精确。 如果这位被置为1，自动校验操作会停止，以及如果CLKERR_STOP_EN (IRCTRIMCTL[8])也被置1，TRIM_SEL (IRCTRIMCTL[1:0]) 会硬件自动清为00， 如果这位被置为1，以及CLKERR_IEN (IRCTRIMIEN [2])为高，会产生中断通知时钟频率不准。向该位写1清零。 0 = 时钟频率准确 1 = 时钟频率不准
[1]	**TRIM_FAIL_INT**	**校准失败中断状态** 该位表明内部22.1184 MHz高速时钟达到校验值更新次数，并且内部22.1184Mhz高速时钟频率仍然没有锁定。一旦这位被置起，自动校验操作会停止，并且TRIM_SEL (IRCTRIMCTL[1:0])会被硬件自动清零。 如果该位被置起，并且TRIM_FAIL_IEN (IRCTRIMIEN[1])为高，会产生一个中断通知HIRC校准值更新限制次数已达到。向该位写1清零。 0 = 校准值没有达到更新极限数 1 = 校准值达到更新极限，并且22.1184 MHz高速时钟频率未锁定
[0]	**FREQ_LOCK**	**HIRC 频率锁定状态** 该位表明内部22.1184 MHz高速时钟频率锁定。 这是个状态位不会触发任何中断。

寄存器写保护控制寄存器 (REGWRPROT)

有些系统控制寄存器需要被保护起来，以防止误操作而影响芯片运行，这些寄存器在上电复位到用户解锁之前是锁定的。用户可以连续依次写入“59h”, “16h” “88h”到寄存器REGWRPROT(地址：0x5000_0100)解锁定。在这三个数据之间写入任何其他数据，不同时序或写入其他地址都会终止整个时序，导致无法解锁。

解锁后，用户可以检测解锁指示位：地址0x5000_0100的bit0,1表示已经解锁定，0表示锁定。用户可以更新目标寄存器的值，向地址0x5000_0100写入任何值，就可以重新锁定保护寄存器。

该位寄存器用于禁用/使能保护寄存器，读取得到REGWRPROT状态。

寄存器	偏移地址	R/W	描述	复位值
REGWRPROT	GCR_BA+0x100	R/W	寄存器写保护控制寄存器	0x0000_0000

31	30	29	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**
REGWRPROT[7:1]							**REGWRPROT [0] REGPROTDIS**

位	描述	
[31:16]	**Reserved**	保留
[7:1]	**REGWRPROT**	**寄存器写保护码 (只写)** 一些寄存器具有写保护功能. 写这些保护寄存器必须通过依次写入“59h”, “16h”, “88h” 到REGPROTDIS解除锁定状态。这个时序完成后， REGPROTDIS位将被置为1，写保护寄存器可以正常写入数据。
[0]	**REGPROTDIS**	**寄存器写保护禁用标志 (只读)** 0 = 写保护已使能。不能写入受保护寄存器。 1 = 写保护已禁用。可以写入受保护寄存器。 受保护的寄存器有: **IPRSTC1**: 地址0x5000_0008 **BODCR**: 地址0x5000_0018 **PORCR**: 地址 0x5000_0024 **PWRCON**: 地址0x5000_0200 (在电源唤醒中断被清时，bit[6]不受保护) **APBCLK bit[0]**: 地址0x5000_0208 (bit[0]是看门狗时钟使能) **CLKSEL0**: 地址0x5000_0210 (选择HCLK 和CPU STCLK 时钟源) **CLKSEL1 bit[1:0]**: 地址0x5000_0214 (用于看门狗时钟源选择) **NMI_SEL bit[8]**: 地址0x5000_0380 (用于 NMI_EN 时钟源选择)

		ISPCON: 地址0x5000_C000 (Flash ISP 控制寄存器) **ISPTRG**: 地址0x5000_C010 (ISP Trigger 控制寄存器) **WTCR**: 地址0x4000_4000 **FATCON**: 地址0x5000_C018 **注:** 这些位在寄存器描述旁注释为“写保护”。

5.2.7 系统定时器

Cortex-M0 包含系统定时器：SysTick。SysTick 提供一种简单的24位写清零、递减、自装载同时具有可灵活控制机制的计数器。该计数器可用作实时系统(RTOS) 的滴答定时器或一个简单的计数器。

当系统定时器使能后，将从 SysTick 的当前值寄存器 (SYST_CVR) 的值向下计数到0，并在下一个时钟周期，重新加载 SysTick 重新加载值寄存器 (SYST_RVR) 的值。当计数器减到0时，标志位COUNTFLAG置位，读 COUNTFLAG 位使其清零。

复位后，SYST_CVR 的值未知。使能前，软件应该向寄存器写入值清零。这样确保定时器以 SYST_RVR 的值计数，而非任意值。

若 SYST_RVR 为0，在重新加载后，定时器将保持当前值0。这个功能可以在计数器使能后用来禁用独立的功能。

详情请参考 “ARM® Cortex™-M0 Technical Reference Manual” 与 “ARM® v6-M Architecture Reference Manual”。

5.2.7.1 *系统定时器控制寄存器映射*

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
SYST 基地址: **SCS_BA = 0xE000_E000**				
SYST_CSR	SCS_BA+0x10	R/W	SysTick 控制与状态寄存器	0x0000_0000
SYST_RVR	SCS_BA+0x14	R/W	SysTick 重新加载值寄存器	0xXXXX_XXXX
SYST_CVR	SCS_BA+0x18	R/W	SysTick 当前值寄存器	0xXXXX_XXXX

5.2.7.2 *系统定时器控制寄存器描述*

SysTick 控制与状态寄存器(SYST_CSR)

寄存器	偏移地址	R/W	描述	复位值
SYST_CSR	SCS_BA+0x10	R/W	SysTick 控制与状态寄存器	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**

位	描述	
[31:17]	**Reserved**	保留
[16]	**COUNTFLAG**	**从上次读取该寄存器后，如果定时器计数到0，则返回1** 计数从1到0，COUNTFLAG 置位 在读或写当前寄存器时，COUNTFLAG被清零
[15:3]	**Reserved**	保留
[2]	**CLKSRC**	**系统Tick时钟源选择** 如果ICLKSRC(SYST_CSR[2]) = 1, SysTick 时钟源是HCLK. 如果CLKSRC(SYST_CSR[2]) = 0, SysTick 时钟源由STCLK_S(CLKSEL0[5:3])定义. 0 = 时钟源为(可选)外部参考时钟 1 = 内核时钟用于SysTick
[1]	**TICKINT**	**系统Tick中断使能** 0 = 向下计数到 0 不会引起 SysTick 异常而挂起。软件根据 COUNTFLAG 来确定是否已经发生计数到 0。 1 = 向下计数到 0 将引起 SysTick 异常而挂起。软件清 SysTick 当前值寄存器 (SYST_CVR) 将不会导致 SysTick 挂起。
[0]	**ENABLE**	**系统Tick计数使能** 0 = 禁用计数器 1 = 计数器运行于连拍方式 (multi-shot manner)

SysTick 重新加载值寄存器(SYST_RVR)

寄存器	偏移地址	R/W	描述	复位值
SYST_RVR	SCS_BA+0x14	R/W	SysTick 重新加载值寄存器	0xXXXX_XXXX

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

位	描述	
[31:24]	**Reserved**	保留
[23:0]	**RELOAD**	当计数器达到 0 时，这个值加载到当前值寄存器

SysTick 当前值寄存器(SYST_CVR)

寄存器	偏移地址	R/W	描述	复位值
SYST_CVR	SCS_BA+0x18	R/W	SysTick 当前值寄存器	0xXXXX_XXXX

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

位	描述	
[31:24]	Reserved	保留
[23:0]	CURRENT	系统Tick当前数值 当前计数值。该值为采样时刻的计数器的值，计数器不提供读修改写保护功能，该寄存器为写清除 (write-clear)。软件写入任何值将清寄存器为 0。

5.2.8 嵌套向量中断控制器(NVIC)

Cortex-M0 提供中断控制器，用于总体管理异常，称之为“嵌套向量中断控制器 (NVIC)”。NVIC和处理器内核紧密相连，它提供以下特征：

- 支持嵌套和向量中断
- 自动保存和恢复处理器状态
- 简化的和确定的中断时间

NVIC 依照优先级处理所有支持的异常，所有异常在“处理器模式”处理。NVIC 结构支持32个(IRQ[31:0])离散中断，每个中断可以支持 4 级离散中断优先级。所有的中断和大多数系统异常可以配置为不同优先级。当中断发生时，NVIC 将比较新中断与当前中断的优先级，如果新中断优先级高，则立即处理新中断。

当接受任何中断时，ISR的开始地址可从内存的向量表中取得。不需要确定哪个中断被响应，也不要软件分配相关中断服务程序（ISR）的开始地址。当开始地址取得时，NVIC 将自动保存处理状态到栈中，包括以下寄存器“PC, PSR, LR, R0~R3, R12” 的值。在ISR结束时，NVIC 将从栈中恢复相关寄存器的值，进行正常操作，因此花费少量且确定的时间处理中断请求。

NVIC 支持末尾连锁 “Tail Chaining”，有效处理背对背中断 “back-to-back interrupts”，即无需保存和恢复当前状态从而减少在切换当前ISR时的延迟时间。NVIC 还支持迟到“Late Arrival”，改善同时发生的ISR的效率。当较高优先级中断请求发生在当前ISR开始执行之前（保持处理器状态和获取起始地址阶段），NVIC 将立即处理更高优先级的中断，从而提高了实时性。

详情请参考“ARM® Cortex™-M0 Technical Reference Manual” 与 “ARM® v6-M Architecture Reference Manual”.

5.2.8.1 *异常模式和系统中断映射*

NuMicro NUC200 系列支持 表6-2 所列的异常模式。与所有中断一样，软件可以对其中一些中断设置4级优先级。最高优先级为“0”，最低优先级为“3”，所有用户可配置的优先级的默认值为“0”。 注意：优先级为“0”在整个系统中为第4优先级，排在“Reset”, “NMI” 与“Hard Fault”之后。

异常名称	向量号	优先级
Reset	1	-3
NMI	2	-2
Hard Fault	3	-1
Reserved	4 ~ 10	保留
SVCall	11	可配置
Reserved	12 ~ 13	保留
PendSV	14	可配置
SysTick	15	可配置
Interrupt (IRQ0 ~ IRQ31)	16 ~ 47	可配置

表 5-2 异常模式

向量号	中断号 (内核中的中断寄存器的对应位)	中断名称	模块	中断描述
1 ~ 15	-	-	-	系统异常
16	0	**BOD_INT**	Brown-out	欠压检测中断
17	1	**WDT_INT**	WDT	看门狗定时器中断
18	2	**EINT0**	GPIO	PB.14 管脚上的外部信号中断
19	3	**EINT1**	GPIO	PB.15 管脚上的外部信号中断
20	4	**GPAB_INT**	GPIO	PA[15:0]/PB[13:0] 的外部信号中断
21	5	**GPCDEF_INT**	GPIO	PC[15:0]/PD[15:0]/PE[15:0]/ PF[3:0] 的外部信号中断
22	6	**PWMA_INT**	PWM0~3	PWM0, PWM1, PWM2 与 PWM3 中断
23	7	**PWMB_INT**	PWM4~7	PWM4, PWM5, PWM6 与 PWM7 中断
24	8	**TMR0_INT**	TMR0	Timer 0中断
25	9	**TMR1_INT**	TMR1	Timer 1中断
26	10	**TMR2_INT**	TMR2	Timer 2中断
27	11	**TMR3_INT**	TMR3	Timer 3中断
28	12	**UART02_INT**	UART0/2	UART0 和 UART2中断
29	13	**UART1_INT**	UART1	UART1中断
30	14	**SPI0_INT**	SPI0	SPI0中断

31	15	**SPI1_INT**	SPI1	SPI1中断
32	16	**SPI2_INT**	SPI2	SPI2中断
33	17	**SPI3_INT**	SPI3	SPI3中断
34	18	**I2C0_INT**	I^2C0	I^2C0中断
35	19	**I2C1_INT**	I^2C1	I^2C1中断
36	20	**-**	-	保留
37	21	**-**	-	保留
38	22	**SC012_INT**	SC0/1/2	SC0, SC1 和 SC2中断
39	23	**USB_INT**	USBD	USB 2.0 FS 设备中断
40	24	**PS2_INT**	PS/2	PS/2 中断
41	25	**ACMP_INT**	ACMP	模拟比较器中断
42	26	**PDMA_INT**	PDMA	PDMA中断
43	27	**I2S_INT**	I^2S	I^2S i中断
44	28	**PWRWU_INT**	CLKC	从掉电状态唤醒的时钟控制器中断
45	29	**ADC_INT**	ADC	ADC中断
46	30	**IRC_INT**	IRC	IRC TRIM中断
47	31	**RTC_INT**	RTC	Real Time Clock中断

表 5-3 系统中断映射

5.2.8.2 *向量表*

响应中断时，处理器自动从内存的向量表中取出中断服务例程（ISR）的起始地址。对于 ARMv6-M，向量表的基地址为 0x00000000。向量表包括复位后堆栈的初始值以及所有异常处理器的入口地址。

上一页的向量号表示处理异常的先后次序。

向量表字偏移地址	描述
0	SP_main – 主栈指针
向量号	异常入口指针，用向量号表示

表 5-4 向量表格式

5.2.8.3 *操作说明*

通过写相应中断使能置位寄存器或清使能寄存器，可以使能 NVIC 中断或禁用 NVIC 中断，这些寄存器通过写 1 使能和写 1 清零，寄存器读取返回当前相应中断的使能状态，当中断禁用时，中断声明将使中断挂起，因此中断不被激活，如果在禁用时中断被激活，该中断就保持在激活状态，直到通过复位或异常返回来清除。清使能位可以阻止新的相应中断被激活。

NVIC 中断可以使用互补的寄存器对来挂起/取消挂起以使能/禁用这些中断，这些寄存器分别为 Set-Pending Register 与 Clear-Pending，可以写 1 使能和写 1 禁用，这些寄存器读取返回当前相应中断的状态。寄存器 Clear-Pending 在中断响应时的不影响执行状态。

NVIC 中断依次更新32位寄存器中的各个8位字段（每个寄存器支持4个中断）。

与 NVIC 相关的的通用寄存器都可以在内存系统控制空间寄存器（SCS_BA）其中的一块寄存器区域中设置，下一节将作出描述。

5.2.8.4 *NVIC 控制寄存器映射*

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
NVIC 基地址: **SCS_BA = 0xE000_E000**				
NVIC_ISER	SCS_BA+0x100	R/W	IRQ0 ~ IRQ31设置使能控制寄存器	0x0000_0000
NVIC_ICER	SCS_BA+0x180	R/W	IRQ0 ~ IRQ31清使能控制寄存器	0x0000_0000
NVIC_ISPR	SCS_BA+0x200	R/W	IRQ0 ~ IRQ31设置挂起控制寄存器	0x0000_0000
NVIC_ICPR	SCS_BA+0x280	R/W	IRQ0 ~ IRQ31清挂起控制寄存器	0x0000_0000
NVIC_IPR0	SCS_BA+0x400	R/W	IRQ0 ~ IRQ3优先级控制寄存器	0x0000_0000
NVIC_IPR1	SCS_BA+0x404	R/W	IRQ4 ~ IRQ7优先级控制寄存器	0x0000_0000
NVIC_IPR2	SCS_BA+0x408	R/W	IRQ8 ~ IRQ11优先级控制寄存器	0x0000_0000
NVIC_IPR3	SCS_BA+0x40C	R/W	IRQ12 ~ IRQ15优先级控制寄存器	0x0000_0000
NVIC_IPR4	SCS_BA+0x410	R/W	IRQ16 ~ IRQ19优先级控制寄存器	0x0000_0000
NVIC_IPR5	SCS_BA+0x414	R/W	IRQ20 ~ IRQ23优先级控制寄存器	0x0000_0000
NVIC_IPR6	SCS_BA+0x418	R/W	IRQ24 ~ IRQ27优先级控制寄存器	0x0000_0000
NVIC_IPR7	SCS_BA+0x41C	R/W	IRQ28 ~ IRQ31优先级控制寄存器	0x0000_0000

5.2.8.5 *NVIC控制寄存器描述*

IRQ0 ~ IRQ31设置使能控制寄存器(NVIC_ISER)

寄存器	偏移地址	R/W	描述	复位值
NVIC_ISER	SCS_BA+0x100	R/W	IRQ0 ~ IRQ31设置使能控制寄存器	0x0000_0000

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

位	描述	
[31:0]	SETENA	**中断使能寄存器** 使能一个或者多个中断，每位代表 IRQ0 ~ IRQ31 的中断号（向量号：从16到47） 写操作: 0 = 无效 1 = 写 1 使能相关中断 读操作: 0 = 相关中断状态被禁止 1 = 相关中断状态已使能 读取该寄存器返回当前使能状态.

IRQ0 ~ IRQ31清使能控制寄存器(NVIC_ICER)

寄存器	偏移地址	R/W	描述	复位值
NVIC_ICER	SCS_BA+0x180	R/W	IRQ0 ~ IRQ31清使能控制寄存器	0x0000_0000

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

位	描述	
[31:0]	CLRENA	中断禁用位 禁用一个或者多个中断，每位代表 IRQ0 ~ IRQ31 的中断号（向量号：从16到 47） 写操作: 0 = 无效 1 = 写1禁用相关中断 读操作: 0 = 相关中断状态被禁止 1 = 相关中断状态已使能 读取该寄存器返回当前使能状态

IRQ0 ~ IRQ31设置挂起控制寄存器(NVIC_ISPR)

寄存器	偏移地址	R/W	描述	复位值
NVIC_ISPR	SCS_BA+0x200	R/W	IRQ0 ~ IRQ31设置挂起控制寄存器	0x0000_0000

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

位	描述	
[31:0]	**SETPEND**	**设置中断挂起寄存器** 写操作: 0 = 无效 写 1，挂起相应中断。每位代表 IRQ0 ~ IRQ31 的中断号（向量号：从16到47）。 读操作: 0 =相关中断不在挂起状态 1 =相关中断在挂起状态 读取该寄存器返回当前等待处理的中断状态

IRQ0 ~ IRQ31清挂起控制寄存器(NVIC_ICPR)

寄存器	偏移地址	R/W	描述	复位值
NVIC_ICPR	SCS_BA+0x280	R/W	IRQ0 ~ IRQ31 清挂起控制寄存器	0x0000_0000

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

位	描述	
[31:0]	CLRPEND	**清中断挂起寄存器** 写操作: 0 = 无效 1=写1清除挂起状态。每位代表 IRQ0 ~ IRQ31 的中断号（向量号：从16到47） 读操作: 0 = 相关寄存器不在挂起状态 1 = 相关寄存器在挂起状态 读取该寄存器返回当前等待处理的中断状态

IRQ0 ~ IRQ3 优先级寄存器 (NVIC_IPR0)

寄存器	偏移地址	R/W	描述	复位值
NVIC_IPR0	SCS_BA+0x400	R/W	IRQ0 ~ IRQ3 中断优先级控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_3[1:0]		Reserved					
23	22	21	20	19	18	17	16
PRI_2[1:0]		Reserved					
15	14	13	12	11	10	9	8
PRI_1[1:0]		Reserved					
7	6	5	4	3	2	1	0
PRI_0[1:0]		Reserved					

位	描述	
[31:30]	**PRI_3**	**IRQ3优先级** “0” 表示最高优先级,“3” 表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_2**	**IRQ2优先级** “0” 表示最高优先级,“3” 表示最低优先级
[21:16]	**Reserved**	保留
[15:14]	**PRI_1**	**IRQ1优先级** “0” 表示最高优先级,“3” 表示最低优先级
[13:8]	**Reserved**	保留
[7:6]	**PRI_0**	**IRQ0优先级** “0” 表示最高优先级,“3” 表示最低优先级
[5:0]	**Reserved**	保留

IRQ4 ~ IRQ7 优先级寄存器 (NVIC_IPR1)

寄存器	偏移地址	R/W	描述	复位值
NVIC_IPR1	SCS_BA+0x404	R/W	IRQ4 ~ IRQ7 优先级控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_7[1:0]		Reserved					
23	22	21	20	19	18	17	16
PRI_6[1:0]		Reserved					
15	14	13	12	11	10	9	8
PRI_5[1:0]		Reserved					
7	6	5	4	3	2	1	0
PRI_4[1:0]		Reserved					

位	描述	
[31:30]	**PRI_7**	**IRQ7优先级** “0” 表示最高优先级,“3” 表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_6**	**IRQ6优先级** “0” 表示最高优先级,“3” 表示最低优先级
[21:16]	**Reserved**	保留
[15:14]	**PRI_5**	**IRQ5优先级** “0” 表示最高优先级,“3” 表示最低优先级
[13:8]	**Reserved**	保留
[7:6]	**PRI_4**	**IRQ4优先级** “0” 表示最高优先级,“3” 表示最低优先级
[5:0]	**Reserved**	保留

IRQ8 ~ IRQ11 优先级寄存器 (NVIC_IPR2)

寄存器	偏移地址	R/W	描述	复位值
NVIC_IPR2	SCS_BA+0x408	R/W	IRQ8 ~ IRQ11 优先级控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_11[1:0]		Reserved					
23	22	21	20	19	18	17	16
PRI_10[1:0]		Reserved					
15	14	13	12	11	10	9	8
PRI_9[1:0]		Reserved					
7	6	5	4	3	2	1	0
PRI_8[1:0]		Reserved					

位	描述	
[31:30]	**PRI_11[1:0]**	**IRQ11优先级** “0” 表示最高优先级,“3” 表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_10[1:0]**	**IRQ10优先级** “0” 表示最高优先级,“3” 表示最低优先级
[21:16]	**Reserved**	保留
[15:14]	**PRI_9[1:0]**	**IRQ9优先级** “0” 表示最高优先级,“3” 表示最低优先级
[13:8]	**Reserved**	保留
[7:6]	**PRI_8[1:0]**	**IRQ8优先级** “0” 表示最高优先级,“3” 表示最低优先级
[5:0]	**Reserved**	保留

IRQ12 ~ IRQ15优先级寄存器(NVIC_IPR3)

寄存器	偏移地址	R/W	描述	复位值
NVIC_IPR3	SCS_BA+0x40C	R/W	IRQ12 ~ IRQ15优先级控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_15[1:0]		Reserved					
23	22	21	20	19	18	17	16
PRI_14[1:0]		Reserved					
15	14	13	12	11	10	9	8
PRI_13[1:0]		Reserved					
7	6	5	4	3	2	1	0
PRI_12[1:0]		Reserved					

位	描述	
[31:30]	**PRI_15**	**IRQ15优先级** “0”表示最高优先级,“3”表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_14**	**IRQ14优先级** “0”表示最高优先级,“3”表示最低优先级
[21:16]	**Reserved**	保留
[15:14]	**PRI_13**	**IRQ13优先级** “0”表示最高优先级,“3”表示最低优先级
[13:8]	**Reserved**	保留
[7:6]	**PRI_12**	**IRQ12优先级** “0”表示最高优先级,“3”表示最低优先级
[5:0]	**Reserved**	保留

IRQ16 ~ IRQ19 优先级寄存器(NVIC_IPR4)

寄存器	偏移地址	R/W	描述	复位值
NVIC_IPR4	SCS_BA+0x410	R/W	IRQ16 ~ IRQ19优先级控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_19[1:0]		Reserved					
23	22	21	20	19	18	17	16
PRI_18[1:0]		Reserved					
15	14	13	12	11	10	9	8
PRI_17[1:0]		Reserved					
7	6	5	4	3	2	1	0
PRI_16[1:0]		Reserved					

位	描述	
[31:30]	**PRI_19**	**IRQ19优先级** “0”表示最高优先级,“3”表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_18**	**IRQ18优先级** “0”表示最高优先级,“3”表示最低优先级
[21:16]	**Reserved**	保留
[15:14]	**PRI_17**	**IRQ17优先级** “0”表示最高优先级,“3”表示最低优先级
[13:8]	**Reserved**	保留
[7:6]	**PRI_16**	**IRQ16优先级** “0”表示最高优先级,“3”表示最低优先级
[5:0]	**Reserved**	保留

IRQ20 ~ IRQ23优先级寄存器(NVIC_IPR5)

寄存器	偏移地址	R/W	描述	复位值
NVIC_IPR5	SCS_BA+0x414	R/W	IRQ20 ~ IRQ23优先级控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_23[1:0]		Reserved					
23	22	21	20	19	18	17	16
PRI_22[1:0]		Reserved					
15	14	13	12	11	10	9	8
PRI_21[1:0]		Reserved					
7	6	5	4	3	2	1	0
PRI_20[1:0]		Reserved					

位	描述	
[31:30]	**PRI_23**	**IRQ23优先级** “0”表示最高优先级,“3”表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_22**	**IRQ22优先级** “0”表示最高优先级,“3”表示最低优先级
[21:16]	**Reserved**	保留
[15:14]	**PRI_21**	**IRQ21优先级** “0”表示最高优先级,“3”表示最低优先级
[13:8]	**Reserved**	保留
[7:6]	**PRI_20**	**IRQ20优先级** “0”表示最高优先级,“3”表示最低优先级
[5:0]	**Reserved**	保留

IRQ24 ~ IRQ27优先级寄存器(NVIC_IPR6)

寄存器	偏移地址	R/W	描述	复位值
NVIC_IPR6	SCS_BA+0x418	R/W	IRQ24 ~ IRQ27优先级控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_27[1:0]		Reserved					
23	22	21	20	19	18	17	16
PRI_26[1:0]		Reserved					
15	14	13	12	11	10	9	8
PRI_25[1:0]		Reserved					
7	6	5	4	3	2	1	0
PRI_24[1:0]		Reserved					

位	描述	
[31:30]	**PRI_27**	**IRQ27优先级** “0”表示最高优先级,“3”表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_26**	**IRQ26优先级** “0”表示最高优先级,“3”表示最低优先级
[21:16]	**Reserved**	保留
[15:14]	**PRI_25**	**IRQ25优先级** “0”表示最高优先级,“3”表示最低优先级
[13:8]	**Reserved**	保留
[7:6]	**PRI_24**	**IRQ24优先级** “0”表示最高优先级,“3”表示最低优先级
[5:0]	**Reserved**	保留

IRQ28 ~ IRQ31优先级寄存器(NVIC_IPR7)

寄存器	偏移地址	R/W	描述	复位值
NVIC_IPR7	SCS_BA+0x41C	R/W	IRQ28 ~ IRQ31优先级控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PRI_31[1:0]		Reserved					
23	22	21	20	19	18	17	16
PRI_30[1:0]		Reserved					
15	14	13	12	11	10	9	8
PRI_29[1:0]		Reserved					
7	6	5	4	3	2	1	0
PRI_28[1:0]		Reserved					

位	描述	
[31:30]	**PRI_31**	**IRQ31优先级** “0”表示最高优先级,“3”表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_30**	**IRQ30优先级** “0”表示最高优先级,“3”表示最低优先级
[21:16]	**Reserved**	保留
[15:14]	**PRI_29**	**IRQ29优先级** “0”表示最高优先级,“3”表示最低优先级
[13:8]	**Reserved**	保留
[7:6]	**PRI_28**	**IRQ28优先级** “0”表示最高优先级,“3”表示最低优先级
[5:0]	**Reserved**	保留

5.2.8.6 *中断源寄存器*

除了 NVIC 相关的中断控制寄存器外，NuMicro™ NUC200 系列还有一些特殊寄存器便于中断处理，包括“中断源识别”，“NMI 源选择”与 “中断测试模式”。描述如下：

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
INT 基地址: **INT_BA = 0x5000_0300**				
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) 中断源识别	0xXXXX_XXXX
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别	0xXXXX_XXXX
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 中断源识别	0xXXXX_XXXX
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别	0xXXXX_XXXX
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (GPA/B) 中断源识别	0xXXXX_XXXX
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GPC/D/E/F) 中断源识别	0xXXXX_XXXX
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) 中断源识别	0xXXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (PWMB) 中断源识别	0xXXXX_XXXX
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别	0xXXXX_XXXX
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别	0xXXXX_XXXX
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别	0xXXXX_XXXX
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别	0xXXXX_XXXX
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (UART0/2) 中断源识别	0xXXXX_XXXX
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (UART1) 中断源识别	0xXXXX_XXXX
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别	0xXXXX_XXXX
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) 中断源识别	0xXXXX_XXXX
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (SPI2) 中断源识别	0xXXXX_XXXX
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (SPI3) 中断源识别	0xXXXX_XXXX
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I^2C0) 中断源识别	0xXXXX_XXXX
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I^2C1) 中断源识别	0xXXXX_XXXX
IRQ20_SRC	INT_BA+0x50	R	保留	0xXXXX_XXXX
IRQ21_SRC	INT_BA+0x54	R	保留	0xXXXX_XXXX
IRQ22_SRC	INT_BA+0x58	R	IRQ22 (SC0/1/2) 中断源识别	0xXXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (USBD) 中断源识别	0xXXXX_XXXX

IRQ24_SRC	INT_BA+0x60	R	IRQ24 (PS/2) 中断源识别	0xXXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (ACMP) 中断源识别	0xXXXX_XXXX
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (PDMA) 中断源识别	0xXXXX_XXXX
IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (I^2S) 中断源识别	0xXXXX_XXXX
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源识别	0xXXXX_XXXX
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (ADC) 中断源识别	0xXXXX_XXXX
IRQ30_SRC	INT_BA+0x78	R	IRQ30 (IRC) 中断源识别	0xXXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	R	IRQ31 (RTC) 中断源识别	0xXXXX_XXXX
NMI_SEL	INT_BA+0x80	R/W	NMI 中断源选择控制寄存器	0x0000_0000
MCU_IRQ	INT_BA+0x84	R/W	MCU中断请求源寄存器	0x0000_0000
MCU_IRQCR	INT_BA+0x88	R/W	MCU中断请求控制寄存器	0x0000_0000

5.2.8.7 *中断源寄存器描述*

中断源识别寄存器(IRQn_SRC)

寄存器	偏移地址	R/W	描述	复位值
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) 中断源识别	0xXXXX_XXXX
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别	0xXXXX_XXXX
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 中断源识别	0xXXXX_XXXX
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别	0xXXXX_XXXX
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (GPA/B) 中断源识别	0xXXXX_XXXX
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GPC/D/E/F) 中断源识别	0xXXXX_XXXX
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) 中断源识别	0xXXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (PWMB) 中断源识别	0xXXXX_XXXX
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别	0xXXXX_XXXX
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别	0xXXXX_XXXX
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别	0xXXXX_XXXX
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别	0xXXXX_XXXX
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (UART0/2) 中断源识别	0xXXXX_XXXX
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (UART1) 中断源识别	0xXXXX_XXXX
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别	0xXXXX_XXXX
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) 中断源识别	0xXXXX_XXXX
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (SPI2) 中断源识别	0xXXXX_XXXX
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (SPI3) 中断源识别	0xXXXX_XXXX
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I^2C0) 中断源识别	0xXXXX_XXXX
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I^2C1) 中断源识别	0xXXXX_XXXX
IRQ20_SRC	INT_BA+0x50	R	保留	0xXXXX_XXXX
IRQ21_SRC	INT_BA+0x54	R	保留	0xXXXX_XXXX
IRQ22_SRC	INT_BA+0x58	R	IRQ22 (SC0/1/2) 中断源识别	0xXXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (USBD) 中断源识别	0xXXXX_XXXX
IRQ24_SRC	INT_BA+0x60	R	IRQ24 (PS/2) 中断源识别	0xXXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (ACMP) 中断源识别	0xXXXX_XXXX
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (PDMA) 中断源识别	0xXXXX_XXXX

IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (I^2S) 中断源识别	0xXXXX_XXXX
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源识别	0xXXXX_XXXX
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (ADC) 中断源识别	0xXXXX_XXXX
IRQ30_SRC	INT_BA+0x78	R	IRQ30 (IRC) 中断源识别	0xXXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	R	IRQ31 (RTC) 中断源识别	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**
Reserved				**INT_SRC[3:0]]**			

位	描述	
[31:4]	**Reserved**	保留
[3:0]	**INT_SRC**	**中断源** 定义中断事件的中断源

位	地址	INT-Num	描述
[2:0]	INT_BA+0x00	0	Bit2: 0 Bit1: 0 Bit0: BOD_INT
[2:0]	INT_BA+0x04	1	Bit2: 0 Bit1: WWDT_INT Bit0: WDT_INT
[2:0]	INT_BA+0x08	2	Bit2: 0 Bit1: 0 Bit0: EINT0 – PB.14 上的外部中断 0
[2:0]	INT_BA+0x0C	3	Bit2: 0 Bit1: 0 Bit0: EINT1 – PB.15 上的外部中断 1
[2:0]	INT_BA+0x10	4	Bit2: 0 Bit1: GPB_INT Bit0: GPA_INT
[3:0]	INT_BA+0x14	5	Bit3: GPF_INT

			Bit2: GPE_INT Bit1: GPD_INT Bit0: GPC_INT
[3:0]	INT_BA+0x18	6	Bit3: PWM3_INT Bit2: PWM2_INT Bit1: PWM1_INT Bit0: PWM0_INT
[3:0]	INT_BA+0x1C	7	Bit3: PWM7_INT Bit2: PWM6_INT Bit1: PWM5_INT Bit0: PWM4_INT
[2:0]	INT_BA+0x20	8	Bit2: 0 Bit1: 0 Bit0: TMR0_INT
[2:0]	INT_BA+0x24	9	Bit2: 0 Bit1: 0 Bit0: TMR1_INT
[2:0]	INT_BA+0x28	10	Bit2: 0 Bit1: 0 Bit0: TMR2_INT
[2:0]	INT_BA+0x2C	11	Bit2: 0 Bit1: 0 Bit0: TMR3_INT
[2:0]	INT_BA+0x30	12	Bit2: 0 Bit1: UART2_INT Bit0: UART0_INT
[2:0]	INT_BA+0x34	13	Bit2: 0 Bit1: 0 Bit0: UART1_INT
[2:0]	INT_BA+0x38	14	Bit2: 0 Bit1: 0 Bit0: SPI0_INT
[2:0]	INT_BA+0x3C	15	Bit2: 0 Bit1: 0 Bit0: SPI1_INT
[2:0]	INT_BA+0x40	16	Bit2: 0 Bit1: 0 Bit0: SPI2_INT
[2:0]	INT_BA+0x44	17	Bit2: 0 Bit1: 0 Bit0: SPI3_INT
[2:0]	INT_BA+0x48	18	Bit2: 0 Bit1: 0 Bit0: I2C0_INT

[2:0]	INT_BA+0x4C	19	Bit2: 0 Bit1: 0 Bit0: I2C1_INT
[2:0]	INT_BA+0x58	22	Bit2: SC2_INT Bit1: SC1_INT Bit0: SC0_INT
[2:0]	INT_BA+0x5C	23	Bit2: 0 Bit1: 0 Bit0: USB_INT
[2:0]	INT_BA+0x60	24	Bit2: 0 Bit1: 0 Bit0: PS2_INT
[2:0]	INT_BA+0x64	25	Bit2: 0 Bit1: 0 Bit0: ACMP_INT
[2:0]	INT_BA+0x68	26	Bit2: 0 Bit1: 0 Bit0: PDMA_INT
[2:0]	INT_BA+0x6C	27	Bit2: 0 Bit1: 0 Bit0: I2S_INT
[2:0]	INT_BA+0x70	28	Bit2: 0 Bit1: 0 Bit0: PWRWU_INT
[2:0]	INT_BA+0x74	29	Bit2: 0 Bit1: 0 Bit0: ADC_INT
[2:0]	INT_BA+0x78	30	Bit2: 0 Bit1: 0 Bit0: IRC_INT
[2:0]	INT_BA+0x7C	31	Bit2: 0 Bit1: 0 Bit0: RTC_INT

NMI中断源选择控制寄存器(NMI_SEL)

寄存器	偏移地址	R/W	描述	复位值
NMI_SEL	INT_BA+0x80	R/W	NMI中断源选择控制寄存器	0x0000_0000

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

位	描述	
[31:8]	Reserved	保留
[8]	NMI_EN	**NMI 中断使能位（写保护位）** 0 = 禁用 NMI 中断 1 = 使能 NMI 中断 **注意:** 该位为写保护位，写该位需要写“59h”, “16h”, “88h” 到地址 0x5000_0100禁止寄存器写保护功能，参考寄存器REGWRPROT， 地址为 GCR_BA+0x100。
[7:5]	Reserved	保留
[4:0]	NMI_SEL	**NMI 中断源选择** 通过设置 NMI_SEL 可以在外围设备中断中选择 Cortex-M0 的 NMI 中断。

MCU 中断请求源寄存器(MCU_IRQ)

寄存器	偏移地址	R/W	描述	复位值
MCU_IRQ	INT_BA+0x84	R/W	MCU 中断请求源寄存器	0x0000_0000

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

位	描述	
[31:0]	**MCU_IRQ**	**MCU IRQ源寄存器** MCU_IRQ 从外围设备收集所有中断，并向 Cortex-M0 内核产生同步中断，产生此中断的模式有两种，分别是正常模式和测试模式。 MCU_IRQ 从每个外围设备收集所有中断和同步这些中断，然后触发 Cortex-M0 中断。 MCU_IRQ[n] 为 0 时：置 MCU_IRQ[n] 为 1，Cortex_M0 NVIC[n] 将产生一个中断。 MCU_IRQ[n] 为 1 时：（意味着有中断请求），这时置 MCU_IRQ[n] 为 1，将清除中断；置 MCU_IRQ[n] 为 0 则无效。

MCU中断请求控制寄存器 (MCU_IRQCR)

寄存器	偏移地址	R/W	描述	复位值
MCU_IRQCR	INT_BA+0x88	R/W	MCU 中断请求控制寄存器	0x0000_0000

31	30	29	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							**FAST_IRQ**

位	描述	
[31:1]	**Reserved**	保留
[0]	**FAST_IRQ**	**快速IRQ等待使能位** 0 = MCU IRQ等待时间在13个HCLK时钟周期完成，中断发生后，经过这段固定的等待时间MCU将进入IRQ 中断. 1 = MCU IRQ等待时间没有完成，中断发生后，MCU进入IRQ中断。

5.2.9 系统控制

系统控制寄存器控制了 Cortex-M0 的状态和操作模式，包括 CPUID，Cortex-M0 的中断优先级和 Cortex-M0 的电源管理。

详情请参考 “ARM® Cortex™-M0 Technical Reference Manual” 与 “ARM® v6-M Architecture Reference Manual”。

5.2.9.1 *系统控制寄存器映射*

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
SCS 基地址: **SCS_BA = 0xE000_E000**				
CPUID	SCS_BA+0xD00	R	CPUID寄存器	0x410C_C200
ICSR	SCS_BA+0xD04	R/W	中断控制和状态寄存器	0x0000_0000
AIRCR	SCS_BA+0xD0C	R/W	应用中断和复位控制寄存器	0xFA05_0000
SCR	SCS_BA+0xD10	R/W	系统控制寄存器	0x0000_0000
SHPR2	SCS_BA+0xD1C	R/W	系统处理优先级寄存器 2	0x0000_0000
SHPR3	SCS_BA+0xD20	R/W	系统处理优先级寄存器 3	0x0000_0000

5.2.9.2 *系统控制寄存器描述*

CPUID 寄存器 (CPUID)

寄存器	偏移地址	R/W	描述	复位值
CPUID	SCS_BA+0xD00	R	CPUID 寄存器	0x410C_C200

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

位	描述	
[31:24]	**IMPLEMENTER**	**ARM分配的执行码** ARM分配的执行码 (ARM = 0x41)
[23:20]	**Reserved**	保留
[19:16]	**PART**	**处理器架构** ARMv6-M 读取值 0xC
[15:4]	**PARTNO**	**处理器产品编号** 读取值0xC20
[3:0]	**REVISION**	**修订版本号** 读取值0x0

中断控制状态寄存器(ICSR)

寄存器	偏移地址	R/W	描述	复位值
ICSR	SCS_BA+0xD04	R/W	中断控制和状态寄存器	0x0000_0000

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

位	描述	
[31]	**NMIPENDSET**	**NMI 设置挂起位** 写操作: 0 = 该位写 0 无效 1 = 更改 NMI 异常状态为挂起 读操作: 0 = NMI 异常没有挂起 1 = NMI 异常挂起 因为 NMI 异常是最高优先级的异常，正常情况下处理器一旦检测到这一位被置1，就会立刻进入 NMI 异常处理程序。进入处理程序后处理器会将该位清为零。这意味着只有当处理器执行 NMI 异常处理程序时再次产生 NMI 信号，NMI 异常处理程序读取这一位的值将返回1。
[30:29]	**Reserved**	保留
[28]	**PENDSVSET**	**PendSV 设置挂起位** 写操作: 0 = 该位写 0 无效 1 = 更改 PendSV 异常状态为挂起 读操作: 0 = PendSV 异常没有挂起 1 = PendSV 异常挂起 **注意:** 向该位写1是将 PendSV 异常状态设为挂起的唯一方法。
[27]	**PENDSVCLR**	**PendSV 清除挂起位** 写操作: 0 = 该位写 0 无效 1 = 移除 PendSV 异常的挂起状态 只写位。当想清除 PENDSV 位时，必须同时“向 PENDSVSET 写 0 和向 PENDSVCLR 写 1。

[26]	**PENDSTSET**	**SysTick异常设置挂起位** 写操作： 0 = 该位写 0 无效 1 = 更改 SysTick 异常状态为挂起 读操作： 0 = SysTick 异常没有挂起 1 = SysTick 异常挂起
[25]	**PENDSTCLR**	**SysTick异常清除挂起位** 写操作： 0 = 该位写 0 无效 1 = 移除SysTick 异常的挂起状态 只写位。当想清除 PENDST 位时，必须同时“向 PENDSTSET 写0 和 向 PENDSTCLR 写1”。
[24]	**Reserved**	保留
[23]	**ISRPREEMPT**	若置位，则挂起的异常将从调试停止状态退出，进入活动状态。 只读位。
[22]	**ISRPENDING**	**中断挂起标志，不包括 NMI 和 Faults：** 0 = 中断没有挂起 1 = 中断挂起. 只读位。
[21:18]	**Reserved**	保留
[17:12]	**VECTPENDING**	**表示挂起异常中最高优先级异常的异常号：** 0 = 没有异常挂起 非 0 = 最高优先级挂起异常的异常号
[11:6]	**Reserved**	保留
[5:0]	**VECTACTIVE**	**当前执行异常处理的异常号** 0 = Thread 模式 非 0 = 当前执行异常处理的异常号

应用程序中断和复位控制寄存器(AIRCR)

寄存器	偏移地址	R/W	描述	复位值
AIRCR	SCS_BA+0xD0C	R/W	应用程序中断和复位控制寄存器	0xFA05_0000

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

位	描述	
[31:16]	**VECTORKEY**	**寄存器的访问键** 写操作: 写该寄存器时，写入值必须为 0x05FA，否则写动作将被忽略。VECTORKEY用于保护系统复位或异常清除误写。 读操作: 读取值 0xFA05.
[15:3]	**Reserved**	保留
[2]	**SYSRESETREQ**	**系统复位请求** 向该位写 1，产生复位信号给芯片表示有复位请求。 该位为只写位，在复位时自动清零。
[1]	**VECTCLRACTIVE**	**产生异常状态清除位** 保留为调试模式使用。写该寄存器时，用户必须向该位写0，否则将产生不可预测 的结果。
[0]	**Reserved**	保留

系统控制寄存器(SCR)

寄存器	偏移地址	R/W	描述	复位值
SCR	SCS_BA+0xD10	R/W	系统控制寄存器	0x0000_0000

31	30	29	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**

位	描述	
[31:5]	**Reserved**	保留
[4]	**SEVONPEND**	**挂起状态时的发送事件位** 0 = 只有使能中断或者事件可以唤醒处理器，不包括禁用中断在内。 1 = 使能事件和所有中断，包括禁用中断，都可以唤醒处理器。 当一个事件或中断进入挂起状态时，事件信号从 WFE 唤醒处理器。如果处理器不是在等待该事件，那么这个事件会被注册到并影响下一个 WFE。 处理器总是会在执行一个 SEV 指令或者一个外部事件时被唤醒。
[3]	**Reserved**	保留
[2]	**SLEEPDEEP**	**系统深度休眠或休眠模式选择** 控制处理器使用休眠或者深度休眠作为它的低功耗模式： 0 = 休眠 1 = 深度休眠
[1]	**SLEEPONEXIT**	**Sleep-On-Exit 使能位** 表示当从 Handler 模式切换到 Thread 模式时，是否使用 sleep-on-exit： 0 = 当切换到 Thread 模式时不休眠。 1 = 当从某个 ISR 切换到 Thread 模式时，进入休眠或者深度休眠。 设置该位为1 使能一个中断驱动应用，避免返回到一个空的主函数应用。
[0]	**Reserved**	保留

系统处理优先级寄存器2 (SHPR2)

寄存器	偏移地址	R/W	描述	复位值
SHPR2	SCS_BA+0xD1C	R/W	系统处理优先级寄存器2	0x0000_0000

31	30	29	28	27	26	25	24
PRI_11[1:0]		**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							

位	描述	
[31:30]	**PRI_11**	**系统处理的优先级11 – SVCall** “0” 表示最高优先级,“3” 表示最低优先级
[29:0]	**Reserved**	保留

系统处理优先级寄存器3 (SHPR3)

寄存器	偏移地址	R/W	描述	复位值
SHPR3	SCS_BA+0xD20	R/W	系统处理优先级寄存器3	0x0000_0000

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

位	描述	
[31:30]	**PRI_15**	**系统处理的优先级 15 – SysTick** “0”表示最高优先级，“3”表示最低优先级
[29:24]	**Reserved**	保留
[23:22]	**PRI_14**	**系统处理的优先级 14 – PendSV** “0”表示最高优先级，“3”表示最低优先级
[21:0]	**Reserved**	保留

5.3 时钟控制器

5.3.1 概述

时钟控制器为整个芯片提供时钟源，包括系统时钟和所有外围设备时钟。该控制器还通过单独时钟的开或关，时钟源选择和分频器来进行功耗控制。PWR_DOWN_EN (PWRCON[7]) 位 和 PD_WAIT_CPU (PWRCON[8])位同时设置为1, 同时CPU Cortex™-M0内核执行WFI指令，芯片将进入掉电模式。直到唤醒中断发生，芯片才会退出掉电模式。在掉电模式下，时钟控制器关闭外部4~24MHz晶振和内部22.1184MHz高速RC振荡器，以降低整个系统功耗。

时钟发生器由如下5个时钟源组成：

- 外部32.768 kHz低速晶振（LXT）
- 外部4~24 MHz高速晶振(HXT)
- 可编程的PLL输出时钟频率(PLL 由外部 4~24 MHz 晶振或内部 22.1184 MHz 振荡器提供时钟源)
- 内部22.1184 MHz高速振荡器(HIRC)
- 内部10 kHz低速RC振荡器(LIRC)

下图所示，系统和个各模块的时钟源

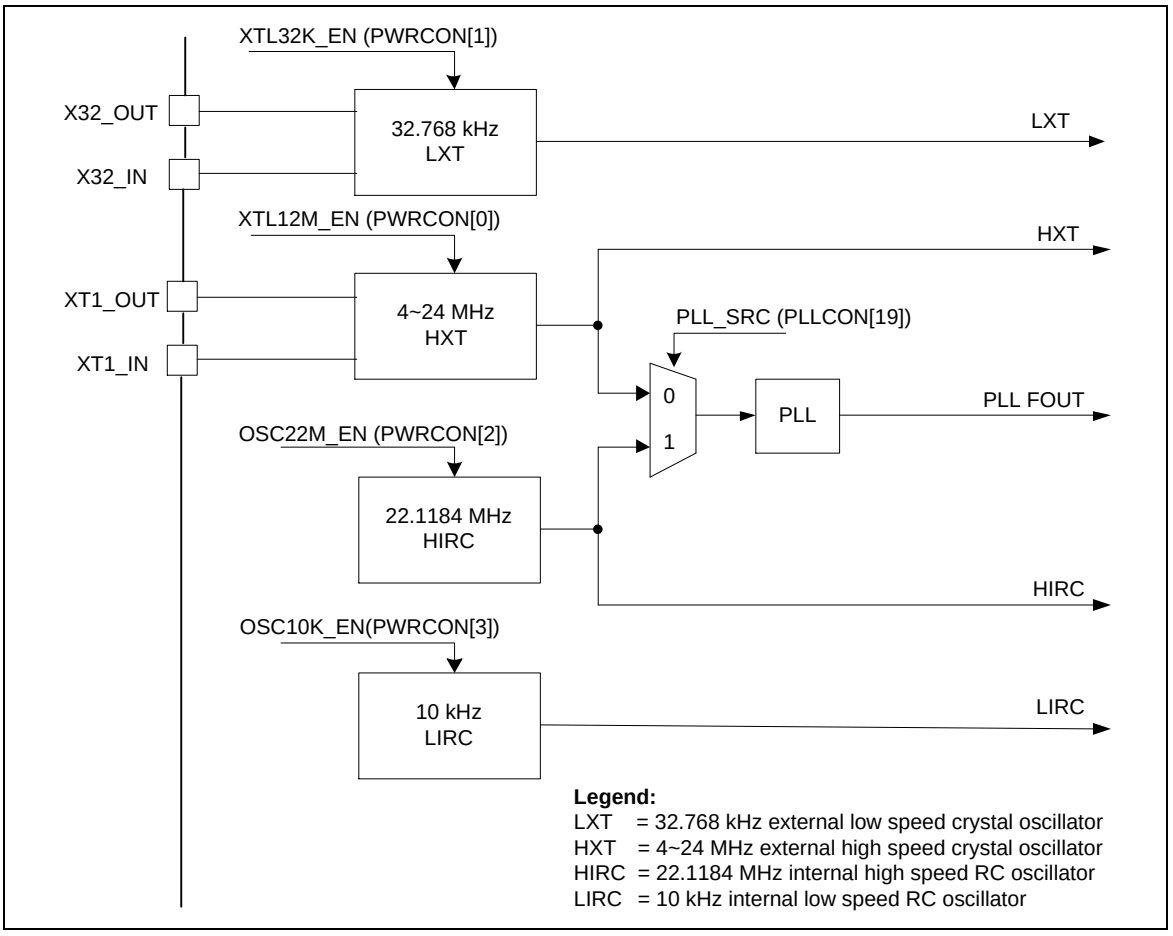


图 5-4 时钟发生器框图

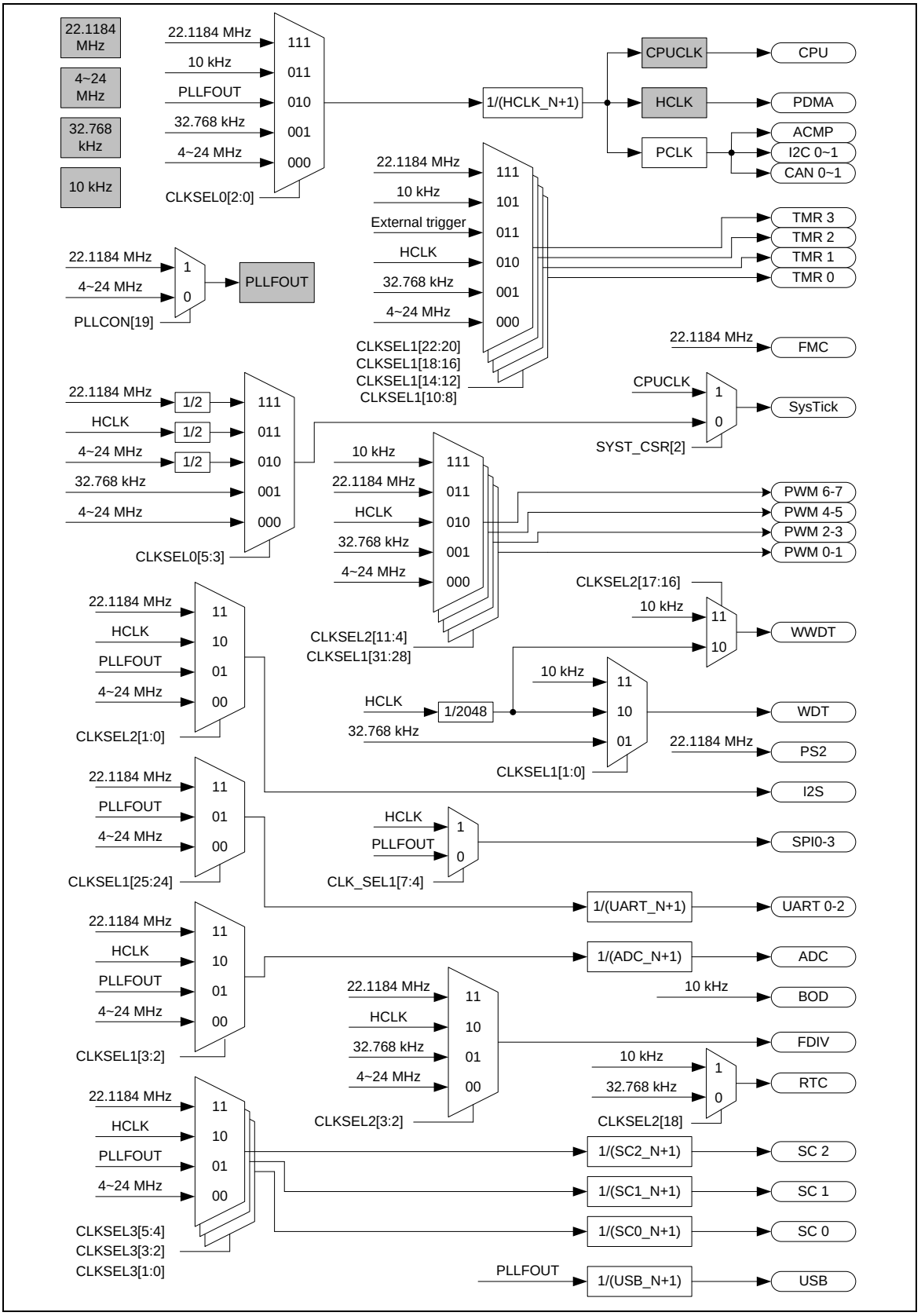


图 5-5 片上时钟源总览

5.3.2 系统时钟和 SysTick 时钟

系统时钟有 5 个时钟源，由时钟发生器发生。时钟源切换取决于寄存器HCLK_S (CLKSEL0[2:0]). 如 图 5-6 所示

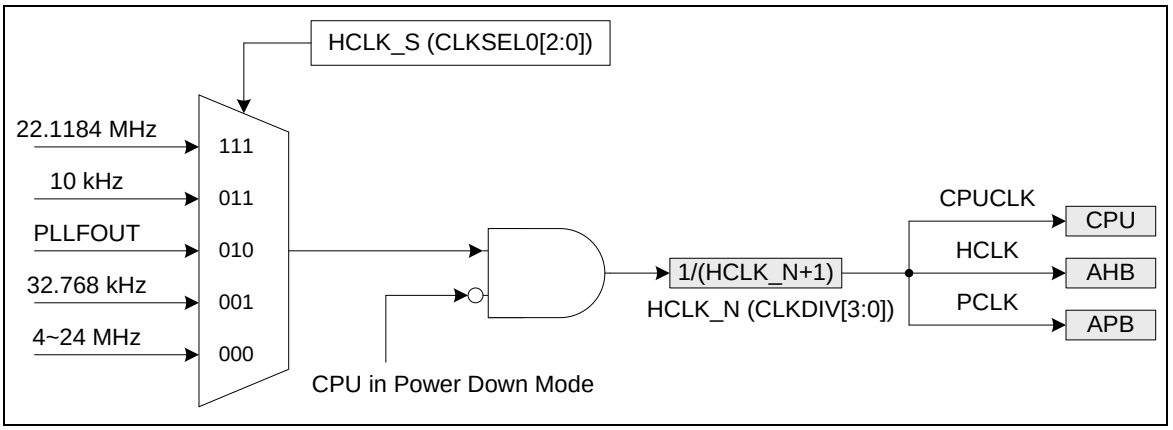


图 5-6 系统时钟框图

Cortex™-M0内核的SysTick 时钟源可以选择CPU时钟或外部时钟(SYST_CSR[2]).如果使用外部时钟，SysTick 时钟 (STCLK) 有 5 个时钟源.时钟源切换取决于寄存器 STCLK_S (CLKSEL0[5:3])。如图 5-7.

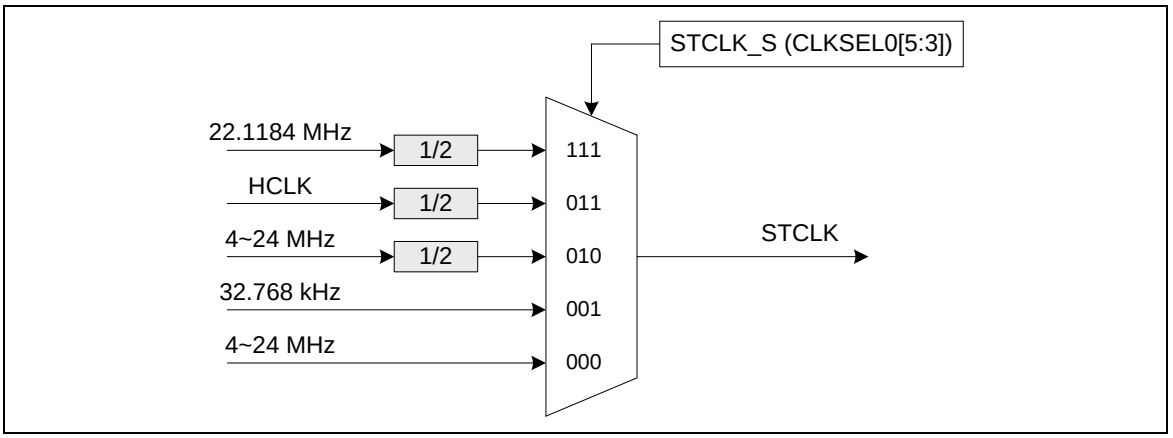


图 5-7 SysTick 时钟控制模块框图

5.3.3 掉电模式时钟

当芯片进入掉电模式，系统时钟，一些时钟源和外设时钟将被关闭。也有一些时钟源与外设时钟仍在工作。

如下时钟仍在工作：

- 时钟发生器
  - 10 kHz内部低速RC振荡器
  - 32.768 kHz外部低速晶振时钟
- RTC/WDT/Timer/PWM 外围 Clock (当时钟源来自32.768 kHz外部低速晶振时钟或10 kHz内部低速RC振荡器)

5.3.4 分频器输出

该设备包含一个16级2分频移位寄存器组成的分频器。其中哪一级的值被输出，由一个16选1的多路转换器选择，该多路转换器接到CLKO管脚上。因此有16种分频时钟选择，分频范围从Fin/21到Fin/216，其中Fin为输入到时钟分频器的时钟频率

输出公式：$F_{out} = F_{in}/2^{(N+1)}$，其中 Fin 为输入时钟频率，Fout 为时钟分频器输出频率，N 为FSEL(FRQDIV[3:0])中的4位值。往DIVIDER_EN (FRQDIV[4])写1，分级计数器开始计数。往DIVIDER_EN (FRQDIV[4])写0，分级计数器持续计数，直到分频时钟达到低电平并会保持在低电平状态。如果DIVIDER1(FRQDIV[5])设置为1，分频器时钟(FRQDIV_CLK)将忽略2分频器。时钟频率将直接输出到CLKO。

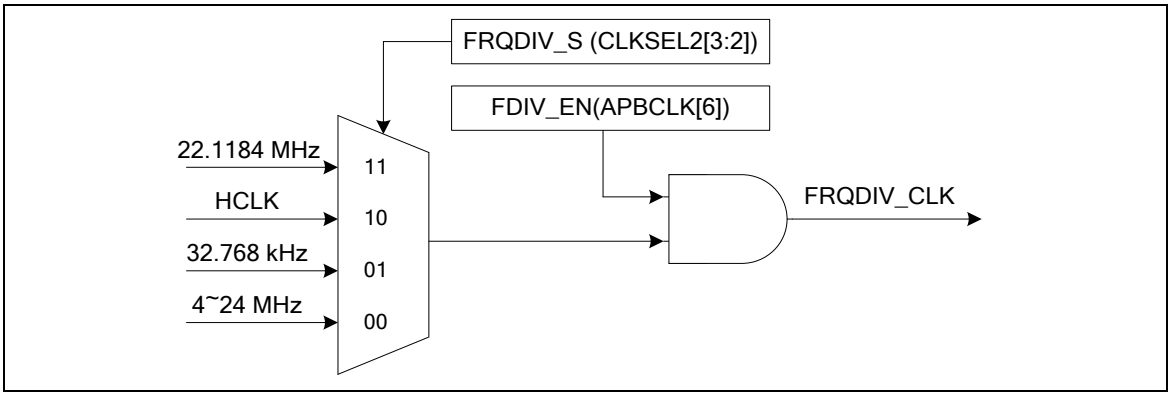


图 5-8 分频器的时钟源

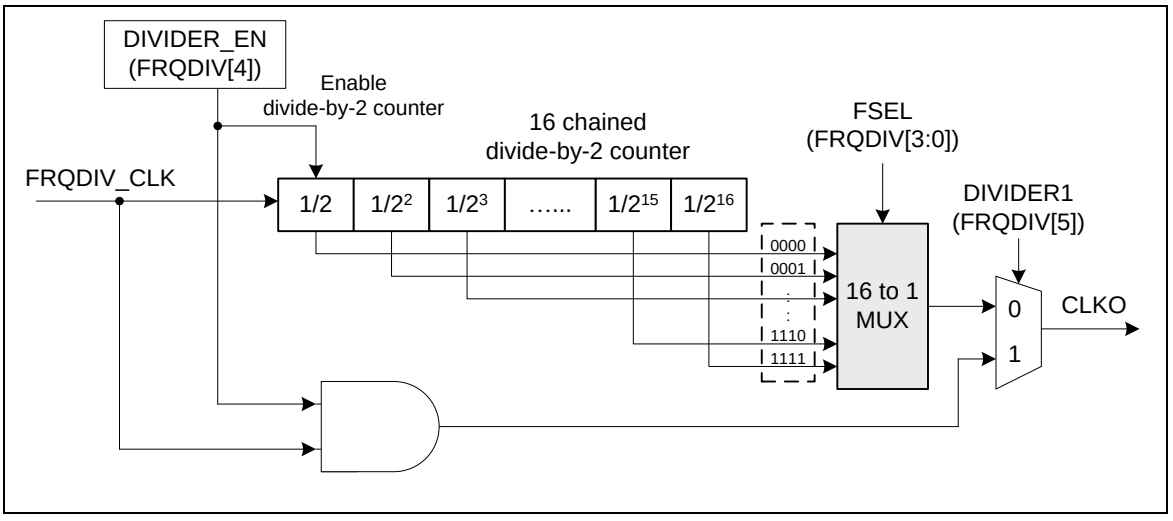


图 5-9 分频器模块框图

5.3.5 寄存器映射

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
CLK 基地址: **CLK_BA = 0x5000_0200**				
PWRCON	CLK_BA+0x00	R/W	系统掉电控制寄存器	0x0000_001X
AHBCLK	CLK_BA+0x04	R/W	AHB设备时钟使能控制寄存器	0x0000_0005
APBCLK	CLK_BA+0x08	R/W	APB设备时钟使能控制寄存器	0x0000_000X
APBCLK1	CLK_BA+0x30	R/W	APB设备时钟使能控制寄存器1	0x0000_0000
CLKSTATUS	CLK_BA+0x0C	R/W	时钟状态监控寄存器	0x0000_00XX
CLKSEL0	CLK_BA+0x10	R/W	时钟源选择控制寄存器0	0x0000_003X
CLKSEL1	CLK_BA+0x14	R/W	时钟源选择控制寄存器1	0xFFFF_FFFF
CLKSEL2	CLK_BA+0x1C	R/W	时钟源选择控制寄存器2	0x0002_00FF
CLKSEL3	CLK_BA+0x34	R/W	时钟源选择控制寄存器3	0x0000_003F
CLKDIV	CLK_BA+0x18	R/W	时钟分频数目寄存器	0x0000_0000
CLKDIV1	CLK_BA+0x38	R/W	时钟分频数目寄存器1	0x0000_0000
PLLCON	CLK_BA+0x20	R/W	PLL控制寄存器	0x0005_C22E
FRQDIV	CLK_BA+0x24	R/W	分频器控制寄存器	0x0000_0000

5.3.6 寄存器描述

系统掉电控制寄存器(PWRCON)

除 BIT[6]，PWRCON 的其他位都受保护，解锁这些位，需要向地址 0x5000_0100 依次写入 “59h”, “16h”, “88h”。 请参考寄存器 REGWRPROT （地址为 GCR_BA+0x100）

寄存器	偏移地址	R/W	描述	复位值
PWRCON	CLK_BA+0x00	R/W	系统掉电控制寄存器	0x0000_001X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							PD_WAIT_CPU
7	6	5	4	3	2	1	0
PWR_DOWN_EN	PD_WU_STS	PD_WU_INT_EN	PD_WU_DLY	OSC10K_EN	OSC22M_EN	XTL32K_EN	XTL12M_EN

位	描述	
[31:9]	**Reserved**	保留
[8]	**PD_WAIT_CPU**	**进入掉电条件控制 (写保护)** 0 =在 PWR_DOWN_EN 位置 1 时，芯片进入掉电模式。 1 =在 PD_WAIT_CPU 和 PWR_DOWN_EN 位都置 1 而且 CPU 执行 WFI 指令时，芯片进入掉电模式 **注意:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[7]	**PWR_DOWN_EN**	**系统掉电模式使能位(写保护)** 该位置 ‘1’，使能芯片的掉电模式，激活芯片的掉电行为取决于 PD_WAIT_CPU 位。 (a) 如果 PD_WAIT_CPU 为 0，则芯片会在置位 PWR_DOWN_EN 后，立即进入掉电模式。 (b) 如果 PD_WAIT_CPU 为 1，则芯片会一直运行到 CPU 的休眠模式被激活，然后芯片才会进入掉电模式。(推荐) 当芯片从掉电模式中被唤醒，该位自动清零。用户需要重新设置该位才能使能下一次的掉电。 在掉电模式下，外部 4~24 MHz 高速晶振与内部 22.1184 MHz高速RC振荡器将被禁用，但是外部32.768 kHz 低速晶振与内部 10 kHz低速振荡器不受掉电模式的控制。 在掉电模式下，PLL 与系统时钟将被禁用，忽略时钟源选择。如果外设时钟源为外部 32.768 kHz 低速晶振或者内部 10 kHz 低速振荡器，则外设的时钟不受掉电模式的控制。 0 = 执行 WFI 命令，芯片工作于正常模式或者芯片进入 idle 模式 1 = 芯片立即进入掉电模式或者等待 CPU 休眠命令 WFI **注意:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)

[6]	**PD_WU_STS**	**芯片掉电模式唤醒中断状态** 该位由掉电唤醒事件设置，表示从掉电模式恢复。 如果GPIO, USB, UART, WDT, I^2C, TIMER, ACMP, BOD 或 RTC被唤醒，该标志置位写 1 该位清零。 **注意：**只有在 PD_WU_INT_EN (PWRCON[5]) 被置 1 的时候，该位才工作。
[5]	**PD_WU_INT_EN**	**掉电模式唤醒中断使能（写保护位）** 0 = 掉电模式唤醒中断禁用 1 = 掉电模式唤醒中断使能 **注意1:** 当 PD_WU_STS 和 PD_WU_INT_EN 都为高时，中断将产生 **注意2:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[4]	**PD_WU_DLY**	**使能唤醒延时计数器（写保护位）** 当芯片从掉电模式中被唤醒时，时钟控制将会延迟一定的时钟周期以等待系统时钟稳定。 当芯片工作在外部 4~24 MHz高速晶振条件下时，延迟时钟周期为 4096 时钟周期；当芯片工作在内部 22.1184 MHz 高速振荡器条件下时，延迟时钟周期为 256 时钟周期。 0 = 禁用时钟周期的延迟 1 = 使能时钟周期的延迟 **注意:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[3]	**OSC10K_EN**	**内部10 KHz低速RC振荡器(LIRC)使能位（写保护）** 0 = 禁用10 kHz 内部低速 RC 振荡器(LIRC) 1 = 使能10 kHz 内部低速 RC 振荡器(LIRC) **注意:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[2]	**OSC22M_EN**	**22.1184 MHz 内部高速RC 振荡器 (HIRC)使能位 (写保护)** 0 = 禁用22.1184 MHz 内部高速RC振荡器(HIRC) 1 = 使能22.1184 MHz 内部高速RC振荡器(HIRC) **注意:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[1]	**XTL32K_EN**	**32.768 KHz 外部低速晶振（(LXT)）使能控制位(写保护)** 0 =禁用32.768 kHz 外部低速晶振(LXT) 1 =使能32.768 kHz 外部低速晶振(LXT) (正常操作). **注意:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)
[0]	**XTL12M_EN**	**4~24 MHz 外部高速晶振(HXT)使能位（写保护）** 该位的默认值由 flash 控制器的用户配置寄存器CONFIG0 [26:24] 设置。当默认的时钟源为外部 4~24 MHz 高速晶振时，该位自动置 1。 0 = 禁用4 ~ 24 MHz 外部高速晶振(HXT) 1 = 使能4~24 MHz 外部高速晶振(HXT) **注意:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)

寄存器或指令模式	SLEEPDEEP (SCR[2])	PD_WAIT_CPU (PWRCON[8])	PWR_DOWN_EN (PWRCON[7])	CPU 运行 WFI 指令	禁用时钟
正常运行模式	0	0	0	NO	通过控制寄存器禁用所有时钟
Idle 模式 (CPU 进入休眠模式)	0	x	0	YES	仅禁用 CPU 时钟
掉电模式 (CPU 进入深度休眠模式)	1	1	1	YES	大部分时钟被禁用，仅 10 kHz/32.768 kHz 及选他们为时钟源的 RTC/WDT/Timer/PWM 外设时钟可运行。

表 5-5 掉电模式控制表

当芯片进入掉电模式时，用户可以通过一些中断源唤醒芯片。用户必须在设置 PWR_DOWN_EN 位 (PWRCON[7]) 之前，使能相关的中断源及相应的NVIC IRQ 使能位 (NVIC_ISER) 从而保证芯片能进入掉电模式以及能够成功被唤醒。

AHB 设备时钟使能控制寄存器(AHBCLK)

该寄存器各位用于使能/禁用系统时钟，PDMA时钟

寄存器	偏移地址	R/W	描述	复位值
AHBCLK	CLK_BA+0x04	R/W	AHB 设备时钟使能控制寄存器	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
Reserved				EBI_EN	ISP_EN	PDMA_EN	Reserved

位	描述	
[31:3]	**Reserved**	保留
[3]	**EBI_EN**	**EBI 控制器时钟使能位** 1 = EBI 时钟使能. 0 = EBI 时钟关闭
[2]	**ISP_EN**	**ISP 控制器时钟使能位** 0 = 禁用Flash ISP外围时钟 1 = 使能Flash ISP外围时钟
[1]	**PDMA_EN**	**PDMA控制器时钟使能位** 0 = 禁用PDMA外围时钟 1 = 使能PDMA外围时钟.
[0]	**Reserved**	保留

APB 设备时钟使能控制寄存器(APBCLK)

该寄存器各位用于使能/禁用外设控制器时钟

寄存器	偏移地址	R/W	描述	复位值
APBCLK	CLK_BA+0x08	R/W	APB 设备时钟使能控制寄存器	0x0000_000X

31	30	29	28	27	26	25	24
PS2_EN	ACMP_EN	I2S_EN	ADC_EN	USBD_EN	Reserved	CAN1_EN	CAN0_EN
23	22	21	20	19	18	17	16
PWM67_EN	PWM45_EN	PWM23_EN	PWM01_EN	Reserved	UART2_EN	UART1_EN	UART0_EN
15	14	13	12	11	10	9	8
SPI3_EN	SPI2_EN	SPI1_EN	SPI0_EN	Reserved		I2C1_EN	I2C0_EN
7	6	5	4	3	2	1	0
Reserved	FDIV_EN	TMR3_EN	TMR2_EN	TMR1_EN	TMR0_EN	RTC_EN	WDT_EN

位	描述	
[31]	PS2_EN	**PS/2时钟使能位** 0 = 禁用PS/2时钟 1 = 使能PS/2时钟
[30]	ACMP_EN	**模拟比较器时钟使能位** 0 = 禁用模拟比较器时钟 1 = 使能模拟比较器时钟
[29]	I2S_EN	**I^2S 时钟使能位** 0 = 禁用I^2S时钟 1 = 使能I^2S时钟
[28]	ADC_EN	**ADC 时钟使能位** 0 = 禁用ADC 时钟 1 = 使能ADC 时钟
[27]	USBD_EN	**USB 2.0 FS设备控制器时钟使能位** 0 = 禁用USB时钟 1 = 使能USB时钟
[26]	Reserved	保留
[25]	CAN1_EN	**CAN 总线控制器-1时钟使能位** 0 = 禁用CAN1 时钟 1 = 使能CAN1 时钟
[24]	CAN0_EN	**CAN 总线控制器-0时钟使能位** 0 = 禁用CAN0 时钟 1 = 使能CAN0 时钟

[23]	**PWM67_EN**	**PWM_67时钟使能位** 0 = 禁用PWM67时钟 1 = 使能PWM67时钟
[22]	**PWM45_EN**	**PWM_45时钟使能位** 0 = 禁用PWM45 时钟 1 = 使能PWM45 时钟
[21]	**PWM23_EN**	**PWM_23时钟使能位** 0 = 禁用PWM23时钟 1 = 使能PWM23时钟
[20]	**PWM01_EN**	**PWM_01时钟使能位** 0 = 禁用PWM01时钟 1 = 使能PWM01时钟
[19]	**Reserved**	保留
[18]	**UART2_EN**	**UART2时钟使能位** 0 = 禁用UART2时钟 1 = 使能UART2时钟
[17]	**UART1_EN**	**UART1时钟使能位** 0 =禁用 UART1时钟 1 =使能 UART1时钟
[16]	**UART0_EN**	**UART0时钟使能位** 0 = 禁用UART0时钟 1 = 使能UART0时钟
[15]	**SPI3_EN**	**SPI3时钟使能位** 0 = 禁用SPI3时钟 1 = 使能SPI3时钟
[14]	**SPI2_EN**	**SPI2时钟使能位** 0 = 禁用SPI2时钟 1 = 使能SPI2时钟
[13]	**SPI1_EN**	**SPI1时钟使能位** 0 = 禁用SPI1时钟 1 = 使能SPI1时钟
[12]	**SPI0_EN**	**SPI0时钟使能位** 0 = 禁用SPI0时钟 1 = 使能SPI0时钟
[11:10]	**Reserved**	保留
[9]	**I2C1_EN**	**I^2C1时钟使能位** 0 = 禁用I^2C1时钟 1 = 使能I^2C1时钟
[8]	**I2C0_EN**	**I^2C0时钟使能位**

		0 = 禁用I²C0时钟 1 = 使能I²C0时钟
[7]	**Reserved**	保留
[6]	**FDIV_EN**	**分频器输出时钟使能位** 0 =禁用 FDIV时钟 1 =使能 FDIV时钟
[5]	**TMR3_EN**	**Timer3时钟使能位** 0 = 禁用Timer3时钟 1 = 使能Timer3时钟
[4]	**TMR2_EN**	**Timer2时钟使能位** 0 = 禁用Timer2时钟 1 = 使能Timer2时钟
[3]	**TMR1_EN**	**Timer1时钟使能位** 0 = 禁用Timer1 时钟 1 = 使能Timer1 时钟
[2]	**TMR0_EN**	**Timer0时钟使能位** 0 = 禁用Timer0时钟 1 = 使能Timer0时钟
[1]	**RTC_EN**	**Real-Time-Clock APB 接口时钟使能位** 该位仅用于控制 RTC APB 时钟，RTC 外设的时钟源由RTC_SEL_10K(CLKSEL2[18])寄存器设置。该时钟源由32.768 kHz外部低速晶振或10 kHz低速RC振荡器提供。 0 = 禁用RTC 时钟 1 = 使能RTC 时钟
[0]	**WDT_EN**	**Watchdog 定时器时钟使能位(写保护)** **0 = 禁用Watchdog 时钟** **1 = 使能Watchdog 时钟** **注意:** 该位受保护，编程该位时，需要依次向地址0x5000_0100写入“59h”, “16h”, “88h” 该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)

APB设备时钟使能寄存器1 (APBCLK1)

该寄存器各位用于使能/禁用外设控制器时钟.

寄存器	偏移地址	R/W	描述	复位后数值
APBCLK1	CLK_BA+0x30	R/W	APB设备时钟使能寄存器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					SC2_EN	SC1_EN	SC0_EN

位	描述	
[31:3]	Reserved	保留
[2]	SC2_EN	SC2时钟使能位 0 = 禁用SC2时钟 1 = 使能SC2时钟
[1]	SC1_EN	SC1时钟使能位 0 = 禁用SC1时钟 1 = 使能SC1时钟
[0]	SC0_EN	SC0时钟使能位 0 = 禁用SC0时钟 1 = 使能SC0时钟

时钟状态寄存器(CLKSTATUS)

该寄存器各位用于监控芯片时钟源是否稳定，时钟切换是否失败。

寄存器	偏移地址	R/W	描述	复位值
CLKSTATUS	CLK_BA+0x0C	R/W	时钟状态监控寄存器	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
CLK_SW_FAIL	Reserved		OSC22M_STB	OSC10K_STB	PLL_STB	XTL32K_STB	XTL12M_STB

位	描述	
[31:8]	Reserved	保留
[7]	CLK_SW_FAIL	**时钟切换失败标志（只读）** 0 = 时钟切换成功 1 = 时钟切换失败 该位是一个标志，表达目前系统时钟源是否与用户在HCLK_S (CLKSEL[2:0])寄存器中定义相同。当用户切换系统时钟，系统时钟将保持原时钟，直到新时钟稳定。在新时钟稳定前这段时间，该位用于指示是否是用户要设定的系统时钟源。
[6:5]	Reserved	保留
[4]	OSC22M_STB	**内部22.1184 MHz高速RC振荡器(HIRC)时钟源稳定标志（只读）** 0 = 22.1184 MHz 内部高速RC 振荡器(HIRC) 时钟不稳定或者禁用 1 = 22.1184 MHz 内部高速RC 振荡器(HIRC) 时钟使能并稳定
[3]	OSC10K_STB	**内部10 KHz 低速振荡器 (LIRC) 时钟源稳定标志（只读）** 0 = 内部10 kHz低速RC振荡器 (LIRC) 时钟不稳定或者禁用 1 =内部10 kHz低速RC振荡器(LIRC) 时钟使能并稳定
[2]	PLL_STB	**内部 PLL 时钟源稳定标志(只读)** 0 = 内部 PLL时钟不稳定或者禁用 1 = 内部PLL 时钟稳定为正常模式
[1]	XTL32K_STB	**外部32.768 KHz 低速晶振 (LXT) 时钟源稳定标志（只读）** 0 = 外部32.768 kHz低速晶振(LXT) 时钟不稳定或者禁用 1 = 外部32.768 kHz低速晶振(LXT) 时钟使能并稳定
[0]	XTL12M_STB	**外部4~24 MHz 高速晶振 (HXT) 时钟源稳定标志（只读）** 0 = 外部4~24 MHz高速晶振(HXT) 时钟不稳定或者禁用

		1 = 外部4~24 MHz高速晶振(HXT) 时钟使能并稳定

时钟源选择控制寄存器0 (CLKSEL0)

寄存器	偏移地址	R/W	描述	复位值
CLKSEL0	CLK_BA+0x10	R/W	时钟源选择控制寄存器0	0x0000_003X

31	30	29	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		**STCLK_S**			**HCLK_S**		

位	描述	
[31:6]	**Reserved**	保留
[5:3]	**STCLK_S**	**Cortex™-M0 SysTick 时钟源选择(写保护)** 如果SYST_CSR[2] = 1, SysTick 时钟源来自 HCLK 如果SYST_CSR[2] = 0, SysTick 时钟源由STCLK_S(CLKSEL0[5:3])定义. 000 = 时钟源为外部 4~24 MHz 高速晶振时钟 001 = 时钟源为外部 32.768 kHz 低速晶振时钟 010 = 时钟源为外部 4~24 MHz高速 晶振时钟/2分频 011 = 时钟源为 HCLK/2分频 111 = 时钟源为内部 22.1184 MHz 高速振荡器时钟/2分频 **注意1:** 该位是受保护位。编程时，需要向地址 0x5000_0100 依次写入 “59h”, “16h”, “88h” 来解锁寄存器保护。请参考寄存器 REGWRPROT （地址：GCR_BA+0x100）。 **注意2:** 如果 SysTick 时钟源不是来自 HCLK (例如 SYST_CSR[2] = 0), SysTick 时钟源必须小于或等于HCLK/2
[2:0]	**HCLK_S**	**HCLK时钟源选择(写保护)** 1. 在时钟切换前，相关时钟源（预选和新选）必须打开。 2. 在任何复位后，这 3 位的默认值将从Flash 控制器的用户配置寄存器 CFOSC (CONFIG0 [26:24]) 加载。因此默认值可能为 000b 或者 111b。 3. 该位是受保护位。编程时，需要向地址 0x5000_0100 依次写入 “59h”, “16h”, “88h” 来解锁寄存器保护。请参考寄存器 REGWRPROT （地址：GCR_BA+0x100） 000 = 时钟源为外部 4~24 MHz高速 晶振时钟 001 = 时钟源为外部 32.768 kHz 低速晶振时钟 010 = 时钟源为 PLL 时钟 011 = 时钟源为内部 10 kHz 低速振荡器时钟 111 = 时钟源为内部 22.1184 MHz高速 振荡器时钟

时钟源选择控制寄存器1(CLKSEL1)

在时钟切换之前，必须打开相关的时钟源（预选和新选)。.

寄存器	偏移地址	R/W	描述	复位值
CLKSEL1	CLK_BA+0x14	R/W	时钟源选择控制寄存器1	0xFFFF_FFFF

31	30	29	28	27	26	25	24
PWM23_S		PWM01_S		Reserved		UART_S	
23	22	21	20	19	18	17	16
Reserved	TMR3_S			Reserved	TMR2_S		
15	14	13	12	11	10	9	8
Reserved	TMR1_S			Reserved	TMR0_S		
7	6	5	4	3	2	1	0
SPI3_S	SPI2_S	SPI1_S	SPI0_S	ADC_S		WDT_S	

位	描述	
[31:30]	**PWM23_S**	**PWM2 和 PWM3时钟源选择** PWM2 和 PWM3使用相同的时钟源和相同的分频。PWM2和PWM3的时钟源由PWM23_S (CLKSEL1[31:30]) 和PWM23_S_E (CLKSEL2[9] 定义。 如果PWM23_S_E = 0，PWM2和PWM3时钟源由PWM23_S定义，如下： 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 01 = 时钟源为外部 32.768 kHz低速 晶振时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟 如果PWM23_S_E = 1，PWM2和PWM3时钟源由PWM23_S定义，如下： 00 =保留 01 =保留 10 =保留 11 =时钟源为内部10 kHz 低速振荡器时钟
[29:28]	**PWM01_S**	**PWM0 和 PWM1 时钟源选择** PWM0和PWM1使用相同的时钟源和相同的分频。PWM0和PWM1的时钟源由PWM01_S (CLKSEL1[29:28]) 和PWM01_S_E (CLKSEL2[8])定义。 如果PWM01_S_E = 1，PWM0和PWM1时钟源由PWM01_S定义，如下： .00 = 时钟源为外部 4~24 MHz 高速晶振时钟 .01 = 时钟源为外部 32.768 kHz低速 晶振时钟 .10 = 时钟源为 HCLK .11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟 如果PWM01_S_E = 1，PWM0和PWM1时钟源由PWM01_S定义，如下： .00 = 保留 .01 = 保留 .10 = 保留

		.11 = 时钟源为内部10 kHz 低速振荡器时钟
[27:26]	**Reserved**	保留
[25:24]	**UART_S**	**UART时钟源选择** 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 01 = 时钟源为 PLL 时钟 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟
[23]	**Reserved**	保留
[22:20]	**TMR3_S**	**TIMER3时钟源选择** 000 = 时钟源为外部 4~24 MHz 高速晶振时钟 001 = 时钟源为外部 32.768 kHz低速晶振时钟 010 = 时钟源为 HCLK 011 = 时钟源为外部触发 101 = 时钟源为内部10 kHz低速振荡器时钟 111 = 时钟源为内部 22.1184 MHz高速振荡器时钟 Others = 保留
[19]	**Reserved**	保留
[18:16]	**TMR2_S**	**TIMER2时钟源选择** 000 = 时钟源为外部 4~24 MHz 高速晶振时钟 001 = 时钟源为外部 32.768 kHz低速晶振时钟 010 = 时钟源为 HCLK 011 = 时钟源为外部触发 101 = 时钟源为内部10 kHz低速振荡器时钟 111 = 时钟源为内部 22.1184 MHz高速振荡器时钟 Others = 保留
[15]	**Reserved**	保留
[14:12]	**TMR1_S**	**TIMER1时钟源选择** 000 = 时钟源为外部 4~24 MHz 高速晶振时钟 001 = 时钟源为外部 32.768 kHz低速晶振时钟 010 = 时钟源为 HCLK 011 = 时钟源为外部触发 101 = 时钟源为内部10 kHz低速振荡器时钟 111 = 时钟源为内部 22.1184 MHz高速振荡器时钟 Others = 保留
[11]	**Reserved**	保留
[10:8]	**TMR0_S**	**TIMER0 Clock Source Selection** 000 = 时钟源为外部 4~24 MHz 高速晶振时钟 001 = 时钟源为外部 32.768 kHz低速晶振时钟 010 = 时钟源为 HCLK 011 = 时钟源为外部触发 101 = 时钟源为内部10 kHz低速振荡器时钟 111 = 时钟源为内部 22.1184 MHz高速振荡器时钟 Others = 保留

[7]	**SPI3_S**	**SPI3时钟源选择** 0 =时钟源来自PLL 时钟 1 =时钟源来自HCLK
[6]	**SPI2_S**	**SPI2时钟源选择** 0 =时钟源来自PLL 时钟 1 =时钟源来自HCLK
[5]	**SPI1_S**	**SPI1时钟源选择** 0 =时钟源来自PLL 时钟 1 =时钟源来自HCLK
[4]	**SPI0_S**	**SPI0时钟源选择** 0 =时钟源来自PLL 时钟 1 =时钟源来自HCLK
[3:2]	**ADC_S**	**ADC时钟源选择** 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 01 = 时钟源来自 PLL 时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟
[1:0]	**WDT_S**	**看门狗定时器时钟源选择（写保护位）** 00 = 保留 01 = 时钟源为外部 32.768 kHz 低速晶振时钟 10 = 时钟源为 HCLK /2048 clock 11 = 时钟源为内部 10 kHz 低速振荡器时钟 **注意:** 该位是受保护位。编程时，需要向地址 0x5000_0100 依次写入 “59h”, “16h”, “88h” 来解锁寄存器保护。请参考寄存器 REGWRPROT （地址：GCR_BA+0x100）。

时钟源选择控制寄存器2 (CLKSEL2)

在时钟切换之前，必须打开相关的时钟源（预选和新选）。

寄存器	偏移地址	R/W	描述	复位值
CLKSEL2	CLK_BA+0x1C	R/W	时钟源选择控制寄存器2	0x0002_00FF

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**RTC_SEL_10K**	**WWDT_S**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**PWM67_S_E**	**PWM45_S_E**	**PWM23_S_E**	**PWM01_S_E**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PWM67_S		**PWM45_S**		**FRQDIV_S**		**I2S_S**	

位	描述	
[31:19]	**Reserved**	保留
[18]	**RTC_SEL_10K**	**RTC 时钟源选择** 0 = 时钟源为外部低速32.768 kHz晶振 1 = 时钟源为内部10 kHz低速振荡器
[17:16]	**WWDT_S**	**窗口看门狗时钟源选择** 10 = 时钟源为HCLK/2048 时钟. 11 = 时钟源为内部10 kHz低速振荡器时钟
[15:12]	**Reserved**	保留
[11]	**PWM67_S_E**	**PWM6和PWM7时钟源扩充选择** PWM6和PWM7使用相同的时钟源和相同的分频。PWM6和PWM7的时钟源由PWM67_S (CLKSEL2[7:6])和PWM67_S_E (CLKSEL2[11])定义。 如果PWM67_S_E = 0，PWM6和PWM7时钟源由PWM67_S定义，如下： .00 = 时钟源为外部 4~24 MHz 高速晶振时钟 .01 = 时钟源为外部 32.768 kHz低速 晶振时钟 .10 = 时钟源为 HCLK .11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟 如果PWM67_S_E = 1，PWM6和PWM7时钟源由PWM67_S定义，如下： .00 = 保留 .01 = 保留 .10 = 保留 .11 = 时钟源为内部10 kHz 低速振荡器时钟
[10]	**PWM45_S_E**	**PWM4和PWM5时钟源扩充选择**

		PWM4和PWM5使用相同的时钟源和相同的分频。PWM4和PWM5的时钟源由PWM45_S (CLKSEL2[5:4])和PWM45_S_E (CLKSEL2[10])定义。 如果PWM45_S_E = 0，PWM4和PWM5时钟源由PWM45_S定义，如下： . 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 . 01 = 时钟源为外部 32.768 kHz低速 晶振时钟 . 10 = 时钟源为 HCLK . 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟 如果PWM45_S_E = 1，PWM4和PWM5时钟源由PWM45_S定义，如下： .00 =保留 .01 =保留 .10 =保留 .11 =时钟源为内部10 kHz 低速振荡器时钟
[9]	**PWM23_S_E**	**PWM2和PWM3时钟源扩充选择** PWM2和PWM3使用相同的时钟源和相同的分频。PWM2和PWM3的时钟源由PWM23_S (CLKSEL1[31:30])和PWM23_S_E (CLKSEL2[9])。 如果PWM23_S_E = 0，PWM2和PWM3时钟源由PWM23_S定义，如下： . 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 . 01 = 时钟源为外部 32.768 kHz低速 晶振时钟 . 10 = 时钟源为 HCLK . 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟 如果PWM23_S_E = 1，PWM2和PWM3时钟源由PWM23_S定义，如下： .00 =保留 .01 =保留 .10 =保留 .11 =时钟源为内部10 kHz 低速振荡器时钟
[8]	**PWM01_S_E**	**PWM0和PWM1时钟源扩充选择** PWM0和PWM1使用相同的时钟源和相同的分频。PWM0和PWM1的时钟源由PWM01_S (CLKSEL1[29:28])和PWM01_S_E (CLKSEL2[8])。 如果PWM01_S_E = 0，PWM0和PWM1时钟源由PWM01_S定义，如下： . 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 . 01 = 时钟源为外部 32.768 kHz低速 晶振时钟 . 10 = 时钟源为 HCLK . 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟 如果PWM01_S_E = 1，PWM0和PWM1时钟源由PWM01_S定义，如下： .00 =保留 .01 =保留 .10 =保留 .11 =时钟源为内部10 kHz 低速振荡器时钟
[7:6]	**PWM67_S**	**PWM6和PWM7 时钟源选择** PWM6和PWM7使用相同的时钟源和相同的分频。PWM6和PWM7的时钟源由PWM67_S (CLKSEL2[7:6])和PWM67_S_E (CLKSEL2[11])定义。 如果PWM67_S_E = 0，PWM6和PWM7时钟源由PWM67_S定义，如下： .00 = 时钟源为外部 4~24 MHz 高速晶振时钟 .01 = 时钟源为外部 32.768 kHz低速 晶振时钟 .10 = 时钟源为 HCLK

		.11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟 如果PWM67_S_E = 1，PWM6和PWM7时钟源由PWM67_S定义，如下： .00 = 保留 .01 = 保留 .10 = 保留 .11 = 时钟源为内部10 kHz 低速振荡器时钟
[5:4]	**PWM45_S**	**PWM4和PWM5时钟源选择** PWM4和PWM5使用相同的时钟源和相同的分频。PWM4和PWM5的时钟源由PWM45_S (CLKSEL2[5:4])和PWM45_S_E (CLKSEL2[10])定义。 如果PWM45_S_E = 0，PWM4和PWM5时钟源由PWM45_S定义，如下： . 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 . 01 = 时钟源为外部 32.768 kHz低速 晶振时钟 . 10 = 时钟源为 HCLK . 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟 如果PWM45_S_E = 1，PWM4和PWM5时钟源由PWM45_S定义，如下： .00 =保留 .01 =保留 .10 =保留 .11 =时钟源为内部10 kHz 低速振荡器时钟
[3:2]	**FRQDIV_S**	**时钟分频器时钟源选择** 00 = 时钟源为外部 4~24 MHz高速 晶振时钟k 01 = 时钟源为外部 32.768 kHz 低速晶振时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟
[1:0]	**I2S_S**	**I^2S时钟源选择** 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 01 = 时钟源为 PLL 时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟

时钟源选择控制寄存器3 (CLKSEL3)

在时钟切换之前，必须打开相关的时钟源（预选和新选）

寄存器	偏移地址	R/W	描述	复位值
CLKSEL3	CLK_BA+0x34	R/W	时钟源选择控制寄存器 3	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		SC2_S		SC1_S		SC0_S	

位	描述	
[31:6]	Reserved	保留
[5:4]	SC2_S	**SC2时钟源选择** 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 01 = 时钟源为 PLL 时钟 10 = HCLK 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟
[3:2]	SC1_S	**SC1 时钟源选择** 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 01 = 时钟源为 PLL 时钟 10 = HCLK 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟
[1:0]	SC0_S	**SC0 时钟源选择** 00 = 时钟源为外部 4~24 MHz 高速晶振时钟 01 = 时钟源为 PLL 时钟 10 = HCLK 11 = 时钟源为内部 22.1184 MHz 高速振荡器时钟

时钟分频寄存器 (CLKDIV)

寄存器	偏移地址	R/W	描述	复位值
CLKDIV	CLK_BA+0x18	R/W	时钟分频寄存器	0x0000_0000

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

位	描述	
[15:12]	**Reserved**	保留
[23:16]	**ADC_N**	**ADC时钟源的时钟除频数** ADC 时钟频率 =（ADC 时钟源频率）/ (ADC_N + 1)
[15:12]	**Reserved**	保留
[11:8]	**UART_N**	**UART 时钟源的时钟除频数** UART 时钟频率=（UART 时钟源频率）/ (UART_N + 1)
[7:4]	**USB_N**	**USB 时钟由 PLL 时钟除频数** USB 时钟频率 =（PLL 频率）/ (USB_N + 1)
[3:0]	**HCLK_N**	**HCLK 时钟源的时钟除频数** HCLK 时钟频率 =（HCLK 时钟源频率）/ (HCLK_N + 1)

时钟分频寄存器1 (CLKDIV1)

寄存器	偏移地址	R/W	描述	复位值
CLKDIV1	CLK_BA+0x38	R/W	时钟分频寄存器1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SC2_N							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SC1_N							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SC0_N							

位	描述	
[31:24]	**Reserved**	保留
[23:16]	**SC2_N**	**SC2时钟源的时钟除频数** SC2 时钟频率 = (SC2 时钟源频率) / (SC2_N + 1)
[15:8]	**SC1_N**	**SC1时钟源的时钟除频数** SC1 时钟频率 = (SC1 时钟源频率) / (SC1_N + 1)
[7:0]	**SC0_N**	**SC0时钟源的时钟除频数** SC0 时钟频率 = (SC0 时钟源频率) / (SC0_N + 1)

PLL控制寄存器(PLLCON)

PLL 的参考时钟源来自外部 4~24 MHz高速晶振时钟输入或者内部 22.1184 MHz 高速振荡器。该寄存器用于控制 PLL 的输出频率和 PLL 的操作模式。

寄存器	偏移地址	R/W	描述	复位值
PLLCON	CLK_BA+0x20	R/W	PLL 控制寄存器	0x0005_C22E

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

位	描述	
[31:20]	Reserved	保留
[19]	PLL_SRC	PLL 时钟源选择 0 = PLL 时钟源为外部 4~24 MHz 高速晶振 1 = PLL 时钟源为内部 22.1184 MHz 高速振荡器
[18]	OE	PLL OE (FOUT enable) 管脚控制 0 = PLL FOUT 使能 1 = PLL FOUT 固定为低
[17]	BP	PLL 旁路控制 0 = PLL 正常模式（默认） 1 = PLL 时钟输出与PLL源时钟输入相同
[16]	PD	掉电模式 如果设置寄存器 PWRCON 的 PWR_DOWN_EN 位 为 1，PLL 将进入掉电模式。 0 = PLL 正常模式 1 = PLL 进入掉电模式（默认）
[15:14]	OUT_DV	PLL 输出分频控制位 参考下表公式
[13:9]	IN_DV	PLL 输入分频控制位 参考下表公式
[8:0]	FB_DV	PLL 反馈分频控制位 参考下表公式

输出时钟频率设置

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

约束条件:

1. $3.2MHz < FIN < 150MHz$
2. $800KHz < \frac{FIN}{2 * NR} < 7.5MHz$
3. $100MHz < FCO = FIN * \frac{NF}{NR} < 200MHz$

   $120MHz <$ FCO 最佳

符号	描述
FOUT	输出时钟频率
FIN	输入（参考）时钟频率
NR	输入分频 (IN_DV + 2)
NF	反馈分频 (FB_DV + 2)
NO	OUT_DV = “00” : NO = 1 OUT_DV = “01” : NO = 2 OUT_DV = “10” : NO = 2 OUT_DV = “11” : NO = 4

默认频率设置

默认值: 0xC22E
FIN = 12 MHz
NR = (1+2) = 3
NF = (46+2) = 48
NO = 4
FOUT = 12/4 x 48 x 1/3 = 48 MHz

频率分频器控制寄存器(FRQDIV)

寄存器	偏移地址	R/W	描述	复位值
FRQDIV	CLK_BA+0x24	R/W	频率分频控制寄存器	0x0000_0000

31	30	29	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	**CLKO_1HZ_EN**	**DIVIDER1**	**DIVIDER_EN**	**FSEL**			

位	描述	
[31:7]	**Reserved**	保留
[6]	**CLKO_1HZ_EN**	**时钟1Hz输出使能位** 0 = 禁用1 Hz 时钟为外部32.768 kHz低速晶振输出补偿 1 = 使能1 Hz 时钟为外部32.768 kHz低速晶振输出补偿
[5]	**DIVIDER1**	**分频器1使能位** 0 = 分频器输出的时钟来自由FSEL分频的时钟源 1 = 分频器输出的时钟来自时钟源
[4]	**DIVIDER_EN**	**频率分频器使能位** 0 = 禁用频率分频器 1 = 使能频率分频器
[3:0]	**FSEL**	**分频器输出频率选择位** 输出频率的公式: $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$. F_{in}为输入时钟频率 F_{out}为分频器输出时钟频率 N为FSEL[3:0] 的4位值。

5.4 Flash 存储控制器(FMC)

5.4.1 概述

NuMicro™ NUC200系列 具有128/64/32K 字节的片上FLASH，用于存储应用程序（APROM）,用户可以通过ISP更新这些FLASH. 在系统编程(ISP)功能，用户可以通过该功能直接更新已经焊接在PCB板上芯片的程序。上电后，Config0的启动选择(CBS)决定Cortex-M0 CPU从APROM或LDROM读取代码。

NuMicro™ NUC200系列也提供了数据 flash，在芯片掉电之前，用来存储数据.对于APROM是128K字节的芯片数据 flash是跟APROM 共用，起始地址由Config1配置。对于APROM是64K/32K字节的芯片，数据 flash固定4KB。

5.4.2 特性

- 连续地址读访问零等待状态时，最高可达50 MHz，连续地址读访问一个周期等待状态时，最高可达72 MHz
- 所有嵌入Flash支持512字节页擦除。
- 128/64/32 KB 应用程序存储空间(APROM)
- 8KB在系统编程 (ISP) 加载程序空间(LDROM)
- 64K/32K字节APROM的设备固定有4kB数据FLASH。
- 128K字节 APROM的设备是可配置数据FLASH。
- 可配置或固定4 KB的数据FLASH，页擦除单元为512字节
- 支持在应用编程(IAP)，可在APROM 和LDROM之间程序切换，不用复位
- 支持在系统编程(ISP)更新片上Flash 。

5.4.3 框图

FLASH存储器控制器包括AHB从接口，ISP 控制逻辑，烧写器接口和FLASH宏接口时序控制逻辑。

FLASH存储器控制器框图如下图所示：

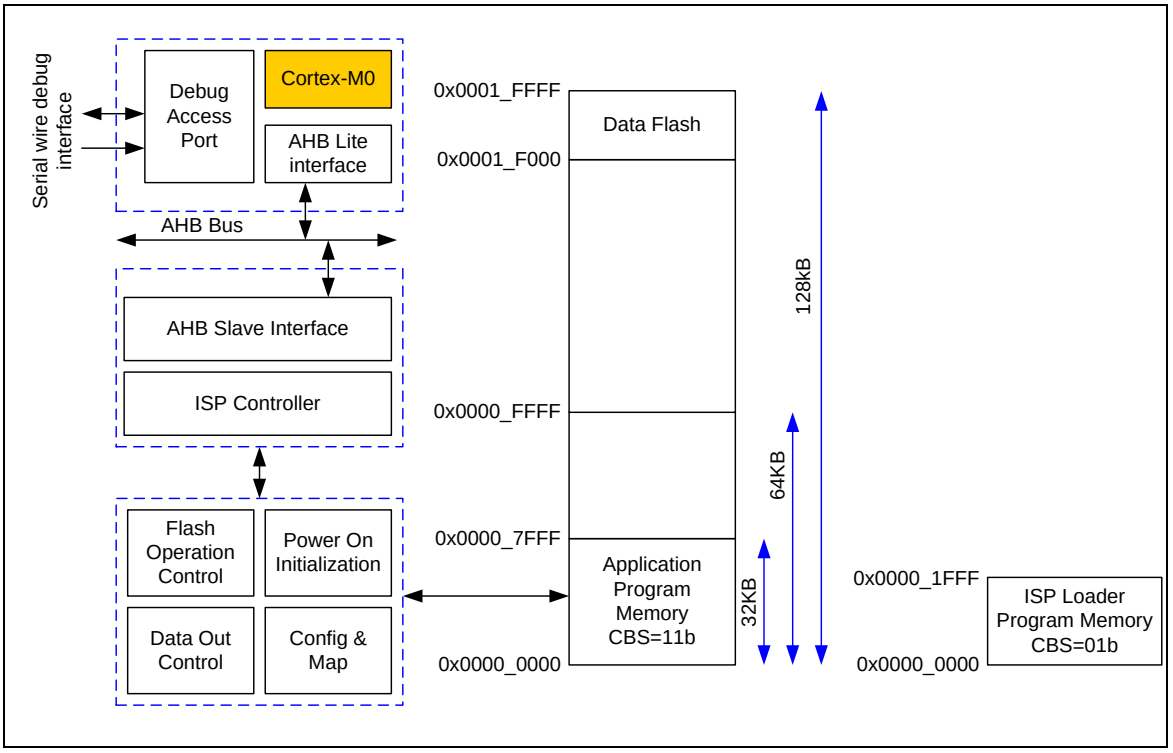


图 5-10 Flash 存储器控制模块框图

5.4.4 功能描述

5.4.4.1 *FMC 存储器结构*

NuMicro™ NUC200系列的flash存储器由程序存储器(APROM)，数据FLASH，ISP载入程序存储器(LDROM)，和用户配置区组成。

程序存储器是用户应用的主要存储器，叫做APROM，用户可以将他们的应用程序写到APROM并设置系统从APROM启动。

设计ISP 装载程序存储器是用来实现在系统编程（ISP）功能。LDROM独立于APROM，也能设置从LDROM启动，因此当APROM的代码损坏时用户可以用LDROM启动系统，避免系统启动失败。

数据FLASH是给用户存储数据的，它可以通过ISP读取或存储器读取，并且通过ISP编程。每个擦除单元是512字节。128KB的APROM设备，数据FLASH与APROM共享128KB存储空间。若 Config0 的 DFEN 位使能，数据FLASH基地址由DFBADR定义，大小是（0x20000 - DFBADR），同时APROM大小是(128 KB– 数据FLASH大小).对于64/32 KB APROM的设备，数据FLASH大小固定4KB，起始地址为0x0001_F000。

用户配置提供了几个字节来控制系统逻辑，如FLASH加密，启动选择，掉电检测电压，数据FLASH基地址等。用户配置如同是上电设置的保险，上电时用户配置从FLASH存储器装载到它相应的控制寄存器。
在NuMicro™家族，flash存储器组织跟系统存储映射不同。当用户用ISP命令去读、编程、或者擦除flash，存储器组织被用到。在CPU访问FLASH存储器获取代码或数据的时候，系统存储器映射被用到。例如，当系统设置为从LDROM启动（CBS = 01b）,CPU将从LDROM的0x0 ~ 0x1FFF取代码。但是，如果用户想用ISP读LDROM，仍然要从LDROM的地址0x0010_0000 ~ 0x0010_1FFF去读。

表6-14和图6-34为32/64/128 KB设备的APROM、LDROM、数据Flash和用户配置 的地址映射信息。

块名称	设备类型	大小		开始地址	结束地址
APROM	32 KB	32 KB		0x0000_0000	0x0000_7FFF
	64 KB	64 KB		0x0000_0000	0x0000_FFFF
	128 KB	数据 Flash 使能	128 KB – 数据 Flash 大小	0x0000_0000	0x20000 – 数据 Flash 大小 - 1
		数据 Flash 关闭	128 KB	0x0000_0000	0x0001_FFFF
数据 Flash	32 KB	4 KB		0x0001_F000	0x0001_FFFF
	64 KB	4 KB		0x0001_F000	
	128 KB	数据 Flash 使能	0x20000-DFBADR	DFBADR	
		数据 Flash 禁止	0 KB	N/A	N/A
LDROM	32/64/128 KB	8 KB		0x0010_0000	0x0010_1FFF
用户配置	32/64/128 KB	2 words		0x0030_0000	0x0030_0004

表 5-6 存储器地址映射

Flash存储器组织结构图 5-11所示:

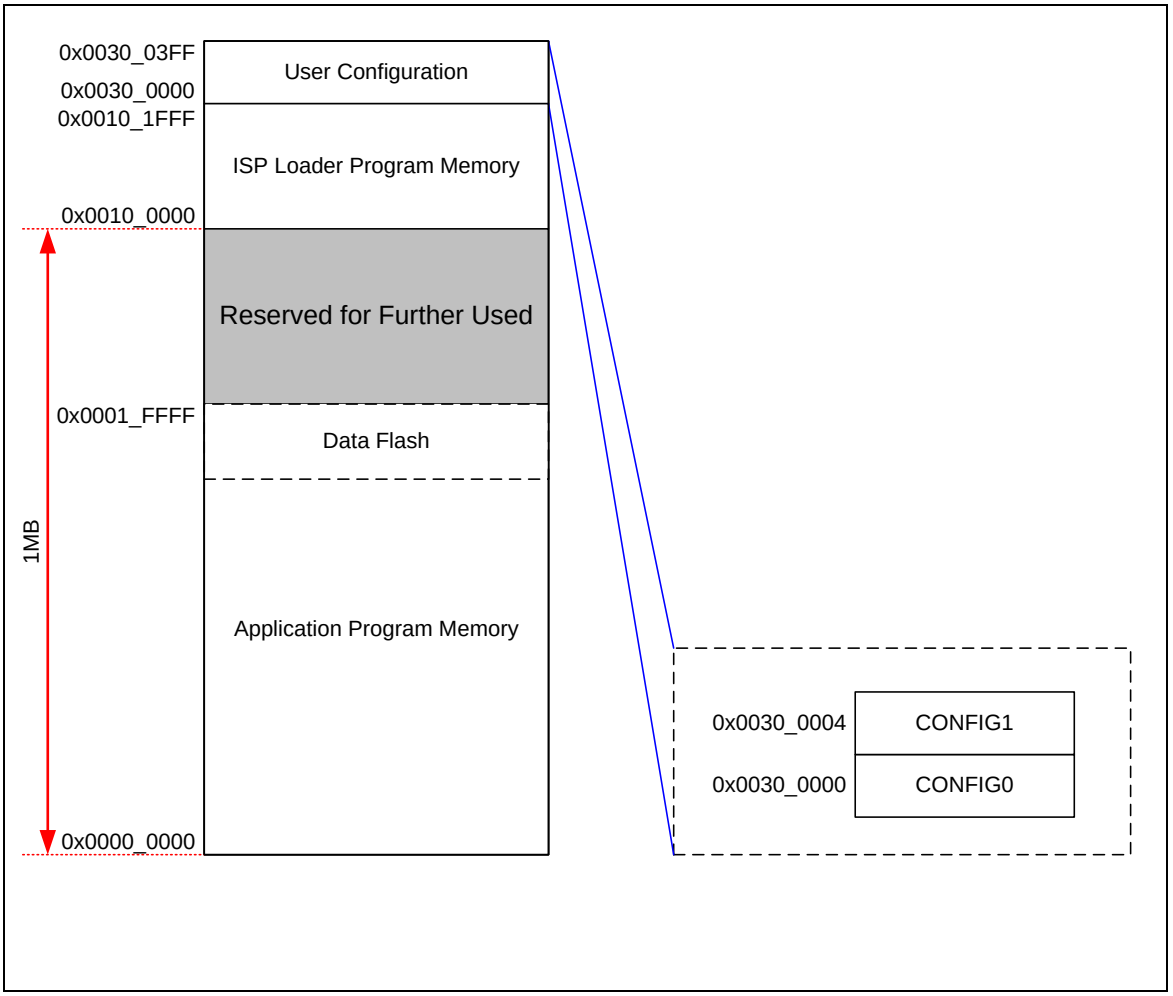


图 5-11 Flash 存储器组织结构

5.4.4.2 *用户配置*

用户配置是内部可编程的配置区域，用于启动选项。用户配置位于Flash存储器组织的0x300000，并有两个32位字节。用户配置所有的更改将在系统重启后生效。

CONFIG0 (地址 = 0x0030_0000)

31	30	29	28	27	26	25	24
CWDTEN	CWDTPDEN	Reserved		CGPFMFP	CFOSC		
23	22	21	20	19	18	17	16
CBODEN	CBOV		CBORST	Reserved			
15	14	13	12	11	10	9	8
Reserved					CIOINI	Reserved	
7	6	5	4	3	2	1	0
CBS		Reserved				LOCK	DFEN

CONFIG0	地址 = 0x0030_0000	
Bit	描述	
[31]	CWDTEN	**看门狗使能** 0 = 芯片上电使能看门狗定时器，时钟源默认为内部10K 1 =芯片上电时默认关闭看门狗定时器.
[30]	CWDTPDEN	**看门狗时钟掉电使能** 0 = 当芯片进入掉电模式时10K看门狗时钟保持使能 1 = 当芯片进入掉电模式时看门狗时钟由OSC10K_EN (PWRCON[3])控制. **注意**: 该位只有在CWDTEN 为0时才起作用
[29:28]	Reserved	保留
[27]	CGPFMFP	**GPF 多功能选择** 0 = XT1_IN 和XT1_OUT脚配置为GPIO功能。 1 = XT1_IN 和XT1_OUT脚用于外部4~24MHz时钟输入 **注意**：XT1_IN, XT1_OUT多功能只由CGPFMFP更改。
[26:24]	CFOSC	**复位后CPU时钟源选择** 000 =外部 4~24 MHz 高速晶振时钟 111 =内部 22.1184 MHz 高速振荡器时钟 其他 =保留 复位后，CFOSC的值将被加载到系统寄存器CLKSEL0.HCLK_S[2:0].
[23]	CBODEN	**欠压检测使能** 0= 上电后，使能欠压检测 1= 上电后，禁用欠压检测

[22:21]	**CBOV**	欠压电压选择 00 = 2.2 V 01 = 2.7 V 10 = 3.7 V 11 = 4.4 V
[20]	**CBORST**	**欠压复位使能** 0 = 上电后，使能欠压复位 1 = 上电后，禁用欠压复位
[19:11]	Reserved	保留
[10]	**CIOINI**	**I/O初始状态选择** 0= 上电后所有GPIO默认为输入高阻模式 1=上电后所有GPIO默认为准双向模式
[9:8]	Reserved	保留
[7:6]	**CBS**	**芯片启动选择** 00=由LDROM启动带IAP功能 01=由LDROM启动不带IAP功能 10=由APROM启动带IAP功能 11=由APROM启动不带IAP功能 IAP功能意味着系统不用重启，CPU就可以执行和访问APROM 、LDROM。当IAP功能被使能时，APROM的基地址是0X0、LDROM基地址是0x100000。
[5:2]	**保留**	保留
[1]	**LOCK**	安全加密 0 = 加密FLASH数据 1 = 解除Flash 数据加密 当FLASH数据被加密，仅有设备ID， Config0 和Config1 可被烧写器和ICP通过串口调试接口读取. 其他数据锁定为0xFFFFFFFF. 无论数据是否锁定，ISP 都可以读取 用户需要用ICP烧录工具擦除整个芯片或者用ISP擦除用户配置来解锁
[0]	**DFEN**	数据FLASH使能(该位仅支持 128KB APROM 器件) 0 = 使能数据FLASH 1 = 禁用数据FLASH

欠压检查功能用来监控V_{DD}脚的电压。如果V_{DD}低于CBOV设置的电压，当BOD有使能时BOD事件将被触发。当BOD被检测到时用户可以决定使能CBORST来BOD复位或者只是由NVIC使能BOD中断。因为无论什么时候V_{DD}电压低于设置的CBOV，BOD复位就会被启用，所以用户必须确定CBOV设置来避免BOD重复复位。例如, $V_{DD=}$3.3V, CBOV只能设置00’b 或 01’b.否则，当BOD被使能CBOV是10’b 或11’b时系统将停留在BOD复位状态。

CONFIG1 (地址 = 0x0030_0004)

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

配置	地址 = 0x0030_0004	
位	描述	
[31:20]	Reserved	保留 (强行给这些保留位写入0x00)
[19:0]	**DFBADR**	数据FLASH的基地址（仅支持 128KB APROM 器件） 128KB APROM 器件的数据FLASH基地址由用户定义，因为片上FLASH擦除单位为512字节，所以强制bit 8-0位为0。这个配置只对128KB APROM 器件有效。

5.4.4.3 *启动选择*

NuMicro™ NUC200系列提供在系统编程(ISP)功能，允许用户通过单机ISP固件更新程序存储器. 提供8kB加载程序内存用于存储ISP固件. 用户可以通过设置Config0中的CBS[1]位来选择从APROM还是从LDROM取指令来运行.

Config0中的CBS除了设置从APROM或者LDROM启动之外还用于启动后控制系统存储器映射。当CBS[0] = 1 和 CBS[1] = 1为从APROM启动， APROM的应用程序不能通过读储存器的方式来访问LDROM.同样当CBS[0] = 1 和 CBS[1] = 0为LDROM启动，LDROM的程序也不能通过读储存器的方式来访问APROM。图6-35 表示当系统从APROM和LDROM启动时的储存器映射。

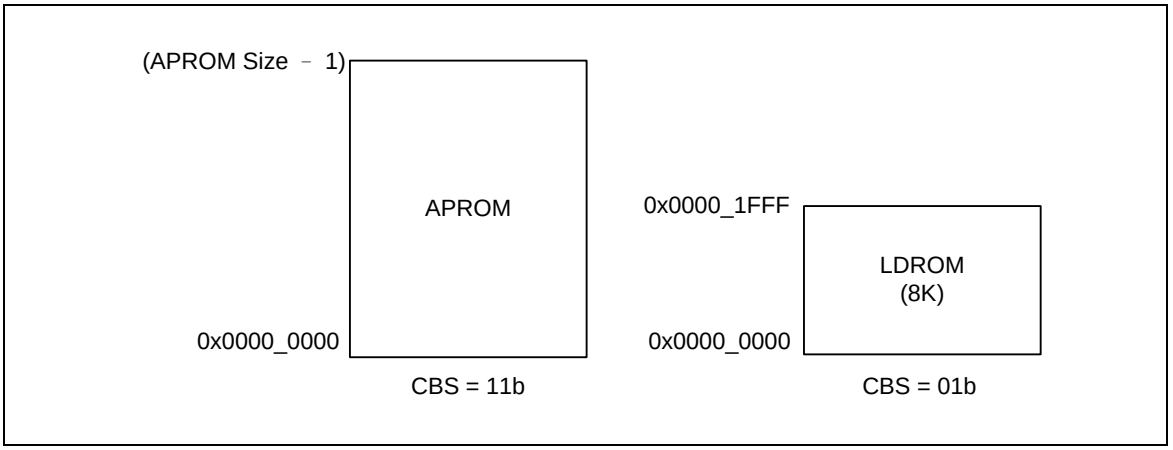


图 5-12从APROM和LDROM启动的程序执行范围

对于应用程序，软件需要在APROM执行代码，并且调用LDROM的功能或在LDROM执行代码，调用APROM功能而不改变启动模式，CBS[0]需要设置为0，这叫做在应用编程(IAP)。

5.4.4.4 *在应用编程（IAP）*

NuMicro NUC200系列提供了应用编程(IAP)功能,用户无需复位系统就可以在APROM和LDROM之间切换执行代码。用户可以通过设置CONFIG0 (CBS[1:0]) 为 10b 或 00b并重启来使能IAP功能。

在芯片从APROM启动使能IAP功能(CBS[1:0] = 10b)的情况下，代码的执行地址范围包括所有的APROM和LDROM。APROM 地址空间保持不变，但8 KB LDROM被映射到0x0010_0000~ 0x0010_1FFF。

在芯片从LDROM启动使能IAP功能(CBS[1:0] = 00b)的情况下，代码的执行地址范围包括所有的LDROM，和除了第一页的APROM。执行代码不能访问APROM的第一页，因为第一页的执行地址默认变成LDROM第一页的镜像。同时8 KB LDROM也被映射到0x0010_0000~ 0x0010_1FFF。

当IAP使能时，地址空间映射请参考**錯誤! 找不到參照來源。**

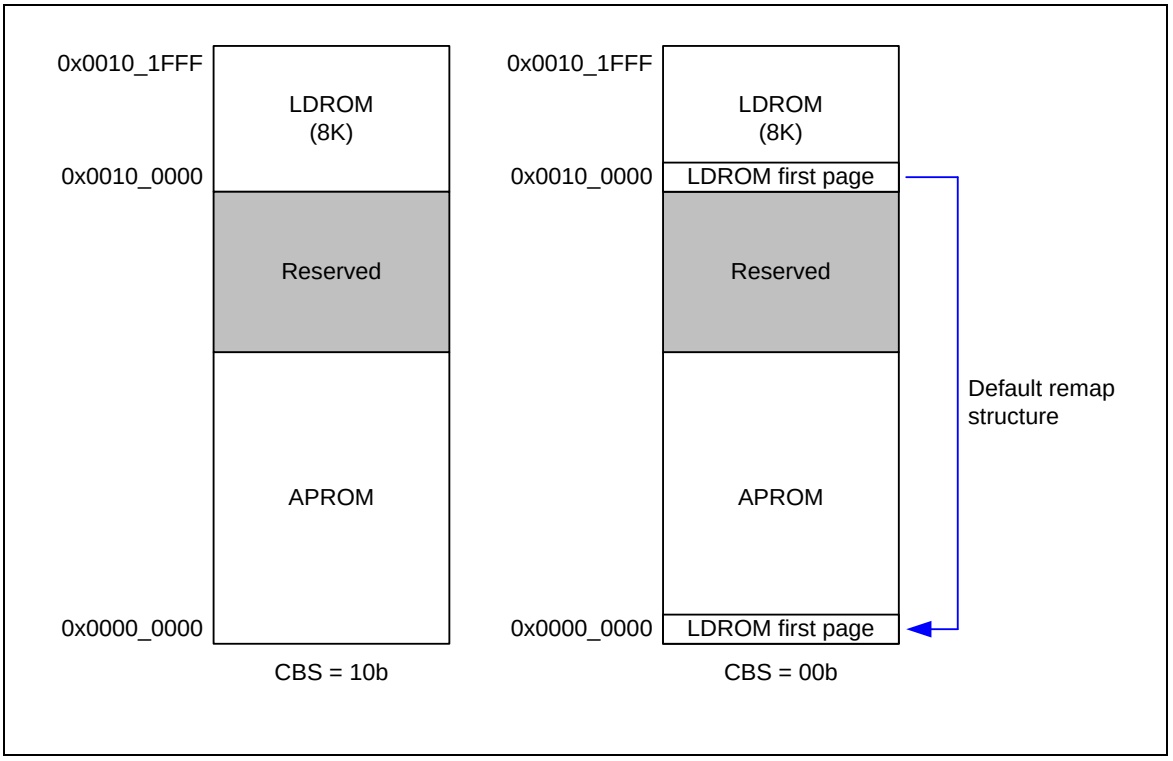


图5-13 IAP使能时代码的执行范围

当芯片使能了IAP功能，任何时候代码可执行范围内的任何其他页都可被镜像到执行代码的第一页(0x0000_0000~0x0000_01FF)。用户可以通过填目标重映射地址到ISPADR改变第一可执行页的重新映射地址，通过读ISPSTA 寄存器中VECMAP域，用户可以检查改变是否成功。

5.4.4.5 *在系统编程(ISP)*

NuMicro™ NUC200系列支持ISP模式，当烧写失败时设备可以在软件控制重新烧写，避免系统崩溃的风险。因此，更新应用固件的功能,可以让应用更为广泛。

ISP可以更新板上的系统固件，各种外设接口使得LDROM更新的固件更容易。执行ISP最常用的方法是在LDROM中的固件通过UART更新，PC一般都是通过串口传输新的APROM代码. ISP下载程序接收后，通过ISP命令，重新对APROM编程。

5.4.4.6 *ISP过程*

NuMicro™ NUC200系列支持用户定义从APROM 或LDROM启动。更改用户配置后重启系统生效。如果用户想不更改用户配置来切换APROM或LDROM模式那么必须控制ISPCON控制寄存器的BS

位。然后设置IPRSTC1控制寄存器复位CPU。通过BS位来切换启动流程如下图：

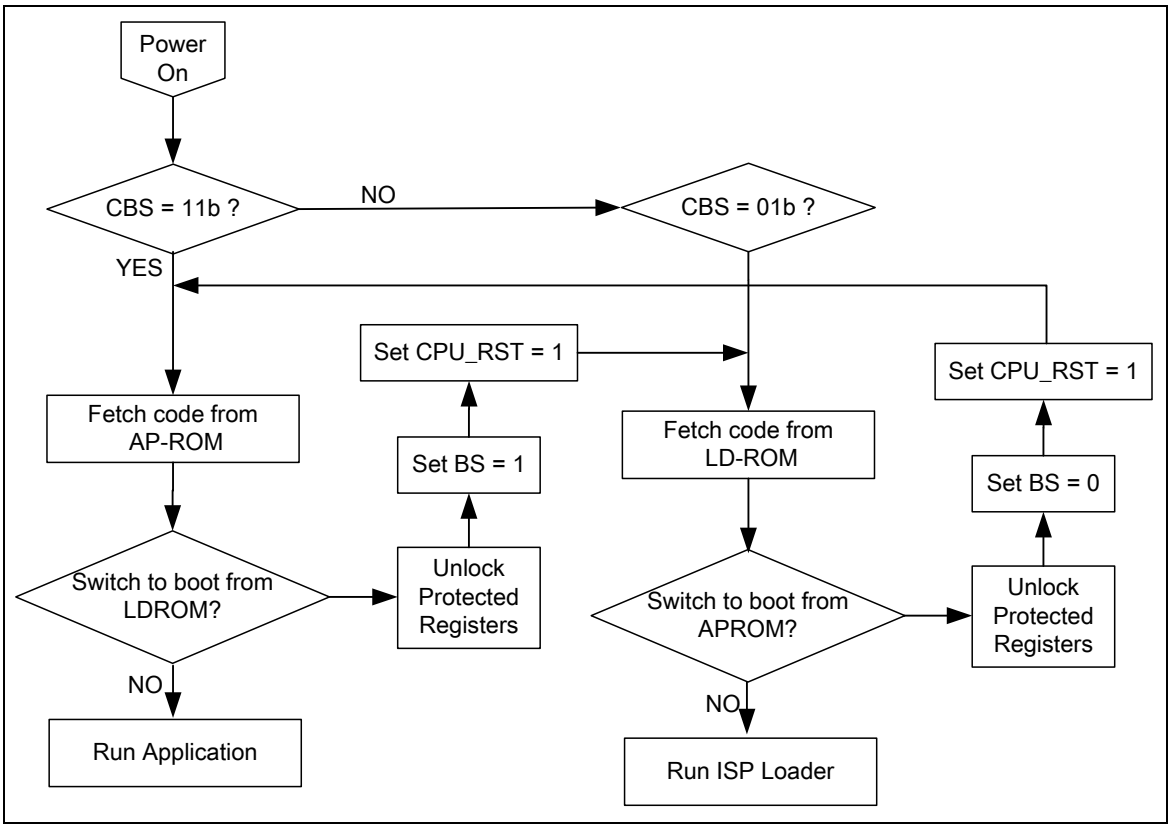


图 5-14 BS 位启动选择流程示例

通过LDROM软件更新APROM或者通过APROM软件更新LDROM可以避免在更新失败时系统崩溃。

ISP控制器支持读、写、擦除内部FLASH存储器。ISP控制器的几个控制位是被写保护的，因此设置前需要解锁。解锁保护寄存器位软件需要依次写0x59, 0x16 和 0x88到REGWRPROT。如果解锁成功REGWRPROT值将被置1.解锁过程不能被其他访问中断，否则会解锁失败。

解锁保护寄存器位后，用户需要设置ISPCON控制寄存器来决定更新LDROM、用户配置区、APROM和使能ISP控制器。

一旦ISPCON寄存器被设置成功，用户可以通过设置ISPCMD来擦除、读或者写。基于flash内存的组织结构，目标flash内存来设置ISPADR。ISPDAT可以用于设置数据写入或者返回 ISPCMD读的数据。

最后设置ISPTRG控制寄存器的ISPGO位来执行相应的ISP功能。当ISP功能完成后ISPGO位将自动清0。为了确保在CPU向前运行之前ISP功能已经完成，在ISPGO设置之后，应该用ISB指令。

ISP完成后几个错误条件需要检查。如果出现错误，ISP操作就不会开始并且ISP失败标志被置位。ISPFF标志只能由软件清除。当ISPFF位保持1下一个ISP程序仍然可以开始，因此，建议在ISP操作完成后检查ISPFF位，如果被置1要清0。

当ISPGO位被置1，CPU 将一直等到ISP操作到完成，这个时期内外围设备跟通常一样保持运行。任何中断请求都不会响应直到CPU完成ISP操作。当ISP操作完成ISPGO位将被硬件自动清0。用户可以通过ISPGO位来检查ISP操作是否完成。用户应该在ISPGO置1之后加ISB指令，确保ISP操作之后的指令正确执行。

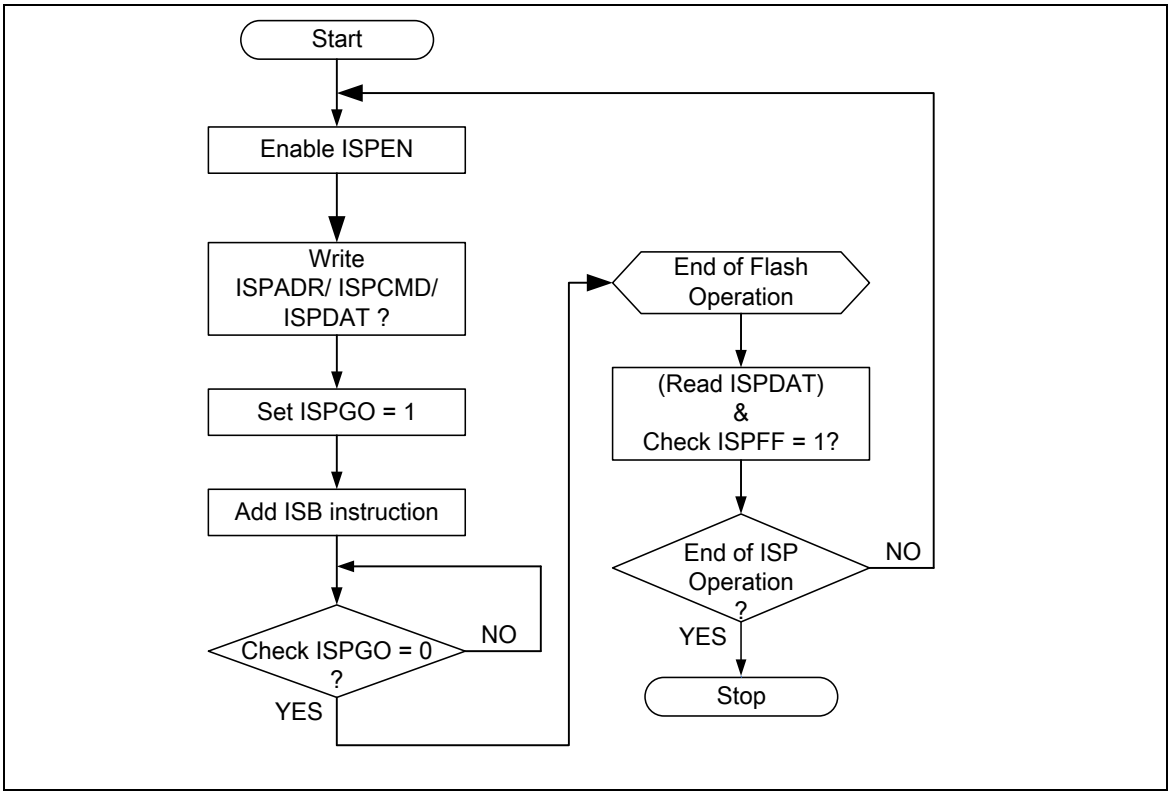


图 5-15 ISP 流程示例

ISP命令	ISPCMD	ISPADR	ISPDAT
FLASH页擦除	0x22	FLASH存储器结构的有效地址 它必须由512字节/页排列	N/A
FLASH 写	0x21	FLASH存储器结构的有效地址	写数据
FLASH 读	0x00	FLASH存储器结构的有效地址	返回数据
读UID	0x04	0x0000_0000	UID字0
		0x0000_0004	UID字1
		0x0000_0008	UID字2
向量页重映射	0x2E	APROM 或 LDROM 的页 它必须由512字节/页排列	N/A

表5-7 ISP 命令列表

5.4.5 寄存器映射

R:只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
FMC基地址: **FMC_BA = 0x5000_C000**				
ISPCON	FMC_BA+0x00	R/W	ISP 控制寄存器	0x0000_0000
ISPADR	FMC_BA+0x04	R/W	ISP 地址寄存器	0x0000_0000
ISPDAT	FMC_BA+0x08	R/W	ISP 数据寄存器	0x0000_0000
ISPCMD	FMC_BA+0x0C	R/W	ISP 命令寄存器	0x0000_0000
ISPTRG	FMC_BA+0x10	R/W	ISP 触发寄存器	0x0000_0000
DFBADR	FMC_BA+0x14	R	数据Flash起始地址	0x000X_XXXX
FATCON	FMC_BA+0x18	R/W	Flash访问时间控制寄存器	0x0000_0000
ISPSTA	FMC_BA+0x40	R/W	ISP 状态寄存器	0x0000_0000

5.4.6 寄存器描述

ISP 控制寄存器 (ISPCON)

寄存器	偏移地址	R/W	描述	复位值
ISPCON	FMC_BA+0x00	R/W	ISP 控制寄存器	0x0000_0000

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

位	描述	
[31:7]	保留	保留.
[6]	ISPFF	**ISP失败标志 (写保护位)** ISP失败标志 (写保护位) 当ISP满足下列条件时，该位由硬件置位: (1) APUEN等于0时，APROM 写APROM. (2) LDUEN等于0时，LDROM 写LDROM. (3) CFGUEN等于0时，CONFIG 被擦除或编程. (4) 定义地址无效，如超过正常范围. 该位写 1 清除
[5]	LDUEN	**LDROM更新使能 位(写保护位)** 0 = 禁止LDROM更新 1 =当芯片在 APROM 中运行时，LDROM 可以更新。
[4]	CFGUEN	**使能由 ISP 更新配置位 (写保护位)** 0 = 禁止ISP更新配置位 1 = 使能ISP更新配置位
[3]	APUEN	APROM 更新使能 (写保护位) 0 = 当芯片在APROM中运行时APROM 不能被更新. 1 = 当芯片在APROM中运行时APROM 可以被更新.
[2]	保留	保留

[1]	**BS**	**启动选择 (写保护位)** 置位/清零该位选择下次是由LDROM启动还是由APROM启动，该位也可作为MCU启动状态的标志，用于检查芯片是由LDROM还是APROM启动的. 该位在复位时（除了CPU 复位(RSTS_CPU 为 1) 或系统复位 (RSTS_SYS)）被初始化为 Config0 的 CBS位的反转值，在其他复位时保持不变. 1 = 由LDROM启动 0 = 由APROM启动
[0]	**ISPEN**	**ISP 使能(写保护位)** ISP 使能位，设置该位可以使能ISP功能. 1 = 使能 ISP 功能 0 = 禁用 ISP 功能

ISP 地址寄存器(ISPADR)

寄存器	偏移地址	R/W	描述	复位值
ISPADR	FMC_BA+0x04	R/W	ISP地址寄存器	0x0000_0000

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

位	描述	
[31:0]	**ISPADR**	**ISP 地址** NuMicro™ NUC200系列内嵌最大FLASH32Kx32 (128 KB)，只支持字编程。执行ISP功能时，ISPARD[1:0] 必须为00b.

ISP 数据寄存器 (ISPDAT)

寄存器	偏移地址	R/W	描述	复位值
ISPDAT	FMC_BA+0x08	R/W	ISP 数据寄存器	0x0000_0000

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

位	描述	
[31:0]	**ISPDAT**	**ISP 数据** ISP写操作之前，写数据到该寄存器 ISP读操作后，可从该寄存器读数据

ISP 命令寄存器(ISPCMD)

寄存器	偏移地址	R/W	描述	复位值
ISPCMD	FMC_BA+0x0C	R/W	ISP命令寄存器	0x0000_0000

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

位	描述	
[31:6]	**保留**	保留
[5:0]	**ISPCMD**	**ISP命令** ISP 命令表如下: 0x00 = 读 0x04 =读UID 0x0B = 读公司ID (0xDA). 0x21 = 写. 0x22 = 页擦除. 0x2E =向量页重映射.

ISP触发控制寄存器(ISPTRG)

寄存器	偏移地址	R/W	描述	复位值
ISPTRG	FMC_BA+0x10	R/W	ISP触发控制寄存器	0x0000_0000

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

位	描述	
[31:1]	**Reserved**	保留.
[0]	**ISPGO**	**ISP开始触发（写保护）** 写 1 开始ISP操作，当ISP操作结束后，该位由硬件自动清零. 0 = ISP 操作结束 1 = ISP 正在执行 这是写保护的位，写这些被保护的位需要依次向地址0x5000_0100写入“59h”，“16h”，“88h”，禁用寄存器保护。参考寄存器REGWRPROT，地址GCR_BA+0x100。

数据FLASH基地址寄存器(DFBADR)

寄存器	偏移地址	R/W	描述	复位值
DFBADR	FMC_BA+0x14	R	数据 FLASH 基地址	0x000X_XXXX

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

位	描述	
[31:0]	**DFBADR**	**数据FLASH基地址** 该寄存器为数据FLASH开始地址寄存器，只读. 对于128KB FLASH存储器器件，数据Flash的大小由用户配置定义。芯片上电后该寄存器的默认值从Config1加载。对于64/32 KB的器件，地址固定为0x0001_F000

Flash访问时间控制寄存器 (FATCON)

寄存器	偏移地址	R/W	描述	复位值
FATCON	FMC_BA+0x18	R/W	Flash访问时间控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
保留							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
保留							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
保留	FOMSEL1	保留	FOMSEL0	保留			

位	描述	
[31:7]	保留	保留.
[6]	**FOMSEL1**	**芯片频率优化模式选择1（写保护位）**
[5]	保留	保留.
[4]	**FOMSEL0**	**芯片频率优化模式选择0（写保护位）** 当芯工作频率低于 72 MHz时，通过设置FOMSEL1 和 FOMSEL0 芯片可以更高效的工作。 00=CPU运行在50MHz零等待周期连续地址读访问。 01=CPU运行在25MHz零等待周期随机地址读访问。 10=CPU运行在50MHz零等待周期连续地址读访问。 11=CPU运行在72MHz零等待周期连续地址读访问。 其中00表示FOMSEL1 = 0, FOMSEL0 = 0; 01 表示 FOMSEL1 = 0, FOMSEL0 = 1，等等
[3:0]	保留	保留.

ISP状态寄存器(ISPSTA)

寄存器	偏移地址	R/W	描述	复位值
ISPSTA	FMC_BA+0x40	R/W	ISP状态寄存器	0x0000_0000

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

位	描述	
[31:21]	Reserved	**保留**.
[20:9]	**VECMAP**	**向量页映射地址（写保护）** 当前 flash 地址空间 0x0000_0000~0x0000_01FF 映射到地址 {VECMAP[11:0], 9'h000} ~ {VECMAP[11:0], 9'h1FF}
[8:7]	Reserved	**保留**.
[6]	**ISPFF**	**ISP失败标志（写保护位）** 当 ISP 满足下列条件时，该位由硬件置位： (1) APROM 写入本身 (2) LDROM 写入本身 (3) 如果 CFGUEN 设置为 0，CONFIG 被擦除/编程 (4) 目的地址无效，比如超过正常范围 写 1 清除该位。 **注意**：该位的功能同ISPCON bit6.
[5:3]	Reserved	**保留**.
[2:1]	**CBS**	**启动选择状态（只读）** 这是CBS在CONFIG0中的镜像
[0]	**ISPGO**	**ISP 开始触发（只读）** 写 1 开始 ISP 操作，当 ISP 操作结束后，该位由硬件自动清零。 0 = ISP 操作结束 1 = ISP 正在执行 **注意**：这位同ISPTRG bit0

5.5 外部总线接口 (EBI)

5.5.1 简介

NuMicro™ NUC200系列的LQFP-64和LQFP-100封装配备了一个外部总线接口（EBI），以供访问外部设备使用。

为节省外部设备和芯片的连接线，EBI支持地址总线与数据总线的多路复用，地址锁存使能(ALE)信号是用来区分地址与数据周期的。

5.5.2 特性

外部总线接口有如下功能：

- 支持外部设备最大空间为64KB（8-位数据宽度）/128KB（16-位数据宽度）
- 支持基于HCLK所产生的可调外部总线基本时钟（MCLK）
- 支持8-位或16-位数据宽度
- 支持可变的数据访问时间（tACC）、地址锁存使能时间（tALE）和地址保持时间（tAHD）
- 支持地址总线和数据总线多路复用以节省地址管脚
- 支持可配置的空闲周期以用于不同的访问条件：写命令完成（W2X），连续读（R2R）

5.5.3 框图

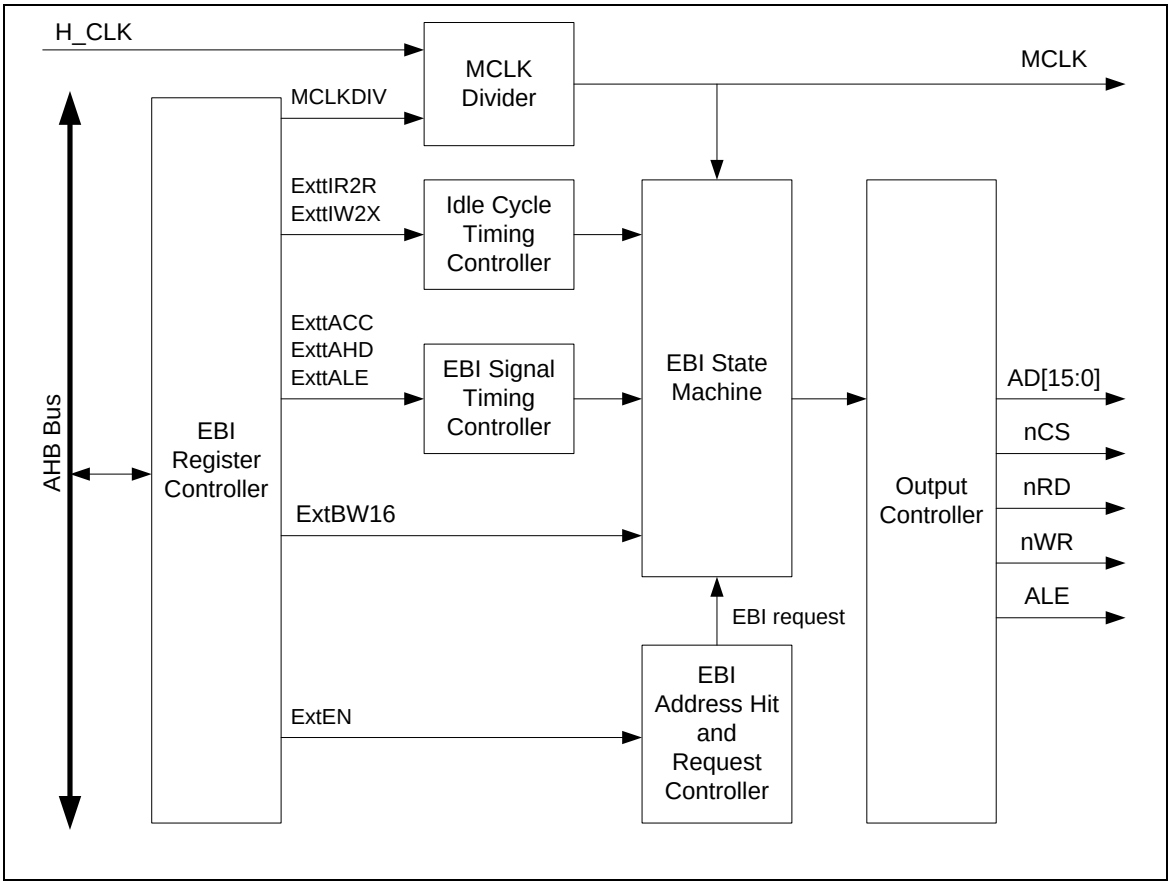


图 5-16 EBI 框图

5.5.4 功能说明

5.5.4.1 *EBI 区域和地址*

EBI 映射地址分布在0x6000_0000 ~ 0x6001_FFFF，最大可用存储空间为128KB。当系统请求地址在EBI的存储空间内时，相应的EBI芯片选择信号（nCS）有效，EBI状态机工作。

对于一个8-位设备（64KB），EBI将同时映射这64KB设备到0x6000_0000 ~ 0x6000_FFFF和0x6001_0000 ~ 0x6001_FFFF。

对于一个16-位设备（128KB），EBI映射这128KB设备到0x6000_0000 ~ 0x6001_FFFF。

5.5.4.2 *EBI 数据宽度连接*

EBI控制器支持连接具有多路复用的地址总线和数据总线的外部设备。对于地址总线和数据总线分开的外部设备，与设备的连接需要额外的逻辑（锁存器）锁存地址。在这种情况下，管脚ALE需要连到锁存器上的锁存地址值。16-位数据宽度管脚AD0~AD15 / 8-位数据宽度管脚AD0~AD7 都是锁存器的输入管脚，锁存器的输出管脚连接到外部设备的 Addr[15:0]。

对于16-位设备，AD [15:0]由地址（Addr [15:0]）和16-位数据（Data [15:0]）共用。对于8-位设备，只有AD [7:0]由地址（Addr [7:0]）和8-位数据（Data [7:0]）共用，AD [15:8]作为专用地址（Addr [15:8]），可以直接与8-位设备连接。

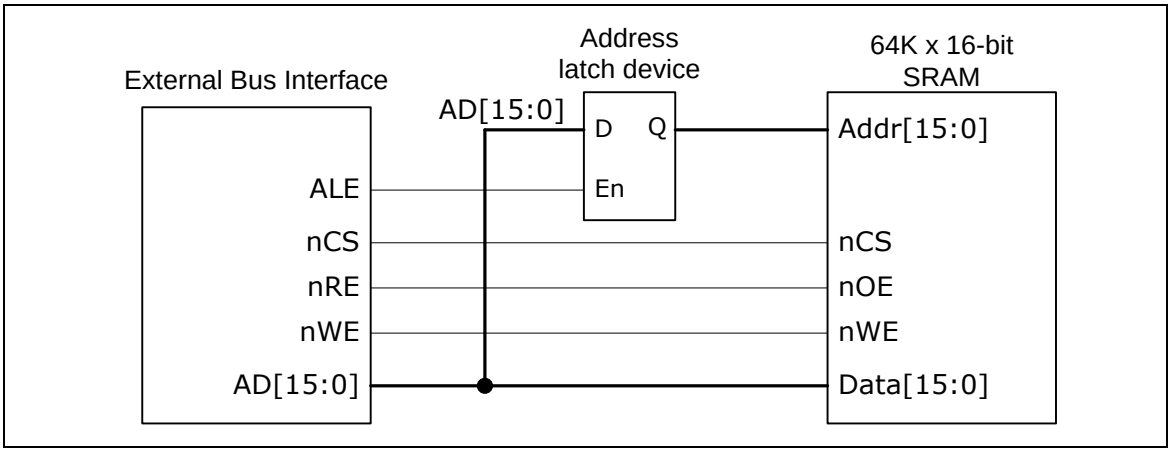


图 5-17 16-位数据宽度与16-位设备的连接

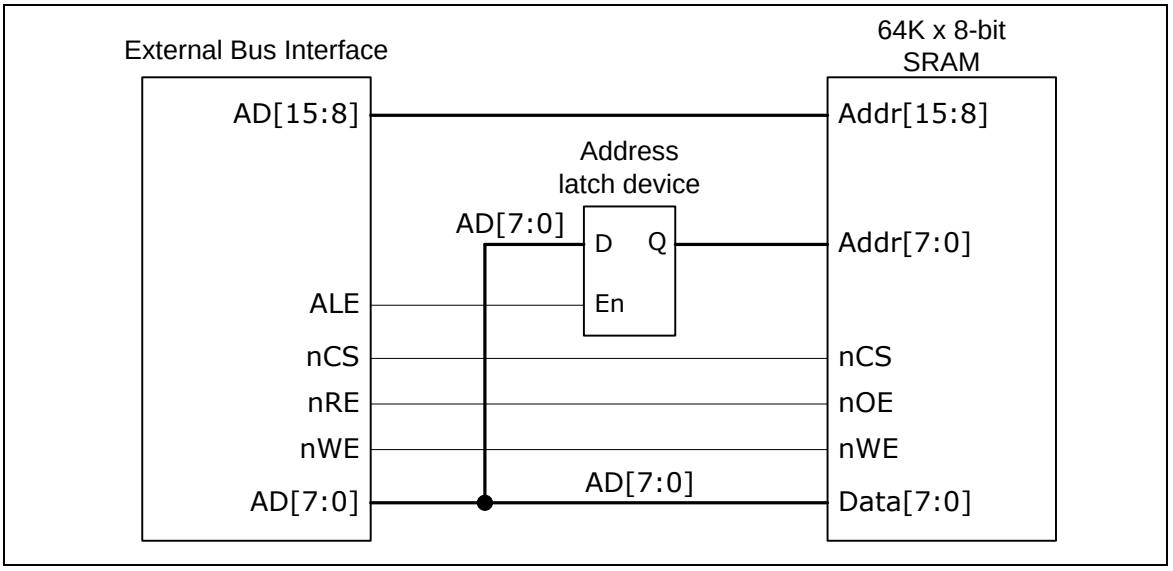


图 5-18 8-位数据宽度与8-位设备的连接

当系统访问数据宽度大于EBI的数据宽度（8-位/16-位数据宽度），EBI控制器将会执行一次以上的EBI访问操作来完成系统访问命令。例如，如果系统通过EBI设备请求32-位数据，当EBI为8-位数据宽度时，EBI控制器将访问4次完成操作。

5.5.4.3 *EBI 操作控制*

MCLK 控制

在芯片内部，当EBI工作时，所有EBI信号通过MCLK进行同步。当芯片以较低工作频率连接到外部设备，MCLK可以通过设定MCLKDIV[2:0]（EBICON [10:8] 外部输出时钟分频器）最多将HCLK进行32分频。因此，EBI控制器可以适用于宽频率范围的EBI设备。如果MCLK频率设置为HCLK/1，EBI信号与MCLK的正边沿同步，否则与MCLK的负边沿同步。

操作与访问时间控制

开始EBI访问时，片选信号（nCS）置低并且等待一个MCLK的时间用于地址设置时间（tASU）以使地址稳定。然后在地址稳定后，ALE信号置高并保持一段时间（tALE）以锁存地址。在锁存地址后，ALE信号置低并保持一个MCLK时间以便地址锁存保持时间（tLHD）。在另一个插入到地址锁存保持时间之后的一个MCLK周期（tA2D）是用于总线上地址到数据的转换时间。然后当读访问时nRD信号置低或者写访问时nWR信号置低。在保持访问时间（tACC）后，nRD或nWR信号置高用于读输出稳定或写完成。之后，EBI信号保持数据访问时间（tAHD）和片选信号（nCS）置高，地址由当前访问控制释放。

EBI控制器提供灵活的时序控制以用于不同外部设备。EBI的时序控制，tASU, tLHD 和tA2D固定为1个MCLK周期。tAHD可以在1~8个MCLK周期调节，通过设定ExttAHD[2:0]（EXTIME [10:8] EBI数据访问保持时间），tACC可以在1~32个MCLK周期调节，通过设定ExttACC[4:0]（EXTIME [7:3] EBI数据访问时间），tALE可以在1~8个MCLK周期调节，通过设定ExttALE [2:0]（EBICON [18:16]ALE扩展时间）。

参数	值	单元	描述
tASU	1	MCLK	地址锁存建立时间
tALE	1 ~ 8	MCLK	ALE 高时期.由寄存器EBICON[18:16]位上的ExttALE[2:0]进行控制
tLHD	1	MCLK	地址锁存保持时间
tA2D	1	MCLK	地址到数据延时（总线转换时间）
tACC	1 ~ 32	MCLK	数据访问时间。由寄存器EXTIME[7:3]位上的ExttACC[4:0]进行控制
tAHD	1 ~ 8	MCLK	数据访问保持时间。由寄存器EXTIME[10:8]位上的ExttAHD[2:0]控制
IDLE	0 ~ 15	MCLK	空闲周期。由寄存器EXTIME[27:24]位上的ExtIR2R[3:0]和寄存器EXTIME[15:12]位上的ExtIW2X[3:0]来进行控制。

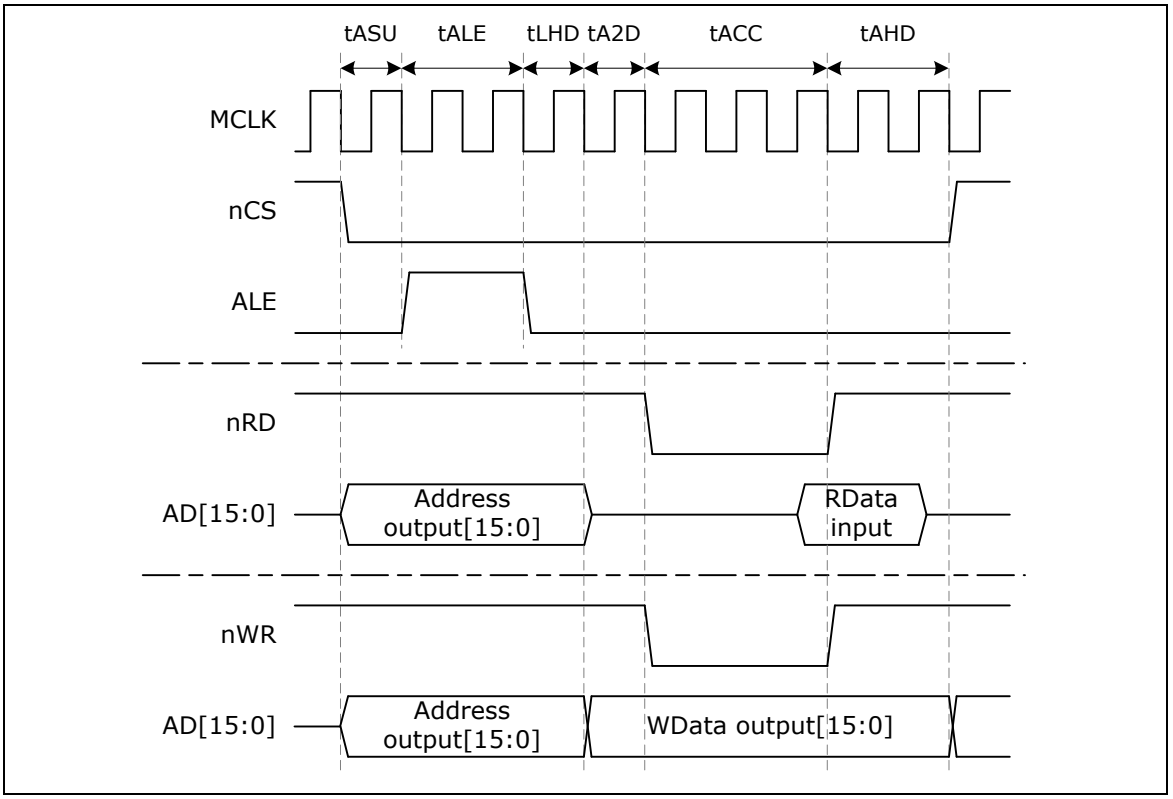


图 5-19 16-位数据宽度的时序控制波形

上图是一个设置16-位数据宽度的EBI应用示例。在该示例中，AD0~AD15被用作地址[15:0]和数据[15:0]。当ALE信号置高，AD0~AD15为地址输出。在地址锁存之后（tLHD），ALE信号置低，并且在读取访问操作时，AD总线转换成高阻态以等待设备输出数据，或用于写数据输出。

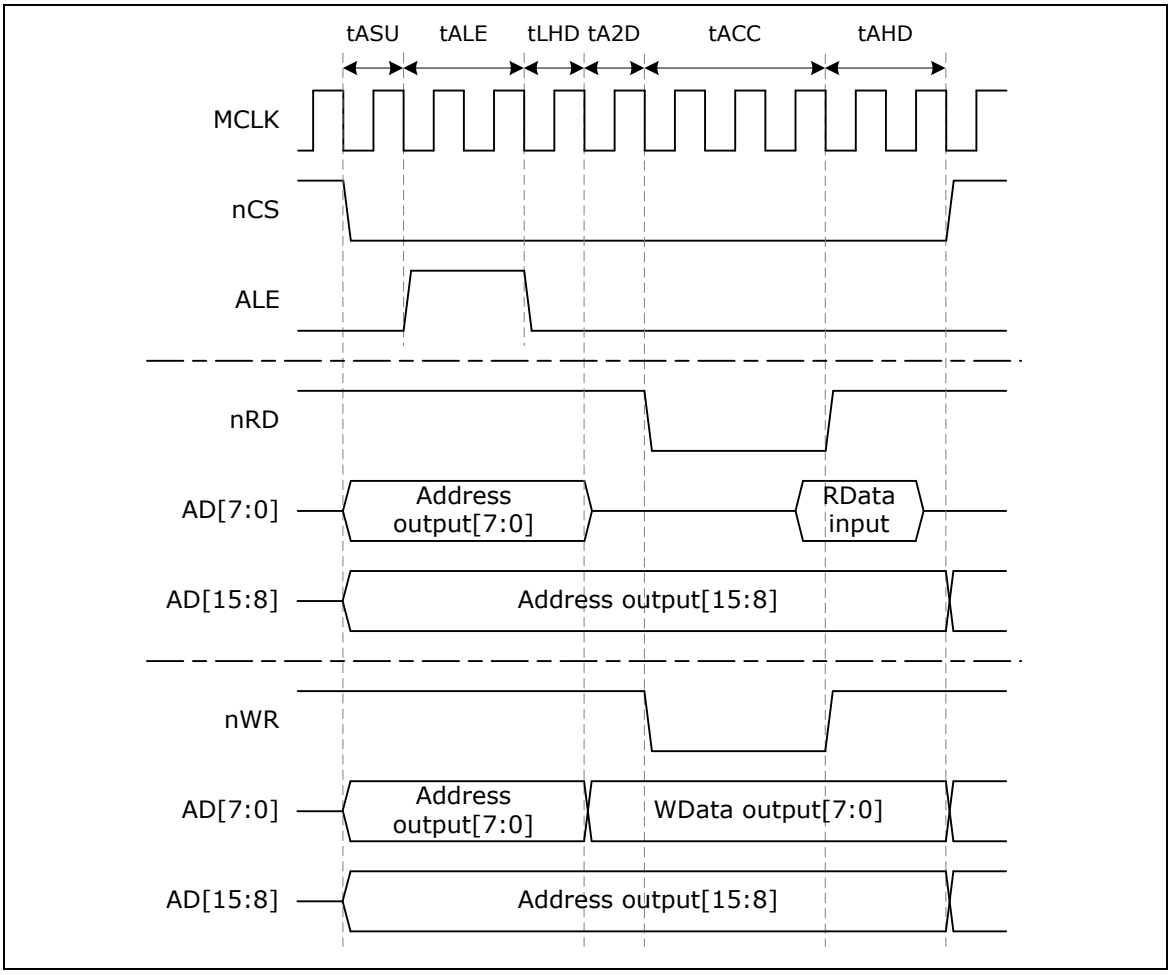


图 5-20 8-位数据宽度的时序控制波形

上图是一个设置8-位数据宽度的EBI应用示例。8-位和16-位数据宽度的不同之处在于AD8 ~ AD15。在8-位数据宽度的设置中，AD8~AD15总为地址[15:8]输出，因此外部锁存只需要8-位宽度。

插入空闲周期

当EBI连续访问时，如果器件访问时间比系统时钟频率还要长，可能会产生总线冲突。EBI控制器支持额外的空闲周期来解决这个问题。在空闲周期内，所有EBI总线上的控制信号都无效。空闲周期如下图所示：

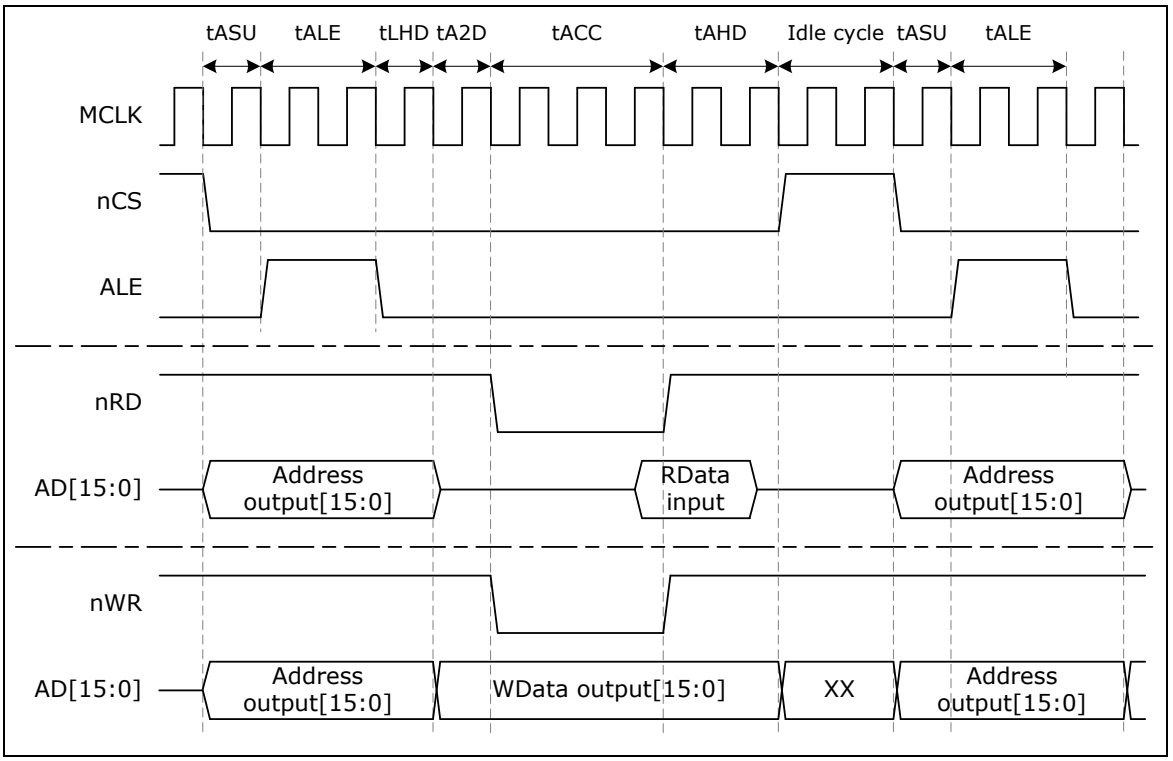


图 5-21 插入空闲周期的时序控制波形图

以下两个条件，EBI可以通过时序控制插入空闲周期：

1. 写访问之后
2. 读访问之后与下一个读访问之前

通过设置寄存器EXTIME [15:12]位上的ExtIW2X[3:0]和寄存器EXTIME[27:24]位上的ExtIR2R[3:0]，空闲周期可以设定在 0~15 个MCLK。

5.5.5 寄存器映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移地址	R/W	描述	复位值
EBI Base Address: **EBI_BA = 0x5001_0000**				
EBICON	EBI_BA+0x00	R/W	外部总线接口通用控制寄存器	0x0000_0000
EXTIME	EBI_BA+0x04	R/W	外部总线接口时序控制寄存器	0x0000_0000
EBICON2	EBI_BA+0x08	R/W	外部总线接口通用控制寄存器2	0x0000_0000

5.5.6 寄存器描述

5.5.6.1.1 外部总线接口通用控制寄存器 (EBICON)

寄存器	偏移地址	R/W	描述	复位值
EBICON	EBI_BA+0x00	R/W	外部总线接口通用控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**ExttALE**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**MCLKDIV**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**ExtBW16**	**ExtEN**

位	描述	
[31:19]	**Reserved**	保留
[18:16]	**ExttALE**	**ALE扩展时间** 该区域被用来控制ALE脉宽（tALE）来锁存地址　tALE = (ExttALE+1)*MCLK
[15:11]	**Reserved**	保留
[10:8]	**MCLKDIV**	**外部输出时钟分频器** EBI频率输出时钟（MCLK）由MCLKDIV控制，如下表： **M LKDIV** / **输出时钟 (MCLK)** 000 / HCLK/1 001 / HCLK/2

		010 HCLK 4 011 HCLK/8 100 HCLK/16 101 HCLK/32 其它 默认 **备注:** 输出时钟的默认值为 HCLK/1
[7:2]	**Reserved**	保留
[1]	**ExtBW16**	**EBI 数据宽度16-位/8-位** 该位定义数据总线是8-位还是16-位 1 = EBI 数据宽度是 16-位 0 = EBI 数据宽度是 8-位
[0]	**ExtEN**	**EBI 使能位** 该位是 EBI的功能使能位 1 = 使能 EBI 功能 0 = 禁用 EBI 功能

5.5.6.1.2 外部总线接口时序控制寄存器 (EXTIME)

寄存器	偏移地址	R/W	描述	复位值
EXTIME	EBI_BA+0x04	R/W	外部总线接口时序控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**ExtIR2R**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ExtIW2X				**Reserved**	**ExttAHD**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ExttACC				**Reserved**			

位	描述	
[31:28]	**Reserved**	保留
[27:24]	**ExtIR2R**	**读-读之间空闲状态周期** 当读动作完成，下一个动作是将要读时，空闲状态被插入，如果ExtIR2R不为零时，nCS信号返回高。 空闲状态周期 = (ExtIR2R*MCLK)
[23:16]	**Reserved**	保留
[15:12]	**ExtIW2X**	**写之后的空闲状态周期** 当写动作完成时，空闲状态被插入，如果 ExtIW2X 不为零时，nCS信号返回高。 空闲状态周期= (ExtIW2X*MCLK)
[11]	**Reserved**	保留
[10:8]	**ExttAHD**	**EBI 数据访问保持时间** ExttAHD 定义数据访问保持时间（tAHD）。 tAHD = (ExttAHD +1) * MCLK
[7:3]	**ExttACC**	**EBI 数据访问时间** ExttACC 定义数据访问时间（tACC）。 tACC = (ExttACC +1) * MCLK
[2:0]	**Reserved**	保留

5.5.6.1.3 外部总线接口控制寄存器 2 (EBICON2)

寄存器	偏移地址	R/W	描述	复位值
EBICON2	EBI_CTL_BA+0x08	R/W	外部总线接口控制寄存器2	0x0000_0000

31	30	29	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					WAHD_OFF	RAHD_OFF	WBUFF_EN

位	描述	
[31:3]	**Reserved**	保留
[2]	**WAHD_OFF**	**当写时，访问保持时间禁止** 0 = 在EBI写期间，tAHD 由 ExttAHD 控制。 1 = 在EBI写期间，没有 tAHD。
[1]	**RAHD_OFF**	**当读时，访问保持时间禁止** 0 = 在EBI读期间，tAHD 由 ExttAHD 控制。 1 =在EBI读期间，没有 tAHD。
[0]	**WBUFF_EN**	**EBI 写缓存使能位** 0 = 禁用 EBI 写缓存。 1 = 使能 EBI 写缓存。

5.6 通用 I/O(GPIO)

5.6.1 概述

NuMicro™ NUC200 系列多达84个通用I/O管脚和其他功能管脚共享，这取决于芯片的配置。84个管脚分配在GPIOA, GPIOB, GPIOC, GPIOD, GPIOE与GPIOF六个端口上。GPIOA/B/C/D/E最多有16个管脚，GPIOF最多4个管脚。每个管脚都是独立的，都有相应的寄存器位来控制管脚功能模式与数据。

I/O管脚的I/O类型可由软件独立地配置为输入，输出，开漏或准双向模式。复位之后，所有管脚的 I/O 类型取决于Config0[10]的设置。在准双向模式中，I/O管脚有一个阻值为110K~300K的弱上拉电阻接到V_{DD} 上，V_{DD}范围从5.0 V 到2.5 V。

5.6.2 特征

- 四种 I/O 模式:
  - 准双向模式
  - 推挽输出
  - 开漏输出
  - 高阻态输入
- 通过GPx_MFP[31:16]中的Px_TYPE[15:0]，可选TTL/Schmitt 触发输入。
- I/O可以配置为边沿/电平触发的中断源
- 通过Config0[10] 可配置所有I/O复位之后的默认模式。
  - 如果 Config[10] 是 0, 复位后所有的GPIO管脚是三态（高阻）模式
  - 如果 Config[10] 是 1, 复位后所有的GPIO管脚是准双向模式
- I/O脚仅在准双向模式，内部上拉电阻才使能。
- 使能管脚中断功能将也使能了唤醒功能。

5.6.3 基本配置

GPIO管脚通过配置GPA_MFP, GPB_MFP, GPC_MFP, GPD_MFP, GPE_MFP, ALT_MFP, ALT_MFP1 和 ALT_MFP2寄存器来实现功能。

5.6.4 功能描述

5.6.4.1 *输入模式说明*

设置 GPIOx_PMD (PMDn[1:0]) 为 00b ，GPIOx port [n] 为输入模式，I/O管脚为三态（高阻），没有输出驱动能力。GPIOx_PIN的值反映相应端口的状态。

5.6.4.2 *推挽输出模式说明*

设置 GPIOx_PMD (PMDn[1:0]) 为 01b，GPIOxport[n]为推挽输出模式，I/O支持数字输出功能，有拉/灌电流能力。GPIOx_DOUT 相应位bit[n]的值被送到相应管脚上。

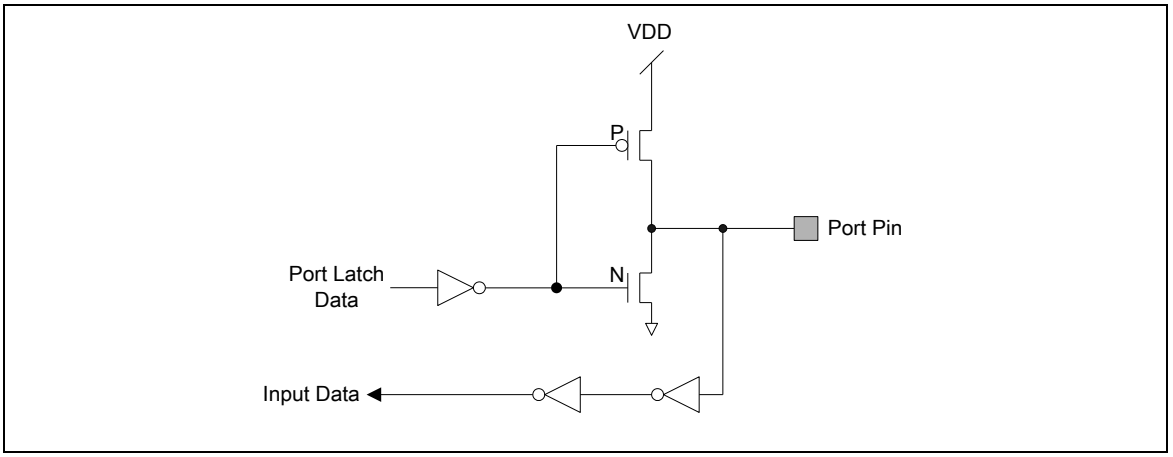


图 5-22 推挽输出

5.6.4.3 *开漏输出模式说明*

设置 GPIOx_PMD (PMDn[1:0])为10b，GPIOx port [n]为开漏模式且I/O管脚数字输出功能仅支持灌电流，驱动到高电平需要一个外加上拉电阻。如果GPIOx_DOUT相应位bit [n]的值为‘0’，管脚上输出低。如果GPIOx_DOUT 相应位bit [n]的值为‘1’，该管脚输出为高，可以由外部上拉电阻控制。

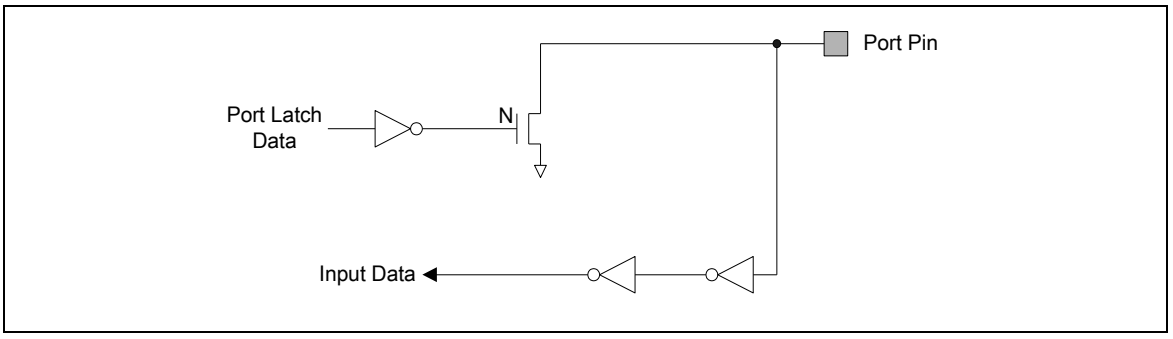


图 5-23 开漏输出

5.6.4.4 *准双向模式说明*

设置GPIOx_PMD(PMDn[1:0])为11b，GPIOxport[n]为准双向模式，I/O同时支持数字输出和输入功能，但拉电流能力仅达数百uA。要实现数字输入，需要先将GPIOx_DOUT相应位置1。准双向输出是80C51 及其派生产品常见的模式。若GPIOx_DOUT相应位bit[n]为‘0’ ，管脚上输出为“低”。若GPIOx_DOUT相应位bit[n]为‘1’，该管脚将检测管脚值。若管脚值为高，没有任何动作，若管脚值为低，在该管脚上将驱动2个时钟周期的高电平，然后禁止强输出驱动，其后管脚状态由内部上拉电阻控制。**注意**: 准双向模式source电流的大小仅有200 uA到30 uA(相应VDD的电压从5.0 V到2.5 V) 。

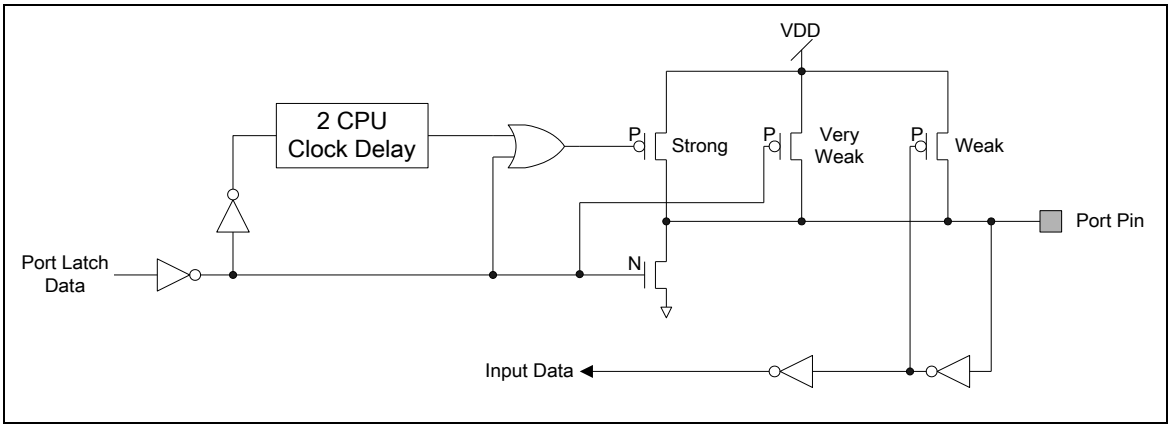


图 5-24 准双向 I/O 模式

5.6.4.5 *GPIO 中断和唤醒功能*

每个GPIO管脚都可以通过GPIOx_IEN位和 GPIOx_IMD设置成芯片的中断源。有四种中断条件可以设置：低电平触发、高电平触发、下降沿触发和上升沿触发。在边沿触发中用户可以通过使能输入信号去抖功能来阻止由噪声引起的意外中断。去抖时钟源和采样周期可以通过DEBOUNCE寄存器来设置。

当芯片进入空闲模式或掉电模式后，GPIO可以用来当作唤醒源，唤醒触发条件设置跟GPIO中断触发设置相同，当GPIO用作唤醒源时要注意

- **进入省电模式前确认I/O状态**

当使用GPIO触发来唤醒系统时，用户必须根据相关的唤醒设置在进入空闲模式或省电模式前确认I/O状态。

例如，配置I/O上升沿/高电平触发唤醒，用户必须确保相应的I/O管脚状态在进入空闲模式/省电模式前是低电平。同样配置I/O下降沿/低电平触发唤醒，用户必须确保相应的I/O管脚状态在进入空闲模式/省电模式前是高电平。

5.6.5 寄存器映射

R: 只读, **W:** 只写, **R/W:** 读/写

寄存器	偏移地址	R/W	描述	复位值
GPIO 基地址: **GPIO_BA = 0x5000_4000**				
GPIOA_PMD	GPIO_BA+0x000	R/W	GPIO 端口 A 管脚模式控制	0xXXXX_XXXX
GPIOA_OFFD	GPIO_BA+0x004	R/W	GPIO 端口 A 关闭数字通路寄存器	0x0000_0000
GPIOA_DOUT	GPIO_BA+0x008	R/W	GPIO 端口 A 数据输出寄存器	0x0000_FFFF
GPIOA_DMASK	GPIO_BA+0x00C	R/W	GPIO 端口 A 数据输出写屏蔽	0x0000_0000
GPIOA_PIN	GPIO_BA+0x010	R	GPIO 端口 A 管脚状态	0x0000_XXXX
GPIOA_DBEN	GPIO_BA+0x014	R/W	GPIO 端口 A 去抖动使能	0x0000_0000
GPIOA_IMD	GPIO_BA+0x018	R/W	GPIO 端口 A 中断模式控制	0x0000_0000
GPIOA_IEN	GPIO_BA+0x01C	R/W	GPIO 端口 A 中断使能	0x0000_0000

寄存器	偏移地址	R/W	描述	复位值
GPIOA_ISRC	GPIO_BA+0x020	R/W	GPIO 端口 A 中断源标志	0x0000_0000
GPIOB_PMD	GPIO_BA+0x040	R/W	GPIO 端口 B 管脚模式控制	0xXXXX_XXXX
GPIOB_OFFD	GPIO_BA+0x044	R/W	GPIO 端口 B 关闭数字通路寄存器	0x0000_0000
GPIOB_DOUT	GPIO_BA+0x048	R/W	GPIO 端口 B 数据输出寄存器	0x0000_FFFF
GPIOB_DMASK	GPIO_BA+0x04C	R/W	GPIO 端口 B 数据输出写屏蔽	0x0000_0000
GPIOB_PIN	GPIO_BA+0x050	R	GPIO 端口 B 管脚状态	0x0000_XXXX
GPIOB_DBEN	GPIO_BA+0x054	R/W	GPIO 端口 B 去抖动使能	0x0000_0000
GPIOB_IMD	GPIO_BA+0x058	R/W	GPIO 端口 B 中断模式控制	0x0000_0000
GPIOB_IEN	GPIO_BA+0x05C	R/W	GPIO 端口 B 中断使能	0x0000_0000
GPIOB_ISRC	GPIO_BA+0x060	R/W	GPIO 端口 B 中断源标志	0x0000_0000
GPIOC_PMD	GPIO_BA+0x080	R/W	GPIO 端口 C 管脚模式控制	0xXXXX_XXXX
GPIOC_OFFD	GPIO_BA+0x084	R/W	GPIO 端口 C 关闭数字通路寄存器	0x0000_0000
GPIOC_DOUT	GPIO_BA+0x088	R/W	GPIO 端口 C 数据输出寄存器	0x0000_FFFF
GPIOC_DMASK	GPIO_BA+0x08C	R/W	GPIO 端口 C 数据输出写屏蔽	0x0000_0000
GPIOC_PIN	GPIO_BA+0x090	R	GPIO 端口 C 管脚状态	0x0000_XXXX
GPIOC_DBEN	GPIO_BA+0x094	R/W	GPIO 端口 C 去抖动使能	0x0000_0000
GPIOC_IMD	GPIO_BA+0x098	R/W	GPIO 端口 C 中断模式控制	0x0000_0000
GPIOC_IEN	GPIO_BA+0x09C	R/W	GPIO 端口 C 中断使能	0x0000_0000
GPIOC_ISRC	GPIO_BA+0x0A0	R/W	GPIO 端口 C 中断源标志	0x0000_0000
GPIOD_PMD	GPIO_BA+0x0C0	R/W	GPIO 端口 D 管脚模式控制	0xXXXX_XXXX
GPIOD_OFFD	GPIO_BA+0x0C4	R/W	GPIO 端口 D 关闭数字通路寄存器	0x0000_0000
GPIOD_DOUT	GPIO_BA+0x0C8	R/W	GPIO 端口 D 数据输出寄存器	0x0000_FFFF
GPIOD_DMASK	GPIO_BA+0x0CC	R/W	GPIO 端口 D 数据输出写屏蔽	0x0000_0000
GPIOD_PIN	GPIO_BA+0x0D0	R	GPIO 端口 D 管脚状态	0x0000_XXXX
GPIOD_DBEN	GPIO_BA+0x0D4	R/W	GPIO 端口 D 去抖动使能	0x0000_0000
GPIOD_IMD	GPIO_BA+0x0D8	R/W	GPIO 端口 D 中断模式控制	0x0000_0000
GPIOD_IEN	GPIO_BA+0x0DC	R/W	GPIO 端口 D 中断使能	0x0000_0000
GPIOD_ISRC	GPIO_BA+0x0E0	R/W	GPIO 端口 D 中断源标志	0x0000_0000
GPIOE_PMD	GPIO_BA+0x100	R/W	GPIO 端口 E 管脚模式控制	0xXXXX_XXXX
GPIOE_OFFD	GPIO_BA+0x104	R/W	GPIO 端口 E 关闭数字通路寄存器	0x0000_0000

寄存器	偏移地址	R/W	描述	复位值
GPIOE_DOUT	GPIO_BA+0x108	R/W	GPIO 端口 E 数据输出寄存器	0x0000_FFFF
GPIOE_DMASK	GPIO_BA+0x10C	R/W	GPIO 端口 E 数据输出写屏蔽	0x0000_0000
GPIOE_PIN	GPIO_BA+0x110	R	GPIO 端口 E 管脚状态	0x0000_XXXX
GPIOE_DBEN	GPIO_BA+0x114	R/W	GPIO 端口 E 去抖动使能	0x0000_0000
GPIOE_IMD	GPIO_BA+0x118	R/W	GPIO 端口 E 中断模式控制	0x0000_0000
GPIOE_IEN	GPIO_BA+0x11C	R/W	GPIO 端口 E 中断使能	0x0000_0000
GPIOE_ISRC	GPIO_BA+0x120	R/W	GPIO 端口 E 中断源标志	0x0000_0000
GPIOF_PMD	GPIO_BA+0x140	R/W	GPIO 端口 F 管脚模式控制	0x0000_00XX
GPIOF_OFFD	GPIO_BA+0x144	R/W	GPIO 端口 F 关闭数字通路寄存器	0x0000_0000
GPIOF_DOUT	GPIO_BA+0x148	R/W	GPIO 端口 F 数据输出寄存器	0x0000_000F
GPIOF_DMASK	GPIO_BA+0x14C	R/W	GPIO 端口 F 数据输出写屏蔽	0x0000_0000
GPIOF_PIN	GPIO_BA+0x150	R	GPIO 端口 F 管脚状态	0x0000_000X
GPIOF_DBEN	GPIO_BA+0x154	R/W	GPIO 端口 F 去抖动使能	0x0000_0000
GPIOF_IMD	GPIO_BA+0x158	R/W	GPIO 端口 F 中断模式控制	0x0000_0000
GPIOF_IEN	GPIO_BA+0x15C	R/W	GPIO 端口 F 中断使能	0x0000_0000
GPIOF_ISRC	GPIO_BA+0x160	R/W	GPIO 端口 F 中断源标志	0x0000_0000
DBNCECON	GPIO_BA+0x180	R/W	外部中断去抖动控制	0x0000_0020
PAn_PDIO n=0,1..15	GPIO_BA+0x200 + 0x04 * n	R/W	GPIO PA.n 端口数据输入/输出	0x0000_000X
PBn_PDIO n=0,1..15	GPIO_BA+0x240 + 0x04 * n	R/W	GPIO PB.n端口数据输入/输出	0x0000_000X
PCn_PDIO n=0,1..15	GPIO_BA+0x280 + 0x04 * n	R/W	GPIO PC.n端口数据输入/输出	0x0000_000X
PDn_PDIO n=0,1..15	GPIO_BA+0x2C0 + 0x04 * n	R/W	GPIO PD.n端口数据输入/输出	0x0000_000X
PEn_PDIO n=0,1..15	GPIO_BA+0x300 + 0x04 * n	R/W	GPIO PE.n端口数据输入/输出	0x0000_000X
PFn_PDIO n=0,1..3	GPIO_BA+0x340 + 0x04 * n	R/W	GPIO PF.n端口数据输入/输出	0x0000_000X

5.6.6 寄存器描述

GPIO 端口 [A/B/C/D/E/F] I/O 模式控制 (GPIOx_PMD)

寄存器	偏移地址	R/W	描述	复位值

GPIOA_PMD	GPIO_BA+0x000	R/W	GPIO 端口 A 管脚 I/O 模式控制	0xXXXX_XXXX
GPIOB_PMD	GPIO_BA+0x040	R/W	GPIO 端口 B 管脚 I/O 模式控制	0xXXXX_XXXX
GPIOC_PMD	GPIO_BA+0x080	R/W	GPIO 端口 C 管脚 I/O 模式控制	0xXXXX_XXXX
GPIOD_PMD	GPIO_BA+0x0C0	R/W	GPIO 端口 D 管脚 I/O 模式控制	0xXXXX_XXXX
GPIOE_PMD	GPIO_BA+0x100	R/W	GPIO 端口 E 管脚 I/O 模式控制	0xXXXX_XXXX
GPIOF_PMD	GPIO_BA+0x140	R/W	GPIO 端口 F 管脚 I/O 模式控制	0x0000_00XX

31	30	29	28	27	26	25	24
PMD15		PMD14		PMD13		PMD12	
23	22	21	20	19	18	17	16
PMD11		PMD10		PMD9		PMD8	
15	14	13	12	11	10	9	8
PMD7		PMD6		PMD5		PMD4	
7	6	5	4	3	2	1	0
PMD3		PMD2		PMD1		PMD0	

位	描述	
[2n+1:2n] n=0,1..15	**PMDn**	**GPIOx I/O pin[n] 模式控制** 决定GPIOx的I/O 类型. 00 = GPIO port [n] 管脚为输入模式 01 = GPIO port [n] 管脚为输出模式 10 = GPIO port [n] 管脚为开漏模式 11 = GPIO port [n] 管脚为准双向模式 **注意**:GPIOF n最大是3，其他n最大是15. 由CIOINI (CONFIG0[10])决定初始值，CIOINI置1 默认值是0xFFFF_FFFF上电后所有端口是准双向模式。CIOINI置0默认值是0x0000_0000，上电后所有端口是输入模式。

GPIO 端口 [A/B/C/D/E/F] 管脚 关闭 数字通路寄存器 (GPIOx_OFFD)

寄存器	偏移地址	R/W	描述	复位值
GPIOA_OFFD	GPIO_BA+0x004	R/W	GPIO 端口 A 管脚关闭数字通路使能	0x0000_0000
GPIOB_OFFD	GPIO_BA+0x044	R/W	GPIO 端口 B 管脚关闭数字通路使能	0x0000_0000
GPIOC_OFFD	GPIO_BA+0x084	R/W	GPIO 端口 C 管脚关闭数字通路使能	0x0000_0000
GPIOD_OFFD	GPIO_BA+0x0C4	R/W	GPIO 端口 D 管脚关闭数字通路使能	0x0000_0000
GPIOE_OFFD	GPIO_BA+0x104	R/W	GPIO 端口 E 管脚关闭数字通路使能	0x0000_0000
GPIOF_OFFD	GPIO_BA+0x144	R/W	GPIO 端口 F 管脚关闭数字通路使能	0x0000_0000

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

位	描述	
[31:16]	**OFFD**	**GPIOx Pin[n] 关闭数字输入通道使能** 用于控制GPIO的数字输入通路是否使能. 如果输入为模拟信号，用户可以关闭输入通道防止漏电 0 = 使能IO数据输入通道 1 = 关闭IO的数字输入通道(数字输入拉低) **注意**:GPIOF n最大是3，其他n最大是15.
[15:0]	**保留**	保留

GPIO 端口 [A/B/C/D/E/F] 数据输出值(GPIOx_DOUT)

寄存器	偏移地址	R/W	描述	复位值
GPIOA_DOUT	GPIO_BA+0x008	R/W	GPIO 端口 A 数据输出值	0x0000_FFFF
GPIOB_DOUT	GPIO_BA+0x048	R/W	GPIO 端口 B 数据输出值	0x0000_FFFF
GPIOC_DOUT	GPIO_BA+0x088	R/W	GPIO 端口 C 数据输出值	0x0000_FFFF
GPIOD_DOUT	GPIO_BA+0x0C8	R/W	GPIO 端口 D 数据输出值	0x0000_FFFF
GPIOE_DOUT	GPIO_BA+0x108	R/W	GPIO 端口 E 数据输出值	0x0000_FFFF
GPIOF_DOUT	GPIO_BA+0x148	R/W	GPIO 端口 F 数据输出值	0x0000_000F

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

位	描述	
[31:16]	**保留**	保留.
[n] n = 0,1..15	**DOUT[n]**	**GPIOx Pin[n] 输出值** 在GPIO配置成输出，开漏和准双向模式时，控制GPIO相应管脚的状态. 1 = GPIO配置成输出，开漏和准双向模式时，GPIO port [A/B/C/D/E/F] Pin[n] 为高 0 = GPIO配置成输出，开漏和准双向模式时，GPIO port [A/B/C/D/E/F] Pin[n] 为低 **注意**:GPIOF n最大是3，其他n最大是15.

GPIO 端口 [A/B/C/D/E/F] 数据输出写屏蔽(GPIOx _DMASK)

寄存器	偏移地址	R/W	描述	复位值
GPIOA_DMASK	GPIO_BA+0x00C	R/W	GPIO 端口 A 数据输出写屏蔽	0x0000_0000
GPIOB_DMASK	GPIO_BA+0x04C	R/W	GPIO 端口 B 数据输出写屏蔽	0x0000_0000
GPIOC_DMASK	GPIO_BA+0x08C	R/W	GPIO 端口 C 数据输出写屏蔽	0x0000_0000
GPIOD_DMASK	GPIO_BA+0x0CC	R/W	GPIO 端口 D 数据输出写屏蔽	0x0000_0000
GPIOE_DMASK	GPIO_BA+0x10C	R/W	GPIO 端口 E 数据输出写屏蔽	0x0000_0000
GPIOF_DMASK	GPIO_BA+0x14C	R/W	GPIO 端口 F 数据输出写屏蔽	0x0000_0000

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

位	描述	
[31:16]	**保留**	保留
[n] n = 0,1..15	**DMASK[n]**	**端口 [A/B/C/D/E/F] 数据输出写屏蔽** 用于保护相应寄存器GPIOx_DOUT bit[n]. 当设置DMASK bit[n] 为 '1'时，相应DOUT[n] bit 被保护，写信号被屏蔽时，不能向保护位写数据。 1 =保护相应的GPIO_DOUT[n] 位 0 =相应的GPIO_DOUT[n] 位可以被更新 **注意**：该功能只保护相应的PIOx_DOUT[n]位，不保护(PAn_PDIO, PBn_PDIO, PCn_PDIO, PDn_PDIO, PEn_PDIO and PFn_PDIO).相应位 GPIOF n最大是3，其他n最大是15.

GPIO 端口 [A/B/C/D/E/F] 管脚数据(GPIOx _PIN)

寄存器	偏移地址	R/W	描述	复位值
GPIOA_PIN	GPIO_BA+0x010	R	GPIO 端口 A 管脚数据	0x0000_XXXX
GPIOB_PIN	GPIO_BA+0x050	R	GPIO 端口 B 管脚数据	0x0000_XXXX
GPIOC_PIN	GPIO_BA+0x090	R	GPIO 端口 C 管脚数据	0x0000_XXXX
GPIOD_PIN	GPIO_BA+0x0D0	R	GPIO 端口 D 管脚数据	0x0000_XXXX
GPIOE_PIN	GPIO_BA+0x110	R	GPIO 端口 E 管脚数据	0x0000_XXXX
GPIOF_PIN	GPIO_BA+0x150	R	GPIO 端口 F 管脚数据	0x0000_000X

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

位	描述	
[31:16]	**保留**	保留
[n] n = 0,1..15	**PIN[n]**	**端口 [A/B/C/D/E/F] 管脚数据** 这些位的值为各个GPIO管脚真实状态的反映。如果值为'1'，表示相应管脚状态为高，否则为低 **注意**：GPIOF n最大是3，其他n最大是15.

GPIO 端口 [A/B/C/D/E/F] 去抖动使能(GPIOx _DBEN)

寄存器	偏移地址	R/W	描述	复位值
GPIOA_DBEN	GPIO_BA+0x014	R/W	GPIO 端口 A 去抖动使能	0x0000_0000
GPIOB_DBEN	GPIO_BA+0x054	R/W	GPIO 端口 B 去抖动使能	0x0000_0000
GPIOC_DBEN	GPIO_BA+0x094	R/W	GPIO 端口 C 去抖动使能	0x0000_0000
GPIOD_DBEN	GPIO_BA+0x0D4	R/W	GPIO 端口 D 去抖动使能	0x0000_0000
GPIOE_DBEN	GPIO_BA+0x114	R/W	GPIO 端口 E 去抖动使能	0x0000_0000
GPIOF_DBEN	GPIO_BA+0x154	R/W	GPIO 端口 F 去抖动使能	0x0000_0000

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

位	描述	
[31:16]	**保留**	保留
[n] n = 0,1..15	**DBEN[n]**	**端口 [A/B/C/D/E/F] 输入信号去抖动使能** DBEN[n]用于使能相应位的去抖动功能. 如果输入信号脉冲宽度不能被两个连续的去抖动采样周期所采样，则被视为信号反弹，从而不触发中断. 去抖动时钟源由DBNCECON[4]控制，一个去抖动周期由DBNCECON[3:0]控制。DBEN[n] 仅用于"边沿触发"中断，不能用于"电平触发"中断。 1 = 使能 bit[n] 去抖动功能 0 = 禁用 bit[n] 去抖动功能 去抖动功能对于边沿触发中断有效， 对于电平触发中断模式，去抖动功能使能位不起作用. **注意**：GPIOF n最大是3，其他n最大是15.

GPIO 端口 [A/B/C/D/E/F] 中断模式控制(GPIOx _IMD)

寄存器	偏移地址	R/W	描述	复位值
GPIOA_IMD	GPIO_BA+0x018	R/W	GPIO 端口 A 中断模式控制	0x0000_0000
GPIOB_IMD	GPIO_BA+0x058	R/W	GPIO 端口 B 中断模式控制	0x0000_0000
GPIOC_IMD	GPIO_BA+0x098	R/W	GPIO 端口 C 中断模式控制	0x0000_0000
GPIOD_IMD	GPIO_BA+0x0D8	R/W	GPIO 端口 D 中断模式控制	0x0000_0000
GPIOE_IMD	GPIO_BA+0x118	R/W	GPIO 端口 E 中断模式控制	0x0000_0000
GPIOF_IMD	GPIO_BA+0x158	R/W	GPIO 端口 F 中断模式控制	0x0000_0000

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

位	描述	
[31:16]	**保留**	保留
[n] n = 0,1..15	**IMD[n]**	**端口 [A/B/C/D/E/F] 边沿或电平检测中断控制** IMD[n] 用于控制电平触发或边沿触发中断. 若为边沿触发中断，触发源可由去抖动控制，如果是电平触发中断，输入源由一个HCLK时钟采样并产生中断。 0 = 边沿触发中断 1 = 电平触发中断 如果设置管脚为电平触发模式，则在寄存器GPIOX_IEN中，只能设置一个电平(高电平或者低电平)；若设置了两个电平都触发中断，则设置被忽略，不会产生中断 去抖动功能对于边沿触发中断有效，对于电平触发中断无效 **注意**：GPIOF n最大是3，其他n最大是15.

GPIO 端口 [A/B/C/D/E/F] 中断使能控制(GPIOx _IEN)

寄存器	偏移地址	R/W	描述	复位值
GPIOA_IEN	GPIO_BA+0x01C	R/W	GPIO 端口 A 中断使能	0x0000_0000
GPIOB_IEN	GPIO_BA+0x05C	R/W	GPIO 端口 B 中断使能	0x0000_0000
GPIOC_IEN	GPIO_BA+0x09C	R/W	GPIO 端口 C 中断使能	0x0000_0000
GPIOD_IEN	GPIO_BA+0x0DC	R/W	GPIO 端口 D 中断使能	0x0000_0000
GPIOE_IEN	GPIO_BA+0x11C	R/W	GPIO 端口 E 中断使能	0x0000_0000
GPIOF_IEN	GPIO_BA+0x15C	R/W	GPIO 端口 F 中断使能	0x0000_0000

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

位	描述	
[n+16] n = 0,1..15	**IR_EN[n]**	**端口 [A/B/C/D/E/F] 输入上升沿或输入高电平中断使能** IR_EN[n] 用于使能相应GPIO_PIN[n]输入的中断. 置'1'也可以使能管脚唤醒功能 当设置 IR_EN[n] 位为'1': 如果中断是电平触发模式，输入PIN[n]的状态为高电平时，产生中断. 如果中断是边沿触发模式，输入PIN[n]的状态由低电平到高电平变化时，产生中断. 1 = 使能PIN[n]高电平或由低电平到高电平变化的中断 0 = 禁用PIN[n]高电平或由低电平到高电平变化的中断 **注意**：GPIOF n最大是3，其他n最大是15.
[n] n = 0,1..15	**IF_EN[n]**	**使能端口 [A/B/C/D/E/F] 输入下降沿或输入低电平的中断** IF_EN[n] 用于使能相应GPIO_PIN[n]输入的中断. 置 '1' 也可以使能管脚唤醒功能 当设置 IF_EN[n] 位为 '1': 如果中断是电平触发模式，输入PIN[n]的状态为低电平时，产生中断. 如果中断是边沿触发模式，输入PIN[n]的状态由高电平到低电平变化时，产生中断. 1 = 使能PIN[n]低电平或由高电平到低电平变化的中断 0 = 禁用PIN[n]低电平或由高电平到低电平变化的中断 **注意**：GPIOF n最大是3，其他n最大是15.

GPIO 端口 [A/B/C/D/E/F] 中断触发源(GPIOx _ISRC)

寄存器	偏移地址	R/W	描述	复位值
GPIOA_ISRC	GPIO_BA+0x020	R/W	GPIO 端口 A 中断触发源标志	0x0000_0000
GPIOB_ISRC	GPIO_BA+0x060	R/W	GPIO 端口 B 中断触发源标志	0x0000_0000
GPIOC_ISRC	GPIO_BA+0x0A0	R/W	GPIO 端口 C 中断触发源标志	0x0000_0000
GPIOD_ISRC	GPIO_BA+0x0E0	R/W	GPIO 端口 D 中断触发源标志	0x0000_0000
GPIOE_ISRC	GPIO_BA+0x120	R/W	GPIO 端口 E 中断触发源标志	0x0000_0000
GPIOF_ISRC	GPIO_BA+0x160	R/W	GPIO 端口 F 中断触发源标志	0x0000_0000

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

位	描述	
[31:16]	**保留**	保留.
[n] n = 0,1..15	**ISRC[n]**	**端口 [A/B/C/D/E/F] 中断触发源标志** 读 : 1 = GPIOx[n]产生中断 0 = GPIOx[n]没有中断 写 : 1= 清相应的未处理中断标志 0= 无动作 **注意**：GPIOF n最大是3，其他n最大是15.

中断去抖动周期控制(DBNCECON)

寄存器	偏移地址	R/W	描述	复位值
DBNCECON	GPIO_BA+0x180	R/W	外部中断去抖动控制	0x0000_0020

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

位	描述	
[5]	ICLK_ON	**中断时钟 On 模式** 1 = 复位之后所有 I/O边沿侦测电路使能. 0 = 边沿侦测电路仅在 I/O管脚对应的 GPIOx_IEN位置 1有效. 如果没有相关的特定应用，建议关掉时钟以减少耗电
[4]	DBCLKSRC	**去抖动计数器时钟源选择** 1 = 去抖动计数器时钟源为内部 10 KHz 时钟 0 = 去抖动计数器时钟源为 HCLK
[3:0]	DBCLKSEL	**去抖动采样周期选择**

DB LKSEL	描述
0	采样中断输入信号，每 1 clocks一次
1	采样中断输入信号，每 2 clocks一次
2	采样中断输入信号，每 4 clocks一次
3	采样中断输入信号，每 8 clocks一次
4	采样中断输入信号，每 16 clocks一次
5	采样中断输入信号，每 32 clocks一次
6	采样中断输入信号，每 64 clocks一次
7	采样中断输入信号，每 128 clocks一次
8	采样中断输入信号，每 256 clocks一次
9	采样中断输入信号，每 2*256 clocks一次
10	采样中断输入信号，每 4*256 clocks一次
11	采样中断输入信号，每 8*256 clocks一次

		12	采样中断输入信号，每 16*256 clocks一次	
		13	采样中断输入信号，每 32*256 clocks一次	
		14	采样中断输入信号，每 64*256 clocks一次	
		15	采样中断输入信号，每 128*256 clocks一次	

GPIO Px.n管脚数据输入/输出控制 (Pxn_PDIO)

寄存器	偏移地址	R/W	描述	复位值
PAn_PDIO n=0,1..15	GPIO_BA+0x200 + 0x04 * n	R/W	GPIO PA.n管脚数据输入/输出	0x0000_000X
PBn_PDIO n=0,1..15	GPIO_BA+0x240 + 0x04 * n	R/W	GPIO PB.n管脚数据输入/输出	0x0000_000X
PCn_PDIO n=0,1..15	GPIO_BA+0x280 + 0x04 * n	R/W	GPIO PC.n 管脚数据输入/输出	0x0000_000X
PDn_PDIO n=0,1..15	GPIO_BA+0x2C0 + 0x04 * n	R/W	GPIO PD.n管脚数据输入/输出	0x0000_000X
PEn_PDIO n=0,1..15	GPIO_BA+0x300 + 0x04 * n	R/W	GPIO PE.n管脚数据输入/输出	0x0000_000X
PFn_PDIO n=0,1..3	GPIO_BA+0x340 + 0x04 * n	R/W	GPIO PF.n 管脚数据输入/输出	0x0000_000X

注意: x = A/B/C/D/E/F ， n = 0~15

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

位	描述	
[0]	**Pxn_PDIO**	**GPIO Px.n管脚 数据输入/输出控制** 写该位可以控制一个GPIO管脚的输出值 1 = 设置相应GPIO管脚为高 0 = 设置相应GPIO管脚为低 读该寄存器得到GPIO管脚状态 例如: 写PA0_PDIO即把值写到GPIOA_DOUT[0]位上, 读PA0_PDIO即读取GPIOA_PIN[0]的值. **注意**：写操作不受GPIOx_DMASK影响

5.7 PDMA 控制器 (PDMA)

5.7.1 概述

NuMicro™ NUC200 系列DMA包含九个通道外设直接存储器存取 (PDMA) 控制器和一个循环冗余检查(CRC)发生器。

PDMA是帮助内存或APB外设搬移数据的模块。PDMA (DMA CH0~CH8)在外围 APB 设备和存储器之间有一个字大小的缓存作为传输缓存。软件可以停止PDMA通过禁止PDMA PDMA_CSRx[PDMACEN]位。通过软件轮询或者收到内部的PDMA 中断，CPU 可以识别PDMA 运作的完成。PDMA控制器可以设置源地址和目的地址的移动方式为累加、递减或固定三种之一。

DMA控制器也包含一个循环冗余检查(CRC)发生器。它可以执行带可编程多项式设定的CRC运算。CRC支持 CPU PIO模式和DMA传输模式。

5.7.2 特征

- 支持9个PDMA通道和一个CRC通道。每个通道能支持一个单向传输。
- AMBA AHB主机/从机接口兼容，用于数据传输和寄存器读/写
- 硬件通道优先级。DMA 通道 0 拥有最高优先级，通道 8 拥有最低优先级。
- PDMA 操作
  - 外设到内存、内存到外设、内存到内存传输。
  - 支持字/半字/字节传输数据宽度 从/到外设。
  - 支持地址方向：递增、固定。
- 循环冗余检查(CRC)
  - 支持四个通用的多项式CRC-CCITT, CRC-8, CRC-16, 和 CRC-32
    - CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$
    - CRC-8: $X^8 + X^2 + X + 1$
    - CRC-16: $X^{16} + X^{15} + X^2 + 1$
    - CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
  - 可编程的CRC种子值.
  - 支持对输入数据和 CRC 校验和的可编程的反序设定。
  - 支持对输入数据和 CRC 校验和的可编程的一次补码设定。
  - 支持 CPU PIO 模式或 DMA 传输模式.
  - 在 CPU PIO 模式下，支持下面写数据宽度
    - 8-bit 写模式 (字节): 1-AHB时钟周期操作
    - 16-bit 写模式 (半字): 2-AHB时钟周期操作
    - 32-bit 写模式 (字): 4-AHB时钟周期操作
  - 在 CRC DMA 模式下，支持字节对齐传输长度和字对齐传输源地址

5.7.3 框图

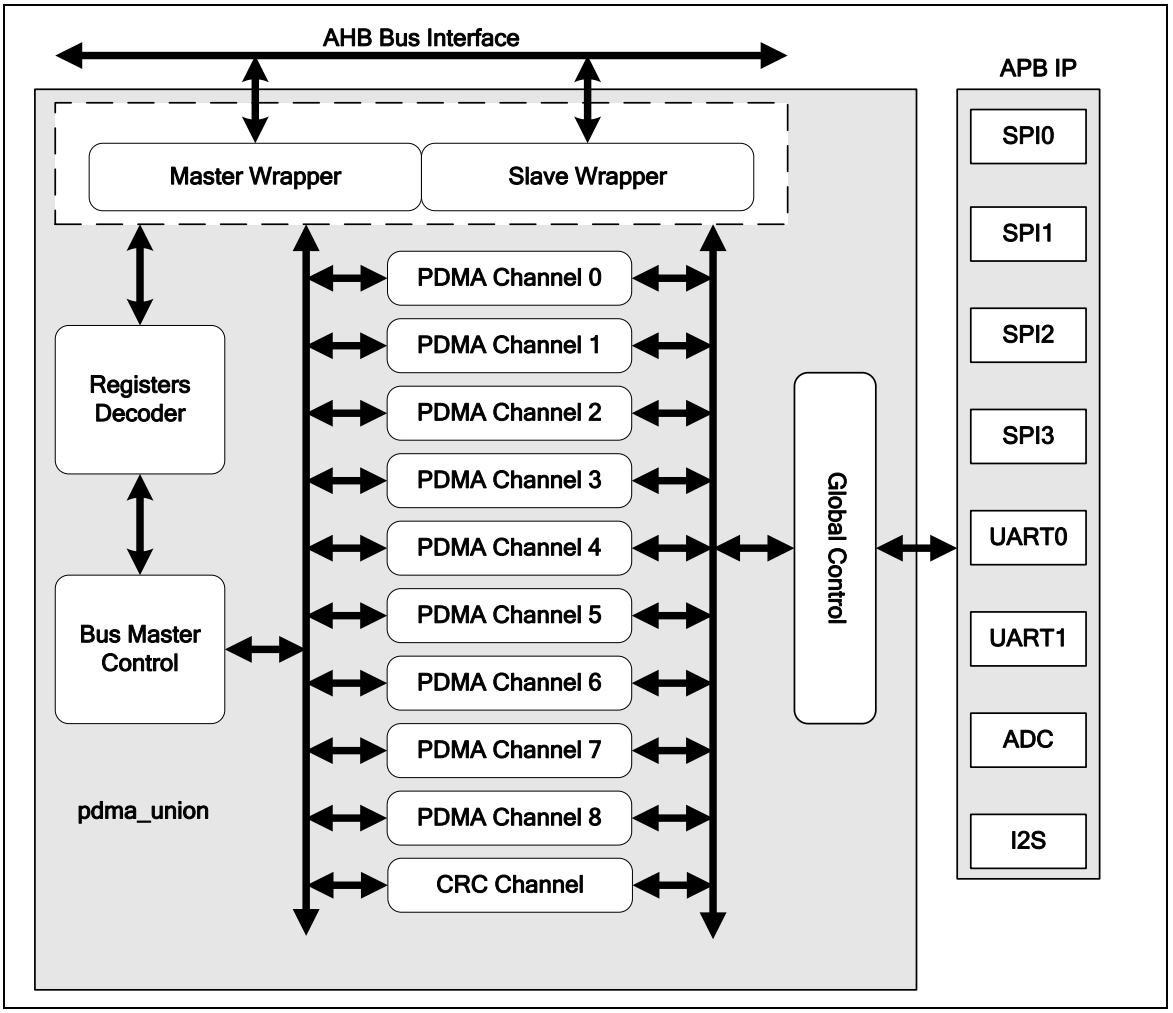


图 5-25 DMA 控制器模块框图

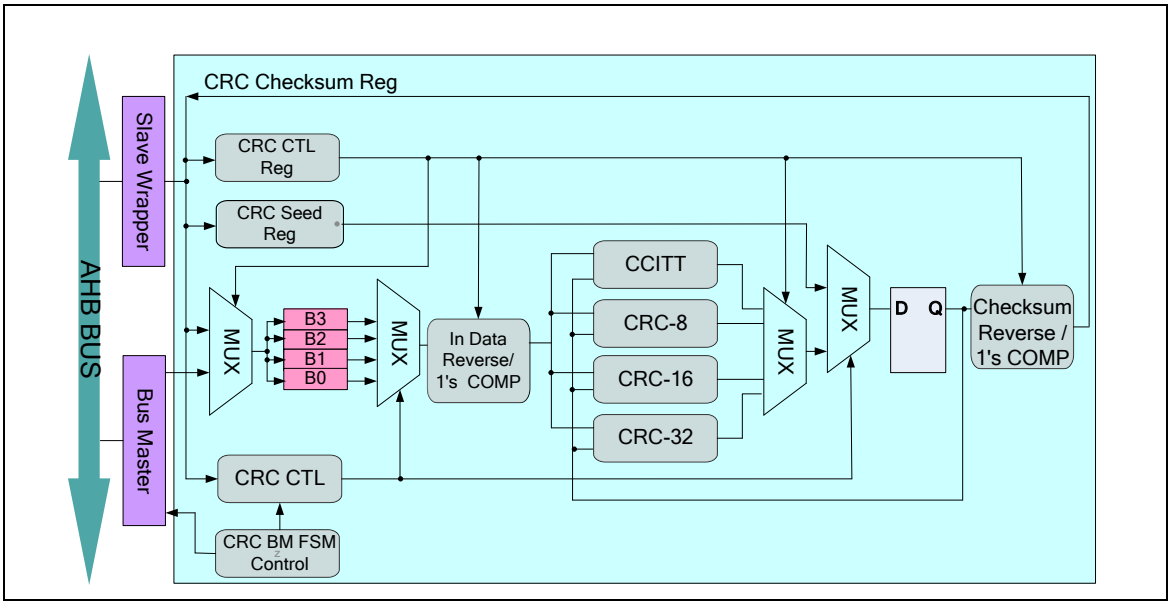


图 5-26 CRC 发生器模块图

5.7.4 基本配置

PDMA控制器外部时钟可以通过PDMA_EN (AHBCLK[1])使能。

5.7.5 功能描述

直接存储器存取(DMA) 控制器模块传输数据从一个地址到另外一个地址，而不需要 CPU 的参与。. DMA 控制器包括九个通道PDMA (外设到内存或内存到外设或内存到内存).和一个CRC发生器通道。

通过软件轮询或者收到内部的DMA 中断，CPU 可以识别DMA 运作的完成。

5.7.5.1 *PDMA*

PDMA 控制器有9通道PDMA (外设到内存或内存到外设或内存到内存)。对于源和目的地址而言，PDMA 控制器具有两种模式：增加模式和固定模式。

每个 PDMA 模拟通道没有预先设定默认值，因此用户必须在开始相关的 PDMA 通道之前通过设定 PDMA_PDSSR0、PDMA_PDSSR1 和 PDMA_PDSSR2 进行配置。

软件必须通过设置PDMACEN (PDMA_CSRx[0])位使能PDMA通道并且写有效的源地址到 PDMA_SARx 寄存器，写目的地址到 PDMA_DARx 寄存器，写传输数量到 PDMA_BCRx 寄存器。然后触发TRIG_EN (PDMA_CSRx[23])。PDMA 将继续传输直到 PDMA_CBCRx 降为 0。下面是一个编程顺序的例子。

- 通过设置 AHBCLK 寄存器中 PDMA_EN 位使能 PDMA 引擎时钟
- 通过设置 PDMA_PDSSR0/ PDMA_PDSSR1/ PDMA_PDSSR2 寄存器配置 PDMA 通道服务
- 配置 PDMA_CSRx 寄存器:
  - 使能 PDMA 通道(PDMACEN (PDMA_CSRx[0]))
  - 设置源/目的地址方向 (SAD_SEL (PDMA_CSRx[5:4]) / DAD_SEL (PDMA_CSRx[7:6]))
  - 配置 PDMA 模式选择(MODE_SEL (PDMA_CSRx[3:2]))
  - 配置外设传输宽度选择 (APB_TWS (PDMA_CSRx[20:19]))
- 通过配置 PDMA_SARx/PDMA_DARx 寄存器，设置源/目的地址。
- 通过设置 PDMA_BCRx 寄存器，配置PDMA传输字节计数。
- 通过设置 BLKD_IE(PDMA_IERx [1])（可选）使能 PDMA 块传输结束中断。
- 通过设置 NVIC_ISER 寄存器 bit 26 为 “1”(可选)，使能 PDMA NVIC。
- 通过设置TRIG_EN (PDMA_CSRx[23])位，使能 PDMA 读/写 传输。
- 如果 PDMA 块传输结束中断产生，软件写 “1” 到BLKD_IF (PDMA_ISRx[1]) 清0中断标志
- 通过设置TRIG_EN (PDMA_CSRx[23])位，使能下一次块传输 PDMA 读/写 传输。

如果在PDMA操作期间发生一个错误，通道停止除非软件清除错误条件，并设置SW_RST (PDMA_CSRx[1]) 来复位 PDMA 通道， 设置 PDMACEN (PDMA_CSRx[0]) 和 TRIG_EN (PDMA_CSRx[23])位域重新开始。

在PDMA(外设到内存或内存到外设)模式，DMA可以传输数据在外设APB IP (e.g. UART, SPI, ADC)和存储器之间。

5.7.5.2 *CRC*

DMA控制器包含一个循环冗余检查(CRC)发生器。它可以执行带可编程多项式设定的CRC运算。多项式操作包括CRC-CCITT, CRC-8, CRC-16 和 CRC-32。通过设置CRC多项式模式(CRC_MODE (CRC_CTL[31:30])，软件可以选择多项式操作模式。

CRC 引擎支持CPU PIO 模式条件：CRC通道使能位CRCCEN (CRC_CLT[0])是1，触发使能位TRIG_EN (CRC_CTL[23]) 是0。支持DMA传输模式条件： CRC 通道使能位CRCCEN (CRC_CLT[0])是 1, 触发使能位(CRC_CTL[23]) 是 1。下面是一个编程顺序的例子：

CPU PIO 模式操作步骤:

- 通过设置CRCCEN 位 (CRC_CLT [0] CRC 通道使能)为1，使能 CRC 引擎。
- 设置传输数据格式 (WDATA_RVS (CRC_CTL[24])， CHECKSUM_RVS (CRC_CTL[25]), WDATA_COM(CRC_CTL[26]) 和 CHECKSUM_COM (CRC_CTL[27]))，初始化种子值(CRC_SEED (CRC_SEED[31:0])) 并通过设置(CPU_WDLEN (CRC_CTL[29:28]))寄存器选择数据长度。
- 设置引擎复位位 CRC_RST (CRC_CTL[1])为 1 来加载初始种子值到 CRC 电路，但是CRT_CTL 寄存器的其他内容不会清 0。该位会被自动清 0。
- 写数据到CRC 写数据寄存器(CRC_WDATA (CRC_WDATA[31:0]))来执行CRC 运算。
- 通过读 CRC 校验和寄存器(CRC_CHECKSUM (CRC_CHECKSUM[31:0]))，获得 CRC 校验和结果

CRC DMA模式操作步骤：

- 通过设置通道使能位 CRCCEN (CRC_CLT[0])为 1，使能 CRC 引擎。
- 设置数据格式 (WDATA_RVS (CRC_CTL[24]), (CHECKSUM_RVS (CRC_CTL[25]), WDATA_COM (CRC_CTL[26]), CHECKSUM_COM (CRC_CTL[27]))，初始化种子值(CRC_SEED (CRC_SEED[31:0]))。.
- 通过设置 CRC_DMASAR (CRC_DMASAR[31:0]) 和 CRC_DMABCR (CRC_DMABCR[15:0])给出一个有效的源地址（字对齐）和传输长度.
- 设置触发使能位 TRIG_EN (CRC_CTL[23])为 1 执行 CRC 运算。
- 等待 CRC DMA 传输，并通过块传输结束中断标志(CRC_BLKD_IF (CRC_DMAISR[1]))检查 CRC DMA 传输是否完成，然后通过读 CRC 校验寄存器(CRC_CHECKSUM (CRC_CHECKSUM[31:0]))获得 CRC 校验和结果。

5.7.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
PDMA 基地址: **PDMA_CHx_BA = 0x5000_8000 + 0x100 * x** **x=0,1 .. 8** **CRC_BA = 0x5000_8E00** **PDMA_GCR_BA = 0x5000_8F00**				
PDMA_CSRx **x=0,1 .. 8**	PDMA_CHx_BA+0x00	R/W	PDMA 通道 x控制寄存器	0x0000_0000
PDMA_SARx **x=0,1 .. 8**	PDMA_CHx_BA+0x04	R/W	PDMA通道x源地址寄存器	0x0000_0000

PDMA_DARx x=0,1 .. 8	PDMA_CHx_BA+0x08	R/W	PDMA通道x目的地址寄存器	0x0000_0000
PDMA_BCRx x=0,1 .. 8	PDMA_CHx_BA+0x0C	R/W	PDMA通道x传输字节计数寄存器	0x0000_0000
PDMA_POINTx x=0,1 .. 8	PDMA_CHx_BA+0x10	R	PDMA通道x内部缓存指针	0xXXXX_0000
PDMA_CSARx x=0,1 .. 8	PDMA_CHx_BA+0x14	R	PDMA通道x当前源地址寄存器	0x0000_0000
PDMA_CDARx x=0,1 .. 8	PDMA_CHx_BA+0x18	R	PDMA通道x当前目的地址寄存器	0x0000_0000
PDMA_CBCRx x=0,1 .. 8	PDMA_CHx_BA+0x1C	R	PDMA通道x当前传输字节计数寄存器	0x0000_0000
PDMA_IERx x=0,1 .. 8	PDMA_CHx_BA+0x20	R/W	PDMA通道x中断使能寄存器	0x0000_0001
PDMA_ISRx x=0,1 .. 8	PDMA_CHx_BA+0x24	R/W	PDMA通道x中断状态寄存器	0x0000_0000
PDMA_SBUF0_Cx x=0,1 .. 8	PDMA_CHx_BA+0x80	R	PDMA通道x共享缓存FIFO 0 寄存器	0x0000_0000
CRC_CTL	CRC_BA+0x00	R/W	CRC 控制寄存器	0x2000_0000
CRC_DMASAR	CRC_BA+0x04	R/W	CRC DMA 源地址寄存器	0x0000_0000
CRC_DMABCR	CRC_BA+0x0C	R/W	CRC DMA 传输字节计数寄存器	0x0000_0000
CRC_DMACSAR	CRC_BA+0x14	R	CRC DMA 当前源地址寄存器	0x0000_0000
CRC_DMACBCR	CRC_BA+0x1C	R	CRC DMA 当前传输字节计数寄存器	0x0000_0000
CRC_DMAIER	CRC_BA+0x20	R/W	CRC DMA 中断使能寄存器	0x0000_0001
CRC_DMAISR	CRC_BA+0x24	R/W	CRC DMA中断状态寄存器	0x0000_0000
CRC_WDATA	CRC_BA+0x80	R/W	CRC 写数据寄存器	0x0000_0000
CRC_SEED	CRC_BA+0x84	R/W	CRC 种子寄存器	0xFFFF_FFFF
CRC_CHECKSUM	CRC_BA+0x88	R	CRC 校验和寄存器	0xFFFF_FFFF
PDMA_GCRCSR	PDMA_GCR_BA+0x00	R/W	PDMA 全局控制寄存器	0x0000_0000
PDMA_PDSSR0	PDMA_GCR_BA+0x04	R/W	PDMA服务选择控制寄存器0	0xFFFF_FFFF
PDMA_PDSSR1	PDMA_GCR_BA+0x08	R/W	PDMA服务选择控制寄存器1	0xFFFF_FFFF
PDMA_GCRISR	PDMA_GCR_BA+0x0C	R	PDMA 全局中断寄存器	0x0000_0000
PDMA_PDSSR2	PDMA_GCR_BA+0x10	R/W	PDMA 服务选择控制寄存器 2	0x0000_00FF

5.7.7 寄存器描述

PDMA通道 x 控制寄存器(PDMA_CSRx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_CSRx x=0,1 .. 8	PDMA_CHx_BA+0x00	R/W	PDMA通道 x 控制寄存器	0x0000_0000

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

位	描述	
[31:24]	**Reserved**	保留.
[23]	**TRIG_EN**	**触发使能** 1 = 使能 PDMA 数据读或写传输 0 = 不起作用 **注意**：当 PDMA传输完成时，该位将被自动清除。 如果总线错误出现，则所有的PDMA 传输将停止。软件必须复位所以的 PDMA 通道，然后再次触发。
[22:21]	**Reserved**	保留.
[20:19]	**APB_TWS**	**外设传输宽度选择** 00 = 每个 PDMA 操作传输一个字 (32-bit) 01 = 每个 PDMA 操作传输一个字节 (8-bit) 10 = 每个 PDMA 操作传输一个半字 (16-bit) 11 = 保留 **注意**：当且仅当 MODE_SEL 是外设到存储器模式 (Peripheral-to-Memory) 或 存储器到外设模式 (Memory-to-Peripheral) 时，该域才有意义。
[18:8]	**Reserved**	保留.
[7:6]	**DAD_SEL**	**传输目标地址方向选择** 00 = 传输目标地址持续增加 01 = 保留. 10 = 传输目标地址固定（该特性可被用于当数据从多个源地址到一个单独的目的地址传输的情况） 11 = 保留.

[5:4]	**SAD_SEL**	**传输源地址方向选择** 00 = 传输源地址持续增加 01 = 保留 10 = 传输源地址固定（该特性可被用于当数据从一个单独的源地址到多个目的地址传输的情况） 11 = 保留
[3:2]	**MODE_SEL**	**PDMA 模式选择** 00 = 存储器到存储器模式 (Memory-to-Memory). 01 = 外设到存储器模式 (Peripheral-to-Memory). 10 = 存储器到外设模式 (Memory-to-Peripheral).
[1]	**SW_RST**	**软件引擎复位** 0 = 无效 1 = 复位内部状态机，指针和内部缓存。控制寄存器的内部不会被清除。该位在几个时钟周期之后将自动清除。
[0]	**PDMACEN**	**PDMA 通道使能** 设置该位为 1 使能 PDMA 的操作。如果该位被清除，PDMA 将忽略所有的 PDMA 请求并强制总线主机进入 IDLE 状态。 **注意**：SW_RST(PDMA_CSRx[1], x= 0~8) 将清除该位。

PDMA通道 x源地址寄存器(PDMA_SARx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_SARx x=0,1 .. 8	PDMA_CHx_BA+0x04	R/W	PDMA通道 x源地址寄存器	0x0000_0000

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

位	描述	
[31:0]	PDMA_SAR	**PDMA 传输源地址寄存器** 该域表示一 32-位 PDMA 源地址。 **注意：**源地址必须字对齐。

PDMA通道 x 目标地址寄存器(PDMA_DARx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_DARx **x=0,1 .. 8**	PDMA_CHx_BA+0x08	R/W	PDMA通道 x 目标地址寄存器	0x0000_0000

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

位	描述	
[31:0]	**PDMA_DAR**	**PDMA 传输目标地址寄存器** 该域表示一 32-位 PDMA目标地址 **注意：**目标地址必须字对齐

PDMA通道 x 传输字节计数寄存器(PDMA_BCRx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_BCRx x=0,1 .. 8	PDMA_CHx_BA+0x0C	R/W	PDMA通道 x 传输字节计数寄存器	0x0000_0000

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

位	描述	
[31:16]	**Reserved**	保留.
[15:0]	**PDMA_BCR**	**PDMA 传输字节计数寄存器** 该域表示一 16-位 PDMA 传输字节计数数目，必须字对齐。

PDMA通道 x内部缓存指针寄存器(PDMA_POINTx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_POINTx x=0,1 .. 8	PDMA_CHx_BA+0x10	R	PDMA通道 x内部缓存指针寄存器	0xXXXX_0000

31	30	29	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				PDMA_POINT			

位	描述	
[31:4]	Reserved	保留
[3:0]	PDMA_POINT	PDMA 内部缓存指针寄存器（只读） 该域表示内部缓存指针。

PDMA通道 x当前源地址寄存器(PDMA_CSARx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_CSARx x=0,1 .. 8	PDMA_CHx_BA+0x14	R	PDMA通道 x当前源地址寄存器	0x0000_0000

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

位	描述	
[31:0]	PDMA_CSAR	PDMA 当前源地址寄存器（只读） 该域指示 PDMA 正在传输的源地址。

PDMA通道 x当前目标地址寄存器(PDMA_CDARx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_CDARx x=0,1 .. 8	PDMA_CHx_BA+0x18	R	PDMA通道 x当前目标地址寄存器	0x0000_0000

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

位	描述	
[31:0]	PDMA_CDAR	**PDMA 当前目的地址寄存器（只读）** 该域指示 PDMA 正在传输的目的地址。

PDMA通道 x当前字节计数寄存器(PDMA_CBCRx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_CBCRx **x=0,1 .. 8**	PDMA_CHx_BA+0x1C	R	PDMA通道 x当前字节计数寄存器	0x0000_0000

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

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**PDMA_CBCR**	**PDMA 当前字节计数寄存器（只读）** 该域指示 PDMA 当前剩下的字节计数。 **注意：** SW_RST (PDMA_CSRx[1])置1，该域值清零。

PDMA通道 x中断使能控制寄存器(PDMA_IERx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_IERx x=0,1 .. 8	PDMA_CHx_BA+0x20	R/W	PDMA通道 x中断使能控制寄存器	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						BLKD_IE	TABORT_IE

位	描述	
[31:2]	**Reserved**	保留.
[1]	**BLKD_IE**	**PDMA 传输完成中断使能** 1 = 使能 PDMA 传输完成中断 0 = 禁用 PDMA 传输完成中断
[0]	**TABORT_IE**	**PDMA 读/写目标中止中断使能** 1 = 使能 PDMA 传输过程中的目标中止中断 0 = 禁用 PDMA 传输过程中的目标中止中断

PDMA通道 x中断状态寄存器(PDMA_ISRx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_ISRx x=0,1 .. 8	PDMA_CHx_BA+0x24	R/W	PDMA通道 x中断状态寄存器	0x0000_0000

31	30	29	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						BLKD_IF	TABORT_IF

位	描述	
[31:2]	**Reserved**	保留
[1]	**BLKD_IF**	**PDMA模块传输完成中断标志** 该位指示 PDMA 已经完成了所有传输。 1 = 完成 0 = 还没完成 写 1 清除该位为 0。
[0]	**TABORT_IF**	**PDMA 读/写 目标中止中断标志** 1 = 收到总线错误应答 0 = 没有收到总线错误应答 软件可写 1 清除该位为 0。 **注意：**该位显示总线主机是否受到 ERROR 响应。如果总线主机收到了 ERROR 响应，则意味着目标中止发生了。PDMA控制器将停止传输和响应该事件到软件，然后进入 IDLE 状态。当目标中止发生，软件必须复位 PDMA，再次传输那些数据。

PDMA共享缓存FIFO 0 寄存器(PDMA_SBUF0_Cx)

寄存器	偏移地址	R/W	描述	复位值
PDMA_SBUF0_Cx x=0,1 .. 8	PDMA_CHx_BA+0x80	R	PDMA共享缓存FIFO 0寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PDMA_SBUF0 [31:24]							
23	22	21	20	19	18	17	16
PDMA_SBUF0 [23:16]							
15	14	13	12	11	10	9	8
PDMA_SBUF0 [15:8]							
7	6	5	4	3	2	1	0
PDMA_SBUF0 [7:0]							

位	描述	
[31:0]	PDMA_SBUF0	PDMA 共享缓存 FIFO 0 （只读） 每通道拥有自己的 1 字大小的内部缓存。

CRC控制寄存器(CRC_CTL)

寄存器	偏移地址	R/W	描述	复位值
CRC_CTL	CRC_BA+0x00	R/W	CRC控制寄存器	0x2000_0000

31	30	29	28	27	26	25	24
CRC_MODE		CPU_WDLEN		CHECKSUM_COM	WDATA_COM	CHECKSUM_RVS	WDATA_RVS
23	22	21	20	19	18	17	16
TRIG_EN	Reserved						
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						CRC_RST	CRCCEN

位	描述	
[31:30]	**CRC_MODE**	**CRC多项式模式** 该域表示 CRC操作多项式模式. 00 = CRC-CCITT多项式模式 01 = CRC-8多项式模式 10 = CRC-16多项式模式 11 = CRC-32多项式模式
[29:28]	**CPU_WDLEN**	**CPU写数据长度** 当工作在CPU PIO 模式， 该域表示写数据的长度 00 =写数据长度是 8 bit 模式 01 =写数据长度是16 bit 模式 10 =写数据长度是32 bit 模式 11 =保留 **注意**1: 该域仅用在 CPU PIO 模式. **注意**2: 当数据长度是8 bit 模式，有效数据是 CRC_WDATA [7:0]，如果数据长度是16 bit 模式， 有效数据是 CRC_WDATA [15:0].
[27]	**CHECKSUM_COM**	**校验和一次补码** 该位用来使能CRC_CHECKSUM 寄存器中校验和结果一次补码功能 1 = CRC 校验和一次补码使能 0 = CRC 校验和一次补码禁止
[26]	**WDATA_COM**	**写数据一次补码** 该位用来使能CRC_WDTAT 寄存器中写数据值的一次补码功能 1 = CRC 写数据一次补码使能 0 = CRC写数据一次补码禁止

[25]	**CHECKSUM_RVS**	**校验和取反** 该位用来使能CRC_CHECKSUM 寄存器中校验和结果的位顺序倒转功能 1 = CRC 校验和位顺序倒转 0 = CRC 校验和位顺序不倒转 **注意**:如果校验和数据是 0XDD7B0F2E，位顺序颠倒后是 0x74F0DEBB
[24]	**WDATA_RVS**	**写数据顺序倒转** 该位用来使能CRC_WDTAT 寄存器中写数据值的位顺序倒转功能 1 = CRC写数据位顺序倒转(每个字节) 0 = CRC写数据位顺序不倒转. **注意**: 如果写数据是 0xAABBCCDD, 位倒转后 CRC 写数据是0x55DD33BB
[23]	**TRIG_EN**	**触发使能** 该位用来触发 CRC DMA 传输. 1 = 使能 CRC DMA数据读或写传输. 0 = 没影响 **注意**1: 如果该位触发，表示 CRC 引擎工作在 CRC DMA 模式，所以不要填任何数载在CRC_WDATA 寄存器. **注意**2: 当 CRC DMA 传输完成，该位会被自动清零. **注意**3: 如果总线错误发生，所有CRC DMA传输会停止。软件必须复位所有 DMA 通道，然后再触发。
[22:2]	**Reserved**	保留.
[1]	**CRC_RST**	**CRC引擎复位** 0 = 无效. 1 = 复位内部CRC状态机和内部缓存，控制寄存器的内容不会被清除，该位自动清0。 **注意**: 当工作在 CPU PIO 模式，设置该位会重新加载初始的种子值(CRC_SEED register)。
[0]	**CRCCEN**	**CRC 通道使能** 0 = 无效. 1= CRC操作使能 **注意**1：当工作在 CRC DMA 模式 (TRIG_EN (CRC_CTL[23]) = 1)，如果用户清0该位，DMA操作会继续直到所有CRC DMA操作完成，TRIG_EN (CRC_CTL[23])位会不确定直到所有CRC DMA操作完成。 但是在这种情况CRC_DMAISR [BLKD_IF] 标志是不可用的，当TRIG_EN = 0，用户可以通过读CRC_CHECKSUM 寄存器读CRC 结果 注意2：当工作在 CRC DMA模式(TRIG_EN (CRC_CTL[23]) = 1)，如果用户需要立即停止发送，用户可写1到 CRC_RST (CRC_CTL [1])位来停止发送

CRC DMA传输源地址寄存器 (CRC_DMASAR)

寄存器	偏移地址	R/W	描述	复位值
CRC_DMASAR	CRC_BA+0x04	R/W	CRC DMA传输源地址寄存器	0x0000_0000

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

位	描述	
[31:0]	**CRC_DMASAR**	**CRC DMA 传输源地址寄存器** 该域表示一个 32-bit CRC DMA.的源地址 (CRC_DMASAR + CRC_DMABCR) = (CRC_DMACSAR + CRC_DMACBCR) **注意**: 源地址必须字对齐

CRC DMA传输字节计数寄存器 (CRC_DMABCR)

寄存器	偏移地址	R/W	描述	复位值
CRC_DMABCR	CRC_BA+0x0C	R/W	CRC DMA传输字节计数寄存器	0x0000_0000

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

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**CRC_DMABCR**	**CRC DMA传输字节计数寄存器** 该域表示一个16-bit CRC DMA的传输字节计数值 (CRC_DMASAR + CRC_DMABCR) = (CRC_DMACSAR + CRC_DMACBCR)

CRC DMA当前源地址寄存器 (CRC_DMACSAR)

寄存器	偏移地址	R/W	描述	复位值
CRC_DMACSAR	CRC_BA+0x14	R	CRC DMA当前源地址寄存器	0x0000_0000

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

位	描述	
[31:0]	**CRC_DMACSAR**	**CRC DMA 当前源地址寄存器 (只读)** 该域表示CRC DMA 正在传输的源地址 (CRC_DMASAR + CRC_DMABCR) = (CRC_DMACSAR + CRC_DMACBCR)

CRC DMA当前传输字节计数寄存器(CRC_DMACBCR)

寄存器	偏移地址	R/W	描述	复位值
CRC_DMACBCR	CRC_BA+0x1C	R	CRC DMA当前传输字节计数寄存器	0x0000_0000

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

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**CRC_DMACBCR**	**CRC DMA 当前字节计数寄存器 (只读)** 该域表示CRC_DMA当前剩下的字节计数。 (CRC_DMASAR + CRC_DMABCR) = (CRC_DMACSAR + CRC_DMACBCR) **注意**: CRC_RST置1 会清零该寄存器值.

CRC DMA中断使能控制寄存器(CRC_DMAIER)

寄存器	偏移地址	R/W	描述	复位值
CRC_DMAIER	CRC_BA+0x20	R/W	CRC DMA中断使能控制寄存器	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						**CRC_BLKD_IE**	**CRC_TABORT_IE**

位	描述	
[31:2]	**Reserved**	保留.
[1]	**CRC_BLKD_IE**	**CRC DMA传输完成中断使能** 使能该位，当CRC_BLKD_IF (CRC_DMAISR[1])置1，会产生CRC DMA传输完成中断信号 1 =当 CRC DMA传输完成，使能中断发生器. 0 =当 CRC DMA传输完成，禁止中断发生器.
[0]	**CRC_TABORT_IE**	**CRC DMA读/写目标中止中断使能** 使能该位，当CRC_TARBOT_IF (CRC_DMAISR[0])置1，会产生CRC DMA目标中止中断信号 1 = 当 CRC DMA 传输过程中，使能目标中止中断. 0 =当 CRC DMA 传输过程中，禁止目标中止中断..

CRC DMA中断状态寄存器(CRC_DMAISR)

寄存器	偏移地址	R/W	描述	复位值
CRC_DMAISR	CRC_BA+0x24	R/W	CRC DMA中断状态寄存器	0x0000_0000

31	30	29	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						CRC_BLKD_IF	CRC_TABORT_IF

位	描述	
[31:2]	Reserved	保留
[1]	CRC_BLKD_IF	**CRC DMA块传输结束中断标志** 该位表示CRC DMA 传输是否完成. 1 =如果TRIG_EN (CRC_CTL[23])位使能，CRC传输完成 0 =如果TRIG_EN (CRC_CTL[23])位使能，CRC传输未完成 软件写1清零该位. (当 CRC DMA 传输结束，TRIG_EN (CRC_CTL[23])位会被自动清0)
[0]	CRC_TABORT_IF	**CRC DMA读/写目标中止中断标志** 该位表示在CRC DMA传输期间，CRC总线是否出错。 1 = 传输期间收到总线 错误反应 0 =传输期间 没收到总线错误反应 软件写1清零该位. **注意**：该域表示总线主机是否接收到错误反应，如果总线主机接收到错误反应，意味着目标中止发生，DMA会停止传输，软件对这事件做出反应，然后进入空闲模式。当目标中止发生，软件必须复位DMA，然后再传输那些数据。

CRC写数据寄存器 (CRC_WDATA)

寄存器	偏移地址	R/W	描述	复位值
CRC_WDATA	CRC_BA+0x80	R/W	CRC写数据寄存器	0x0000_0000

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

位	描述	
[31:0]	CRC_WDATA	**CRC写数据寄存器** 当工作在CPU PIO 模式, 软件可以写数据到该域执行 CRC 操作; 当工作在 DMA 模式，该域表示DMA从存储器读数据，不能写。 **注意:** 当写数据长度是8-bit 模式， CRC_WDATA 寄存器中有效的数据仅是CRC_WDATA [7:0] **位**； 当写数据长度是16-bit 模式， CRC_WDATA 寄存器中有效的数据仅是CRC_WDATA [15:0].。

CRC种子寄存器(CRC_SEED)

寄存器	偏移地址	R/W	描述	复位值
CRC_SEED	CRC_BA+0x84	R/W	CRC种子寄存器	0xFFFF_FFFF

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

位	描述	
[31:0]	**CRC_SEED**	**CRC 种子寄存器** 该域表示 CRC种子值.

CRC 校验和寄存器(CRC_CHECKSUM)

寄存器	偏移地址	R/W	描述	复位值
CRC_CHECKSUM	CRC_BA+0x88	R	CRC 校验和寄存器	0xFFFF_FFFF

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

位	描述	
[31:0]	CRC_CHECKSUM	CRC 校验和 寄存器 该域表示CRC 校验和结果。

PDMA全局控制寄存器(PDMA_GCRCSR)

寄存器	偏移地址	R/W	描述	复位值
PDMA_GCRCSR	PDMA_GCR_BA+0x00	R/W	PDMA全局控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							CRC_CLK_EN
23	22	21	20	19	18	17	16
Reserved							CLK8_EN
15	14	13	12	11	10	9	8
CLK7_EN	CLK6_EN	CLK5_EN	CLK4_EN	CLK3_EN	CLK2_EN	CLK1_EN	CLK0_EN
7	6	5	4	3	2	1	0
Reserved							

位	描述	
[31:25]	**Reserved**	保留.
[24]	**CRC_CLK_EN**	**CRC控制器时钟使能控制** 1 = 使能 0 = 禁用
[23:17]	**Reserved**	保留.
[16]	**CLK8_EN**	**DMA 控制器通道 8时钟使能控制** 1 = 使能 0 = 禁用
[15]	**CLK7_EN**	**DMA 控制器通道7 时钟使能控制** 1 = 使能 0 = 禁用
[14]	**CLK6_EN**	**DMA 控制器通道 6时钟使能控制** 1 = 使能 0 = 禁用
[13]	**CLK5_EN**	**DMA 控制器通道 5时钟使能控制** 1 = 使能 0 = 禁用
[12]	**CLK4_EN**	**DMA 控制器通道 4时钟使能控制** 1 = 使能 0 = 禁用
[11]	**CLK3_EN**	**DMA 控制器通道 3时钟使能控制** 1 = 使能 0 = 禁用

[10]	**CLK2_EN**	**DMA 控制器通道 2时钟使能控制** 1 = 使能 0 = 禁用
[9]	**CLK1_EN**	**DMA 控制器通道 1时钟使能控制** 1 = 使能 0 = 禁用
[8]	**CLK0_EN**	**DMA 控制器通道 0时钟使能控制** 1 = 使能 0 = 禁用
[7:0]	**Reserved**	保留

PDMA服务选择控制寄存器 0 (PDMA_PDSSR0)

寄存器	偏移地址	R/W	描述	复位值
PDMA_PDSSR0	PDMA_GCR_BA+0x04	R/W	PDMA服务选择控制寄存器 0	0xFFFF_FFFF

31	30	29	28	27	26	25	24
SPI3_TXSEL				SPI3_RXSEL			
23	22	21	20	19	18	17	16
SPI2_TXSEL				SPI2_RXSEL			
15	14	13	12	11	10	9	8
SPI1_TXSEL				SPI1_RXSEL			
7	6	5	4	3	2	1	0
SPI0_TXSEL				SPI0_RXSEL			

位	描述	
[31:28]	SPI3_TXSEL	**PDMA SPI3 TX 选择** 该域定义了哪一个 PDMA 通道和片上外设 SPI3 TX 相连。软件可以通过 SPI3_TXSEL 配置 TX 通道。该通道配置和 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 域相同，请参考 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 的说明。
[27:24]	SPI3_RXSEL	**PDMA SPI3 RX 选择** 该域定义了哪一个 PDMA 通道和片上外设 SPI3 RX 相连。软件可以通过 SPI3_RXSEL 配置 RX 通道。该通道配置和SPI0_RXSEL (PDMA_PDSSR0[3:0]). 域相同，请参考 SPI0_RXSEL (PDMA_PDSSR0[3:0]).的说明。
[23:20]	SPI2_TXSEL	**PDMA SPI2 TX 选择** 该域定义了哪一个 PDMA 通道和片上外设 SPI2 TX 相连。软件可以通过 SPI2_TXSEL 配置 TX 通道。该通道配置和 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 域相同，请参考 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 的说明。
[19:16]	SPI2_RXSEL	**PDMA SPI2 RX 选择** 该域定义了哪一个 PDMA 通道和片上外设 SPI2 RX 相连。软件可以通过 SPI2_RXSEL 配置 RX 通道。该通道配置和 SPI0_RXSEL PDMA_PDSSR0[3:0]) 域相同，请参考 SPI0_RXSEL PDMA_PDSSR0[3:0])的说明。
[15:12]	SPI1_TXSEL	**PDMA SPI1 TX 选择** 该域定义了哪一个 PDMA 通道和片上外设 SPI1 TX 相连。软件可以通过 SPI1_TXSEL 配置 TX 通道。该通道配置和 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 域相同，请参考 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 的说明。
[11:8]	SPI1_RXSEL	**PDMA SPI1 RX 选择** 该域定义了哪一个 PDMA 通道和片上外设 SPI1 RX 相连。软件可以通过 SPI1_RXSEL 配置 RX 通道。该通道配置和 SPI0_RXSEL (PDMA_PDSSR0[3:0])域相同，请参考 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 的说明。
[7:4]	SPI0_TXSEL	**PDMA SPI0 TX 选择** 该域定义了哪一个 PDMA 通道和片上外设 SPI0 TX 相连。软件可以通过 SPI0_TXSEL 配置 TX 通道。该通道配置和 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 域相同，请参考 SPI0_RXSEL (PDMA_PDSSR0[3:0]) 的说明

[3:0]	**SPI0_RXSEL**	**PDMA SPI0 RX 选择** 该域定义了哪一个 PDMA 通道和片上外设 SPI0 RX 相连。软件可以通过 SPI0_RXSEL 配置 RX 通道。例如：SPI0_RXSEL(PDMA_PDSSR0[3:0]) = 0110，则 SPI0_RX 和 PDMA_CH6 相连。 0000: CH0 0001: CH1 0010: CH2 0011: CH3 0100: CH4 0101: CH5 0110: CH6 0111: CH7 1000: CH8 其他：保留

PDMA服务选择控制寄存器1 (PDMA_PDSSR1)

寄存器	偏移地址	R/W	描述	复位值
PDMA_PDSSR1	PDMA_GCR_BA+0x08	R/W	PDMA服务选择控制寄存器1	0xFFFF_FFFF

31	30	29	28	27	26	25	24
Reserved				ADC_RXSEL			
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
UART1_TXSEL				UART1_RXSEL			
7	6	5	4	3	2	1	0
UART0_TXSEL				UART0_RXSEL			

位	描述	
[31:28]	Reserved	保留
[27:24]	ADC_RXSEL	**PDMA ADC RX 选择** 该域定义了哪一个 PDMA 通道和片上外设 ADC RX 相连。软件可以通过ADC_RXSEL. 配置 RX 通道。该通道配置和 UART0_RXSEL (PDMA_PDSSR1[3:0]) 域相同，请参考 UART0_RXSEL (PDMA_PDSSR1[3:0]) 的说明。
[23:16]	Reserved	保留.
[15:12]	UART1_TXSEL	**PDMA UART1 TX 选择** 该域定义了哪一个 PDMA 通道和片上外设 UART1 TX 相连。软件可以通过 UART1_TXSEL.配置 TX 通道。该通道配置和 UART0_RXSEL(PDMA_PDSSR1[3:0]) 域相同，请参考 UART0_RXSEL(PDMA_PDSSR1[3:0]) 的说明。
[11:8]	UART1_RXSEL	**PDMA UART1 RX 选择** 该域定义了哪一个 PDMA 通道和片上外设 UART1 RX 相连。软件可以通过 UART1_RXSEL 配置 RX 通道。该通道配置和 UART0_RXSEL (PDMA_PDSSR1[3:0])域相同，请参考 UART0_RXSEL (PDMA_PDSSR1[3:0])的说明。
[7:4]	UART0_TXSEL	**PDMA UART0 TX 选择** 该域定义了哪一个 PDMA 通道和片上外设 UART0 TX 相连。软件可以通过 UART0_TXSEL 配置 TX 通道。该通道配置和 UART0_RXSEL(PDMA_PDSSR1[3:0])域相同，请参考 UART0_RXSEL(PDMA_PDSSR1[3:0])的说明。

[3:0]	**UART0_RXSEL**	该域定义了哪一个 PDMA 通道和片上外设 UART0 RX 相连。软件可以通过 UART0_RXSEL 配置 RX 通道。例如：UART0_RXSEL(PDMA_PDSSR1[3:0])= 0110，则 UART0_RX 和 PDMA_CH6 相连。 0000: CH0 0001: CH1 0010: CH2 0011: CH3 0100: CH4 0101: CH5 0110: CH6 0111: CH7 1000: CH8 其他：保留

PDMA 全局中断状态寄存器 (PDMA_GCRISR)

寄存器	偏移地址	R/W	描述	复位值
PDMA_GCRISR	PDMA_GCR_BA+0x0C	R	PDMA 全局中断状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
INTR	Reserved						
23	22	21	20	19	18	17	16
Reserved							INTRCRC
15	14	13	12	11	10	9	8
Reserved							INTR8
7	6	5	4	3	2	1	0
INTR7	INTR6	INTR5	INTR4	INTR3	INTR2	INTR1	INTR0

位	描述	
[31]	**INTR**	**中断状态** 该位是 PDMA 控制器的中断状态。 **注意：**该位只读
[30:17]	**Reserved**	保留.
[16]	**INTRCRC**	**CRC 控制器的中断状态** 该位是CRC控制器的中断状态 **注意：**该位只读
[15:9]	**Reserved**	保留.
[8]	**INTR8**	**通道 8 中断状态** 该位是 PDMA 通道 8 的中断状态。 **注意：**该位只读
[7]	**INTR7**	**通道 7 中断状态** 该位是 PDMA 通道 7 的中断状态。 **注意：**该位只读
[6]	**INTR6**	**通道 6 中断状态** 该位是 PDMA 通道 6 的中断状态。 **注意：**该位只读
[5]	**INTR5**	**通道 5中断状态** 该位是 PDMA 通道 5的中断状态。 **注意：**该位只读
[4]	**INTR4**	**通道4 中断状态** 该位是 PDMA 通道 4 的中断状态。 **注意：**该位只读

[3]	**INTR3**	**通道 3中断状态** 该位是 PDMA 通道 3的中断状态。 **注意：**该位只读
[2]	**INTR2**	**通道 2中断状态** 该位是 PDMA 通道 2的中断状态。 **注意：**该位只读
[1]	**INTR1**	**通道 1中断状态** 该位是 PDMA 通道 1 的中断状态。 **注意：**该位只读
[0]	**INTR0**	**通道 0 中断状态** 该位是 PDMA 通道 0的中断状态。 **注意：**该位只读

PDMA服务选择控制寄存器2 (PDMA_PDSSR2)

寄存器	偏移地址	R/W	描述	复位值
PDMA_PDSSR2	PDMA_GCR_BA+0x10	R/W	PDMA服务选择控制寄存器2	0x0000_00FF

31	30	29	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**
I2S_TXSEL				**I2S_RXSEL**			

位	描述	
[31:8]	**Reserved**	保留
[7:4]	**I2S_TXSEL**	**PDMA I²S TX 选择** 该域定义了哪一个 PDMA 通道和片上外设 I²S TX 相连。软件可以通过 I2S_TXSEL 配置 TX 通道。该通道配置和 I2S_RXSEL(PDMA_PDSSR2[3:0]) 域相同，请参考 I2S_RXSEL(PDMA_PDSSR2[3:0]) 的说明。
[3:0]	**I2S_RXSEL**	**PDMA I²S RX 选择** 该域定义了哪一个 PDMA 通道和片上外设 I²S RX 相连。软件可以通过 I2S_RXSEL 配置 RX 通道。例如：I2S_RXSEL(PDMA_PDSSR2[3:0]) = 0110，则 I2S_RX 和 PDMA_CH6 相连。 0000: CH0 0001: CH1 0010: CH2 0011: CH3 0100: CH4 0101: CH5 0110: CH6 0111: CH7 1000: CH8 其他：保留

5.8 定时器控制器(TIMER)

5.8.1 概述

定时器控制器包含 4 组 32-位定时器，TIMER0~TIMER3，提供用户便捷的计数定时功能。定时器可执行很多功能，如频率测量，时间延迟，外部输入管脚事件计数和外部捕捉管脚脉宽测量等。

5.8.2 特性

- 4 组 32-位定时器，带24位向上定时器和一个8位的预分频计数器
- 每个定时器都有独立的时钟源
- 提供 one-shot, periodic, toggle 和 continuous 计数操作模式
- 超时周期 =（输入的定时器时钟周期）* （8-位预分频计数器 + 1） *（24-位 TCMP）
- 最大计数周期 = (1 / T MHz) * (2^8) * (2^{24})，T 是定时器周期
- 通过 TDR （定时器数据寄存器）可读取内部 24 位向上计数器的值
- 支持事件计数功能可用于计数外部管脚的事件(TM0~TM3)
- 支持外部管脚捕捉(TM0_EXT~TM3_EXT)，可用于脉宽测量
- 支持外部引脚捕捉(TM0_EXT~TM3_EXT)，可用于复位24位向上定时器
- 如果定时器中断信号产生，支持芯片从空闲/掉电模式唤醒

5.8.3 框图.

定时器控制器的框图和时钟控制如下所示：

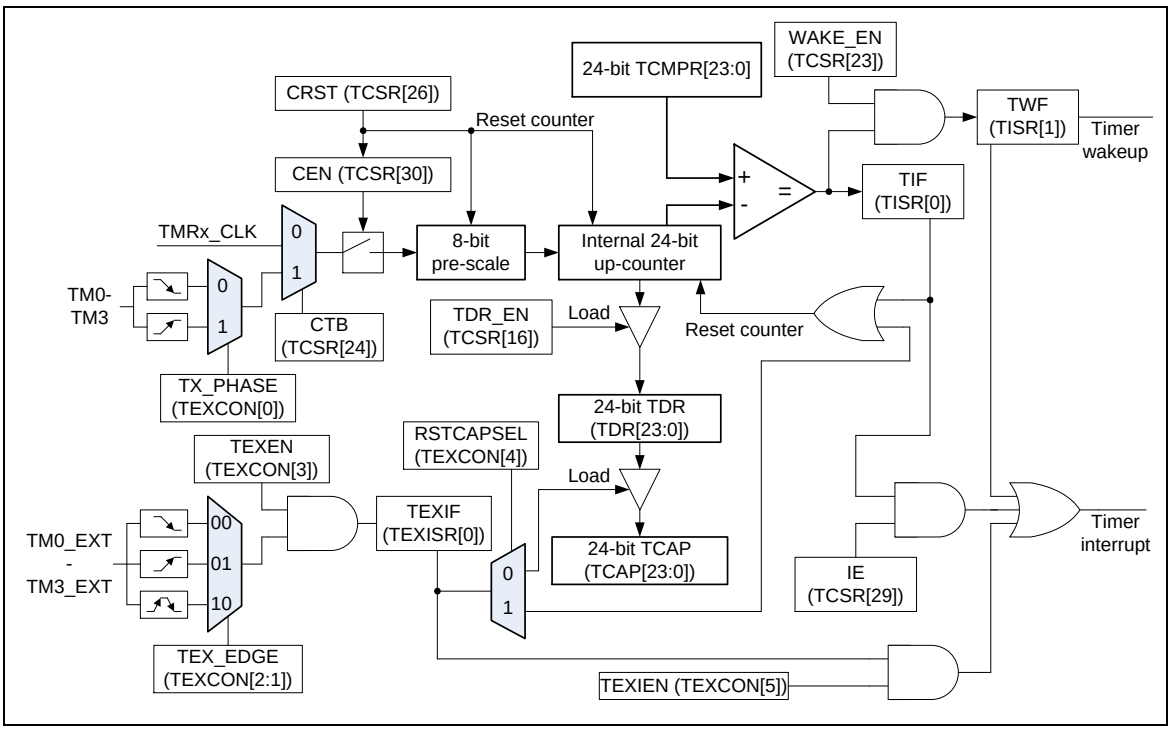


图 5-27 定时器控制器框图

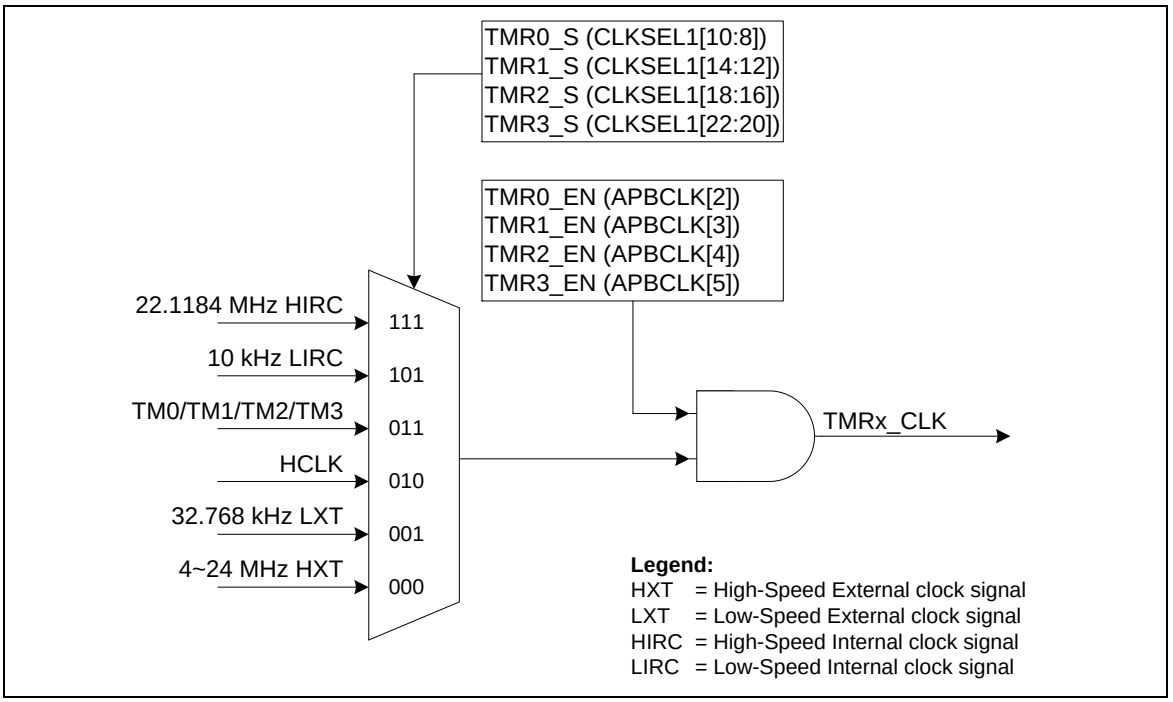


图 5-28 定时器控制器时钟源

5.8.4 基本配置

定时器0~定时器3的外围时钟源可以在寄存器APBCLK[5:2]使能和在寄存器CLKSEL1[10:8](定时器0)，CLKSEL1[14:12](定时器1)，CLKSEL1[18:16](定时器2)，CLKSEL1[22:20](定时器3)选择不同时钟。

5.8.5 功能描述

5.8.5.1 *定时器中断标志*

定时器控制器支持两个中断标志：一个是TIF标志，该标志在当定时器计数器值(TDR)与定时器比较值(TCMP)相匹配时置位，另一个是TEXIF标志，该标志在当TMx_EXT管脚的变化与TEX_EDGE的设置一致时置位。

5.8.5.2 *One–shot 模式*

如果定时器工作在单周期 (one-shot) 模式(TCSR[28:27]为00)且 CEN (TCSR[30] 定时器使能位)置1，则定时器的计数器开始计数。一旦定时器计数器的值达到定时器比较器寄存器 (TCMP) 的值时，TIF标志将变为1，TDR的值和CEN位将由定时器控制器清零，然后定时器计数操作停止。与此同时，如果 IE（TCSR[29] 中断使能位）使能，则定时器中断信号产生并送到 NVIC 通知 CPU 。

5.8.5.3 *Periodic 模式*

如果定时器工作在周期 (periodic) 模式(TCSR[28:27]为01)且 CEN (TCSR[30] 定时器使能位)置1，则定时器的计数器开始计数。一旦定时器计数器的值达到定时器比较器寄存器 (TCMP) 的值时，TIF标志将变为1，TDR的值将由定时器控制器清零，然后定时器重新计数。与此同时，如果 IE（TCSR[29] 中断使能位）使能，则定时器中断信号产生并送到 NVIC 通知 CPU 。在该模式，定时器控制器周期性地操作计数和 与TCMPR 的值比较，直到 CEN位由软件清0。

5.8.5.4 *Toggle-output 模式*

如果定时器工作在触发输出 (toggle-out) 模式(TCSR[28:27]为10)且 CEN (TCSR[30] 定时器使能位)置1，则定时器的计数器开始计数。toggle-out 模式的计数操做大部分与周期模式是一样的，除了该模式当TIF位设置时，有相关的TM0~TM3管脚来输出信号，因此，管脚TM0~TM3上的触发输出信号以 50% 的占空周期反复改变.

5.8.5.5 *Continuous Counting 模式*

如果定时器工作在连续计数 (continuous counting) 模式(TCSR[28:27]为11)且 CEN (TCSR[30] 定时器使能位)置1，则定时器的计数器开始计数。一旦定时器计数器(TDR)的值达到定时器比较器寄存器 (TCMP) 的值时，TIF标志将变为1，但TDR的值继续保持向上计数。与此同时，如果 IE（TCSR[29] 中断使能位）使能，则定时器中断信号产生并送到 NVIC 通知 CPU 。在该模式，用户可以立刻改变不同的TCMP值，而不需要停止定时器计数和重新开始定时器计数。

例如，先把定时器比较寄存器 (TCMP)的值设置为 80。当定时器计数器的值 (TDR 的值) 达到 80 时, TIF标志将被置1，定时器计数器继续计数，而且TDR的值将不回到0，而是继续计数，81, 82, 83,⋯ to $(2^{24}-1)$，然后再一次 0, 1, 2, 3, ⋯ 到 $2^{24}-1$，如此往复。再次，如果软件改变TCMP的值为200 并且清除TIF标志位，当TDR的值达到200时，TIF标志将再次变为1,。最后，软件改变TCMP的值为500并且清除TIF标志，当TDR的值达到500时，TIF标志将再次变为1

在该模式，计数器计数时连续的。所以该操作模式叫做连续计数模式。

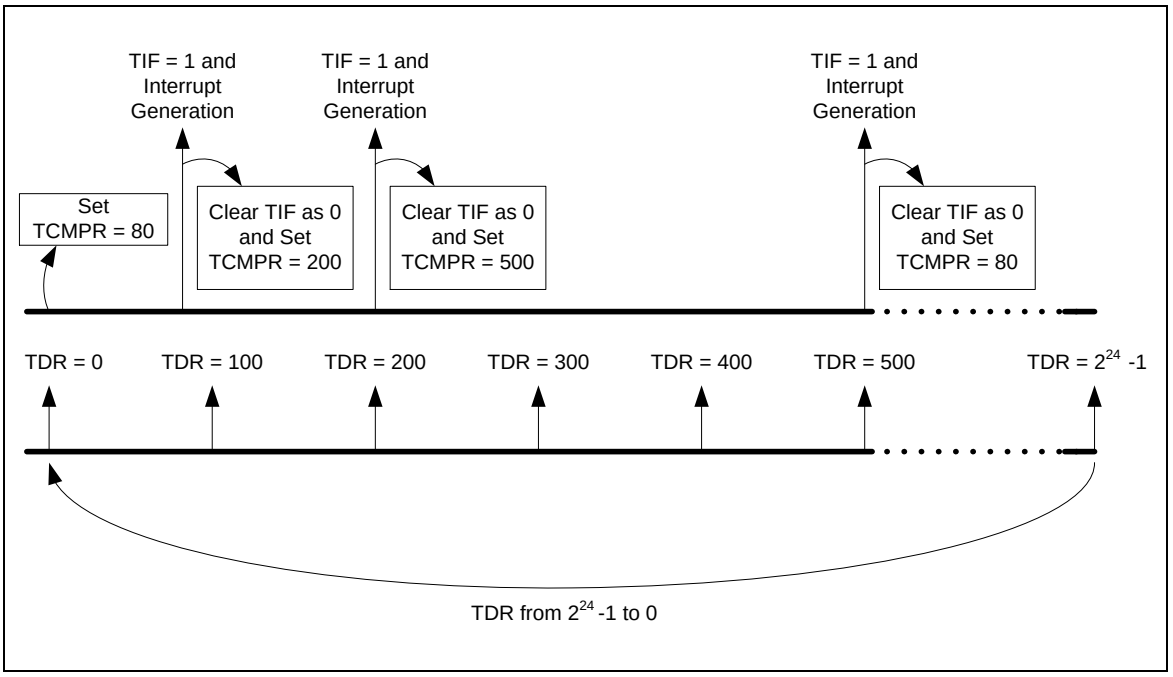


图 5-29 连续计数模式

5.8.5.6 *事件计数模式*

定时器控制器也提供这样的应用，能对输入事件(来自管脚 TMx x=0~3)计数并将事件的次数反应到TDR的值。也可以称为事件计数功能。要使用该功能，CTB(TCSR[24])位需置位并且定时器外设时钟源必须设为HCLK。

软件可以通过TCDB(TEXCON[7])位来使能或关闭TMx管脚消抖电路。如果TMx管脚的消抖电路关闭，输入事件频率必须少于1/3HCLK，如果消抖电路打开，输入事件的频率须小于1/8HCLK，以保证TDR的值是正确的。软件也可以通过设置TX_PHASE(TEXCON[0])来选择边沿检测TMx管脚的相位。

在事件计数模式，定时器计数操作模式可以设置为单次，周期，和连续计数模式来计算来自TMx管脚的输入事件TDR的值。

5.8.5.7 *外部捕捉模式*

事件捕捉功能是当检测到TMx_EXT管脚(x=0~3)边沿电平有变化时，定时器计数器值(TDR)会送到捕捉数据寄存器(TCAP)。在该模式，需把RSTCAPSEL(TEXCON[4])位设置为0，用来选择TMx_EXT变化时用作事件捕捉功能，而且定时器外设时钟源必须设为HCLK。

软件可以通过TEXDB(TEXCON[6])位来使能或关闭TMx_EXT管脚消抖电路。在TMx_EXT的消抖电路关闭时，TMx_EXT管脚的转变频率必须少于1/3 HCLK，在TMx_EXT的消抖电路打开时，TMx_EXT管脚的转变频率必须少于1/8 HCLK，以保证捕捉功能能够正常工作。软件也可以通过设置TEX_EDGE(TEXCON[2:1])位来选择TMx_EXT管脚的边沿转变检测方式。

在事件捕捉模式，软件不用考虑定时器计数器工作模式的选择，捕捉事件的发生只有当检测到TMx_EXT管脚有边沿变化时。

5.8.5.8 *事件复位计数模式*

当检测到TMx_EXT管脚(x=0~3)有边沿转变时，定时器同样提供事件复位计数器功能来复位TDR的值。在该模式，大部分设置与事件捕捉功能相同，除了RSTCAPSEL(TEXCON[4])位必须设置为1来选择TMx_EXT转变时用作为事件复位计数器。

5.8.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
TIMER基地址: **TMR01_BA = 0x4001_0000** **TMR23_BA = 0x4011_0000**				
TCSR0	TMR01_BA+0x00	R/W	Timer0 控制和状态寄存器	0x0000_0005
TCMPR0	TMR01_BA+0x04	R/W	Timer0 比较寄存器	0x0000_0000
TISR0	TMR01_BA+0x08	R/W	Timer0 中断状态寄存器	0x0000_0000
TDR0	TMR01_BA+0x0C	R	Timer0 数据寄存器	0x0000_0000
TCAP0	TMR01_BA+0x10	R	Timer0 捕捉数据寄存器	0x0000_0000
TEXCON0	TMR01_BA+0x14	R/W	Timer0 外部控制寄存器	0x0000_0000
TEXISR0	TMR01_BA+0x18	R/W	Timer0 外部中断状态寄存器	0x0000_0000
TCSR1	TMR01_BA+0x20	R/W	Timer1 控制和状态寄存器	0x0000_0005
TCMPR1	TMR01_BA+0x24	R/W	Timer1 比较寄存器	0x0000_0000
TISR1	TMR01_BA+0x28	R/W	Timer1 中断状态寄存器	0x0000_0000
TDR1	TMR01_BA+0x2C	R	Timer1 数据寄存器	0x0000_0000
TCAP1	TMR01_BA+0x30	R	Timer1 捕捉数据寄存器	0x0000_0000
TEXCON1	TMR01_BA+0x34	R/W	Timer1 外部控制寄存器	0x0000_0000
TEXISR1	TMR01_BA+0x38	R/W	Timer1 外部中断状态寄存器	0x0000_0000
TCSR2	TMR23_BA+0x00	R/W	Timer2 控制和状态寄存器	0x0000_0005
TCMPR2	TMR23_BA+0x04	R/W	Timer2 比较寄存器	0x0000_0000
TISR2	TMR23_BA+0x08	R/W	Timer2 中断和状态寄存器	0x0000_0000
TDR2	TMR23_BA+0x0C	R	Timer2 数据寄存器	0x0000_0000
TCAP2	TMR23_BA+0x10	R	Timer2 捕捉数据寄存器	0x0000_0000
TEXCON2	TMR23_BA+0x14	R/W	Timer2 外部控制寄存器	0x0000_0000
TEXISR2	TMR23_BA+0x18	R/W	Timer2 外部中断状态寄存器	0x0000_0000
TCSR3	TMR23_BA+0x20	R/W	Timer3 控制和状态寄存器	0x0000_0005
TCMPR3	TMR23_BA+0x24	R/W	Timer3 比较寄存器	0x0000_0000
TISR3	TMR23_BA+0x28	R/W	Timer3 中断状态寄存器	0x0000_0000
TDR3	TMR23_BA+0x2C	R	Timer3 数据寄存器	0x0000_0000
TCAP3	TMR23_BA+0x30	R	Timer3 捕捉数据寄存器	0x0000_0000

TEXCON3	TMR23_BA+0x34	R/W	Timer3 外部控制寄存器	0x0000_0000
TEXISR3	TMR23_BA+0x38	R/W	Timer3 外部中断状态寄存器	0x0000_0000

5.8.7 寄存器描述

定时器控制寄存器(TCSR)

寄存器	偏移地址	R/W	描述	复位值
TCSR0	TMR01_BA+0x00	R/W	Timer0 控制和状态寄存器	0x0000_0005
TCSR1	TMR01_BA+0x20	R/W	Timer1 控制和状态寄存器	0x0000_0005
TCSR2	TMR23_BA+0x00	R/W	Timer2 控制和状态寄存器	0x0000_0005
TCSR3	TMR23_BA+0x20	R/W	Timer3 控制和状态寄存器	0x0000_0005

31	30	29	28	27	26	25	24
DBGACK_TMR	CEN	IE	MODE[1:0]		CRST	CACT	CTB
23	22	21	20	19	18	17	16
WAKE_EN	Reserved						TDR_EN
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PRESCALE[7:0]							

位	描述	
[31]	DBGACK_TMR	**仿真器 (ICE) 调试模式响应禁止位 （写保护位）** 0 = 仿真器调试模式响应影响定时器计数。 当调试器调试模式响应时，定时器计数器将被固定住。 1 = 仿真器调试模式响应禁用。 无论调试器调试模式响应与否，定时器计数器将持续计数下去。
[30]	CEN	**定时器使能位** 0 = 停止/暂停计数 1 = 开始计数 **注意**1：在停止状态，设置 CEN 为 1 将使能 24-位向上计数器从上次停止的计数值继续计数。 **注意**2：在 one-shot 模式下 (MODE [28:27] =00)，当相应的定时中断标志(TISR[0] TIF)产生时，该位由硬件自动清零。
[29]	IE	**中断使能位** 0 = 禁用定时器中断 1 = 使能定时器中断 如果该位使能，当定时器中断标志 (TISR[0] TIF) 置 1， 定时器中断信号产生并通知CPU。

[28:27]	**MODE**	**定时器操作模式** 00 = 定时器工作在单触发模式 (one-shot) 01 = 定时器工作在周期模式 (period) 10 = 定时器工作在Toggle 模式 11 = 定时器工作在连续计数模式(Continuous Counting）
[26]	**CRST**	**定时器复位** 0 = 该位写 0 无效 1 = 复位定时器的8-位预分频计数器，内部24-位向上计数器的值和CEN位如果CACT为1
[25]	**CACT**	**定时器激活状态位（只读）** 该位表示当前定时器计数器的状态。 0 = 定时器计数器未激活 1 = 定时器计数器激活
[24]	**CTB**	**计数模式使能位** 该位用来使能外部计数管脚功能。当定时器用作事件计数，该位需要设置成1.并选择HCLK 作为定时器时钟源。 0 = 禁用计数器模式 1 = 使能计数器模式 详细内容请参考6.17.5.6章节
[23]	**WAKE_EN**	**唤醒使能** 0 = 唤醒触发事件禁止 1 = 唤醒触发事件使能
[22:17]	**Reserved**	保留
[16]	**TDR_EN**	**数据加载使能** 当该位置1，计数器正在计数时，定时器计数器值(TDR) 会不断更新来监视内部24位向上计数器值。 0 = 定时器数据寄存器更新禁止 1 = 当定时器计数器激活， 定时器数据寄存器更新使能
[15:8]	**Reserved**	保留
[7:0]	**PRESCALE**	**预分频计数器** 时钟输入根据 PRESCALE +1 进行预分频。如果 PRESCALE 数值为0，不进行预分频。

定时器比较寄存器(TCMPR)

寄存器	偏移地址	R/W	描述	复位值
TCMPR0	TMR01_BA+0x04	R/W	Timer0 比较寄存器	0x0000_0000
TCMPR1	TMR01_BA+0x24	R/W	Timer1 比较寄存器	0x0000_0000
TCMPR2	TMR23_BA+0x04	R/W	Timer2 比较寄存器	0x0000_0000
TCMPR3	TMR23_BA+0x24	R/W	Timer3 比较寄存器	0x0000_0000

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

位	描述	
[31:24]	**Reserved**	保留
[23:0]	**TCMP**	**定时器比较值** TCMP 是24-位比较寄存器。当内部 24-位向上计数器的值与 TCMP 的值相等时，TIF标志将置1。 超时周期 = (定时器时钟输入周期) * (8-bit PRESCALE + 1) * (24-bit TCMP) **注意**1：不能向 TCMP 里写 0x0 或 0x1，否则内核将运行到未知状态。 **注意**2：当定时器工作在 continuous counting 模式时，如果软件写一个新的值到 TCMP，24-位向上计数定时器将继续计数。如果定时器工作在其他模式，如果软件写一个新的值到 TCMP，定时器将使用新比较值并退出当前计数，开始重新计数。

定时器中断状态寄存器(TISR)

寄存器	偏移地址	R/W	描述	复位值
TISR0	TMR01_BA+0x08	R/W	Timer0 中断状态寄存器	0x0000_0000
TISR1	TMR01_BA+0x28	R/W	Timer1 中断状态寄存器	0x0000_0000
TISR2	TMR23_BA+0x08	R/W	Timer2 中断状态寄存器	0x0000_0000
TISR3	TMR23_BA+0x28	R/W	Timer3 中断状态寄存器	0x0000_0000

31	30	29	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						**TWF**	**TIF**

位	描述	
[31:2]	**Reserved**	保留
[1]	**TWF**	**定时器唤醒标志** 该位表示定时器中断唤醒标志状态。 0 = 定时器不会引起CPU 唤醒 1 =如果定时器中断信号产生， CPU 从空闲或掉电模式唤醒 该位必须通过软件写1清0.
[0]	**TIF**	**定时器中断标志** 当内部 24-位向上计数定时器与定时器比较值 (TCMP)匹配时，定时器中断状态标志位。. 0 = 无影响 1 = 计数定时器与TCMP的值相匹配 该位写 1 清零。.

定时器数据寄存器(TDR)

寄存器	偏移地址	R/W	描述	复位值
TDR0	TMR01_BA+0x0C	R	Timer0 数据寄存器	0x0000_0000
TDR1	TMR01_BA+0x2C	R	Timer1 数据寄存器	0x0000_0000
TDR2	TMR23_BA+0x0C	R	Timer2 数据寄存器	0x0000_0000
TDR3	TMR23_BA+0x2C	R	Timer3 数据寄存器	0x0000_0000

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

位	描述	
[31:24]	**Reserved**	保留.
[23:0]	**TDR**	**定时器数据寄存器** 如果 TDR_EN (TCSR[16]) 置 1, 定时器计数器值(TDR) 会不断更新来监视内部24位向上计数器值。

定时器捕捉数据寄存器(TCAP)

寄存器	偏移地址	R/W	描述	复位值
TCAP0	TMR01_BA+0x10	R	Timer0 捕捉数据寄存器	0x0000_0000
TCAP1	TMR01_BA+0x30	R	Timer1 捕捉数据寄存器	0x0000_0000
TCAP2	TMR23_BA+0x10	R	Timer2 捕捉数据寄存器	0x0000_0000
TCAP3	TMR23_BA+0x30	R	Timer3 捕捉数据寄存器	0x0000_0000

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

位	描述	
[31:24]	**Reserved**	保留
[23:0]	**TCAP**	**定时器捕捉数据寄存器** 当 TEXIF (TEXISR[0])标志和RSTCAPSEL (TEXCON[4]) 标志被置1时，当前内部24-位向上计数定时器的值将立刻自动被载入到 TCAP域。

定时器外部控制寄存器 (TEXCON)

寄存器	偏移地址	R/W	描述	复位值
TEXCON0	TMR01_BA+0x14	R/W	Timer0 外部控制寄存器	0x0000_0000
TEXCON1	TMR01_BA+0x34	R/W	Timer1 外部控制寄存器	0x0000_0000
TEXCON2	TMR23_BA+0x14	R/W	Timer2 外部控制寄存器	0x0000_0000
TEXCON3	TMR23_BA+0x34	R/W	Timer3 外部控制寄存器	0x0000_0000

31	30	29	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**
TCDB	**TEXDB**	**TEXIEN**	**RSTCAPSEL**	**TEXEN**	**TEX_EDGE**		**TX_PHASE**

位	描述	
[31:8]	**Reserved**	保留
[7]	**TCDB**	**定时器外部计数器输入管脚防抖动使能位** 0 = 禁用防抖动 1 = 使能防抖动 如果该位使能，TMx管脚边沿检测带防抖动电路。
[6]	**TEXDB**	**定时器外部捕捉输入管脚防抖动使能位** 0 = TMx_EXT 管脚禁用防抖动 1 = TMx_EXT 管脚使能防抖动 如果该位使能，TMx_EXT 管脚边沿检测带防抖动电路。
[5]	**TEXIEN**	**定时器外部捕捉中断使能位** 0 = TMx_EXT 管脚检测中断禁止. 1 = TMx_EXT 管脚检测中断使能 如果TEXIEN 使能，当TEXIF 标志设为1时，定时器会产生一个外部捕捉中断信号并通知CPU发生了中断。.
[4]	**RSTCAPSEL**	**定时器外部复位计数器/ 定时器外部捕捉模式选择** 0 = TMx_EXT 管脚上的转变用来保存24位定时器计数器(TDR)值到定时器捕捉值寄存器(TCAP)上(事件捕捉功能) 1 = TMx_EXT管脚上的转变用来复位24位定时器计数器(事件复位计数器功能)
[3]	**TEXEN**	**定时器外部管脚使能使能位** 该位使能TMx_EXT管脚上的RSTCAPSEL功能。

		0 = TMx_EXT 管脚上的RSTCAPSEL功能忽略。 1 = TMx_EXT 管脚上的RSTCAPSEL功能激活。 .
[2:1]	**TEX_EDGE**	**定时器外部捕捉管脚边沿检测选择** 00 = TMx_EXT管脚上 1 到 0的转变将被检测。 01 = TMx_EXT管脚上 0 到 1的转变将被检测。 10 = TMx_EXT管脚上 1 到 0 或 0 到 1的转变将被检测。 11 = 保留。
[0]	**TX_PHASE**	**定时器外部计数管脚相位检测选择** 该位表示TMx_EXT管脚相位检测。 0 = TMx_EXT管脚的下降沿将被统计。 1 = TMx_EXT管脚的上升沿将被统计。

定时器外部中断状态寄存器 (TEXISR)

寄存器	偏移地址	R/W	描述	复位值
TEXISR0	TMR01_BA+0x18	R/W	Timer0 外部中断状态寄存器	0x0000_0000
TEXISR1	TMR01_BA+0x38	R/W	Timer1 外部中断状态寄存器	0x0000_0000
TEXISR2	TMR23_BA+0x18	R/W	Timer2 外部中断状态寄存器	0x0000_0000
TEXISR3	TMR23_BA+0x38	R/W	Timer3 外部中断状态寄存器	0x0000_0000

31	30	29	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							**TEXIF**

位	描述	
[31:1]	**Reserved**	保留
[0]	**TEXIF**	**定时器外部中断标志位** 该位表示外部捕捉中断标志状态。. 当TEXEN (TEXCON[3]使能，且TMx_EXT管脚选择为外部捕捉功能，在TMx_EXT管脚边沿转变与TEX_EDGE (TEXCON[2:1]) 设定相匹配时，该标志将由硬件置位。 0 = TMx_EXT管脚中断没有发生 1 = TMx_EXT管脚中断发生 **注意**：该位写 1 清零。

5.9 PWM 发生器和捕捉定时器 (PWM)

5.9.1 简介

NuMicro™ NUC200系列有2组PWM，每组有4个PWM，他们可以配置成8个独立的PWM：PWM0~PWM7 ，或配置成4对PWM:(PWM0, PWM1), (PWM2, PWM3), (PWM4, PWM5) 和(PWM6, PWM7)，4对PWM具有可编程的死区设定.

每个PWM发生器有一个8位预分频器，一个时钟除频提供5级时钟除频(1, 1/2, 1/4, 1/8, 1/16)，两个PWM定时器包括两个时钟选择，两个16位PWM计数器用于PWM周期控制，两个16位比较器用于PWM占空比控制以及一个死区产生器。4对PWM发生器提供8个独立的PWM中断标志，当PWM 向下计数周期达到零时触发中断（硬体来完成），每个PWM中断源有独立的使能位。PWM 发生器可以配置为单触发模式或连续输出PWM波形的模式（自动重载）。

当DZEN01（PCR[4]）置位, PWM0 与PWM1 形成互补的PWM 周期，这一对PWM 的周期，占空比和死区时间由PWM0 定时器和死区发生器0决定. 同样, PWM 互补对(PWM2, PWM3) 、(PWM4, PWM5)和(PWM6, PWM7)由PWM2、PWM4 、PWM6定时器和死区发生器2、4、6相应地控制。关于PWM 定时器内部结构，参考图6-76和图6-83

为防止PWM 输出抖动不稳定波形，16位向下计数计数器和16位比较器采用双缓存。当用户向计数器/比较器寄存器写入新的值时，只有当计数器下数到0后，新写入的值才会被重新加载到计数器/比较器。双缓存可以避免PWM输出时产生干扰波形.

当16位向下计数计数器达到0时，中断请求产生。如果PWM定时器被定义自动装载模式，当向下计数器达到0时，会自动重新加载PWM计数寄存器(CNRx)的值，并重复地开始递减。如果定时器设为单次触发模式，向下计数器达到0时将停止计数，并产生一个中断请求.

PWM 计数比较器的值用于调制高脉冲的宽度，在下数计数器的值等于比较寄存器的值时，计数器控制逻辑将改变，使得PWM输出变成高电平

PWM 定时器的另一个功能是数字输入捕捉功能。当捕捉功能使能时，PWM 输出引脚被切为输入捕捉模式。捕捉器0和PWM0 共享PWM0 的定时器，捕捉器1和PWM1 使用PWM1 的定时器，以此类推。因此，在使用捕捉功能之前，用户必须先启动PMW 定时器。捕捉功能使能后，当输入端口有上升沿时转变时，PWM 计数器的值被锁存入CRLR ，输入端口有下降沿转变时，PWM 计数器的值被锁存入CFLR 。通过软件设定CRL_IE0(CCR0[1])(使能上升沿锁存中断)和CFL_IE0(CCR0[2]) (使能下降沿锁存中断),可以判定捕捉器通道0产生中断的条件。同样设定CRL_IE1(CCR0[17])和CFL_IE1(CCR0[18])，可以判定捕捉器通道1产生中断的条件。通过设定CCR2的相应的数据位，捕捉通道2和3有同样的功能 。对每组而言 ，每当捕捉器触发中断0/1/2/3 时，PWM 计数器0/1/2/3 的值也会同时被重新加载。

PWM 可以支持的最大的捕捉频率由捕捉中断延迟决定.捕捉中断发生时，软件至少执行以下三步:第一步，读PIIR 获取中断源，第二步，读CRLRx/CFLRx(x=0~3) 获取捕捉的值，第三步写1清PIIR为0. 如果中断延迟花T0完成，捕捉信号在（T0）间隔内一定不能改变. 此条件下，最大捕捉频率为1/T0. 例如：

HCLK = 50 MHz, PWM_CLK = 25 MHz, 中断处理时间为900 ns

因此最大捕捉频率为1/900ns ≈ 1000 kHz

5.9.2 特性

5.9.2.1 *PWM 功能:*

- 两组PWM(PWMA/PWMB) 支持 8 个PWM 通道或4对互补的PWM

- PWM 组有两个PWM发生器。每个PWM支持8-位预分频器，两个时钟除频器，两个PWM 定时器（向下计数），一个死区发生器和两个PWM 输出。
- 最高16位解析度
- PWM中断与PWM周期同步
- 单次或自动装载模式
- 边沿对齐或中心对齐两种选择
- PWM触发ADC转换功能

5.9.2.2 *捕捉功能:*

- 与PWM产生器共用时序逻辑控制
- 支持8个捕捉输入通道，与8个PWM输出通道共享
- 每个通道支持1个上升沿锁存寄存器(CRLR)，一个下降沿锁存寄存器(CFLR) 和捕捉中断标志(CAPIFx)

5.9.3 框图

图 5-30到图 5-37 说明了PWM 对的结构(PWM-Timer 0&1 为一对，PWM-Timer 2&3 为一对)

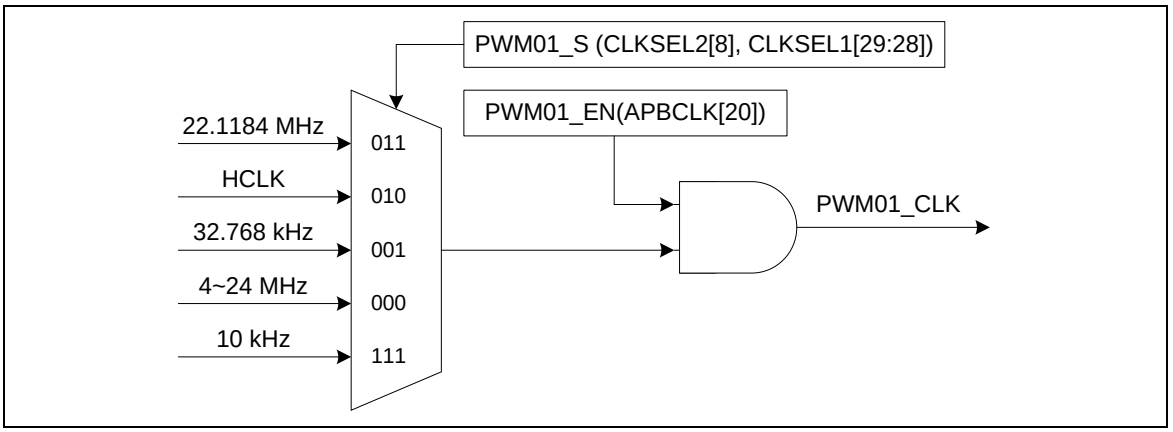


图 5-30 PWM 发生器 0 时钟源控制

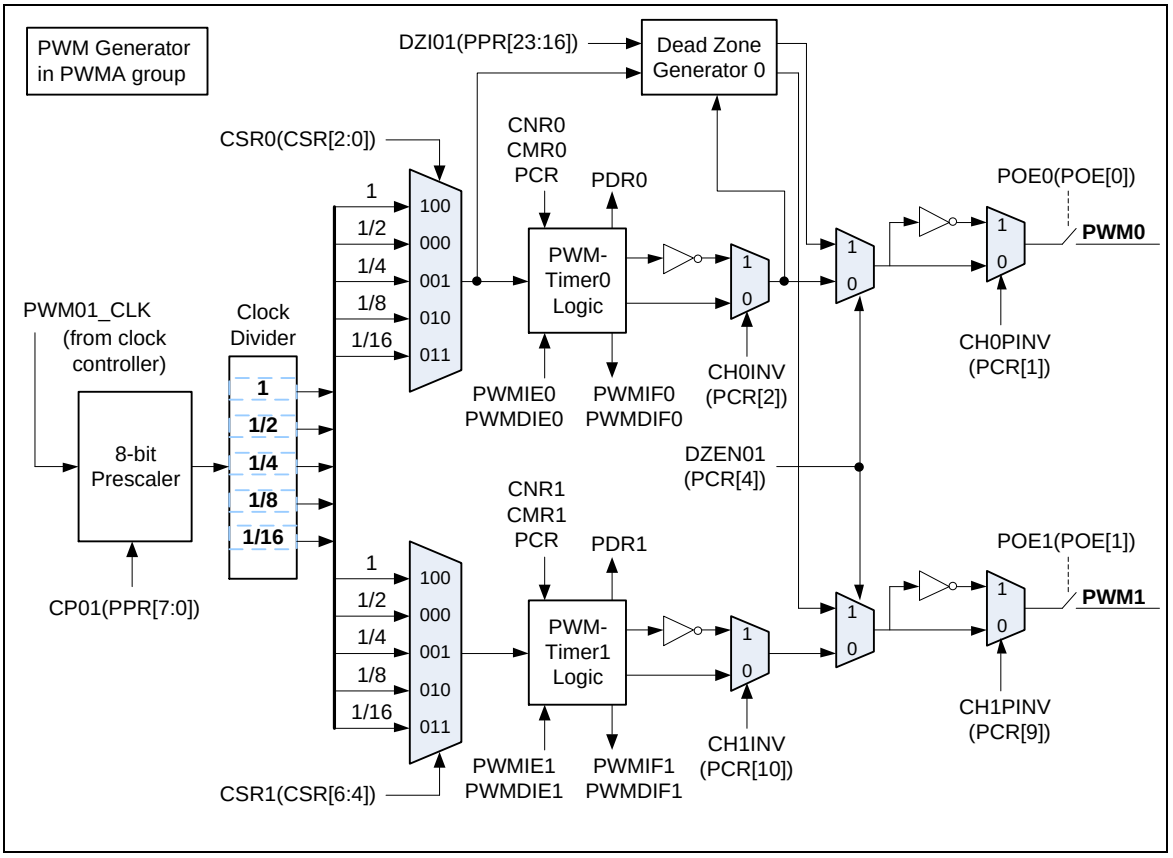


图 5-31 PWM 发生器 0 结构框图

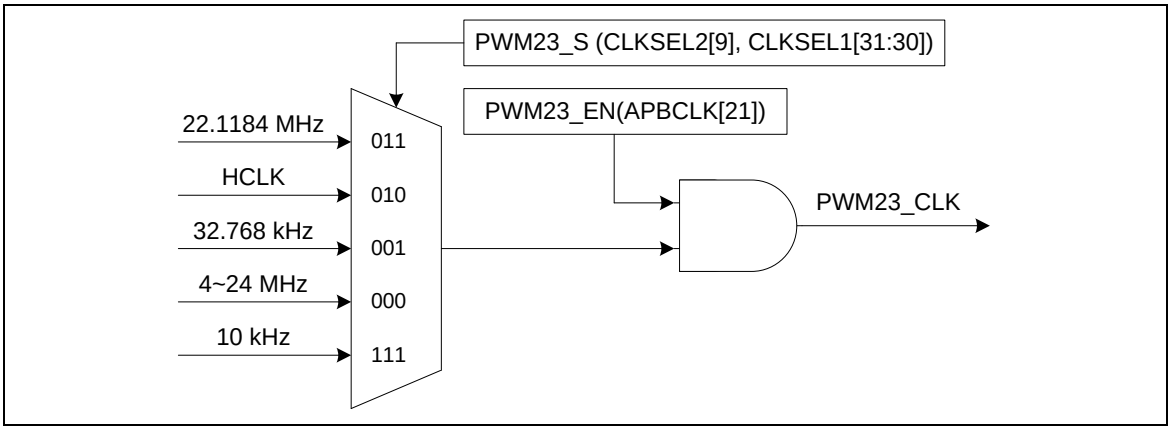


图 5-32 PWM 发生器 2 时钟源控制

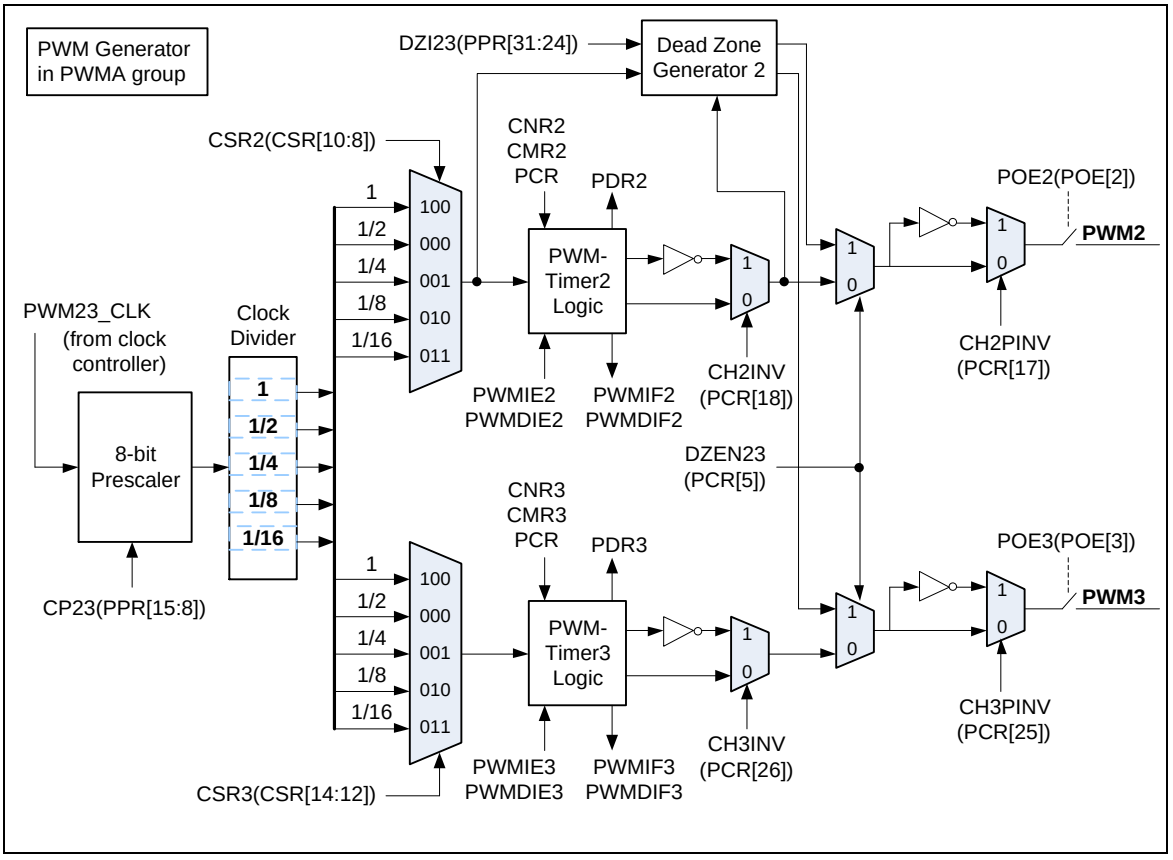


图 5-33 PWM 发生器 2 结构框图

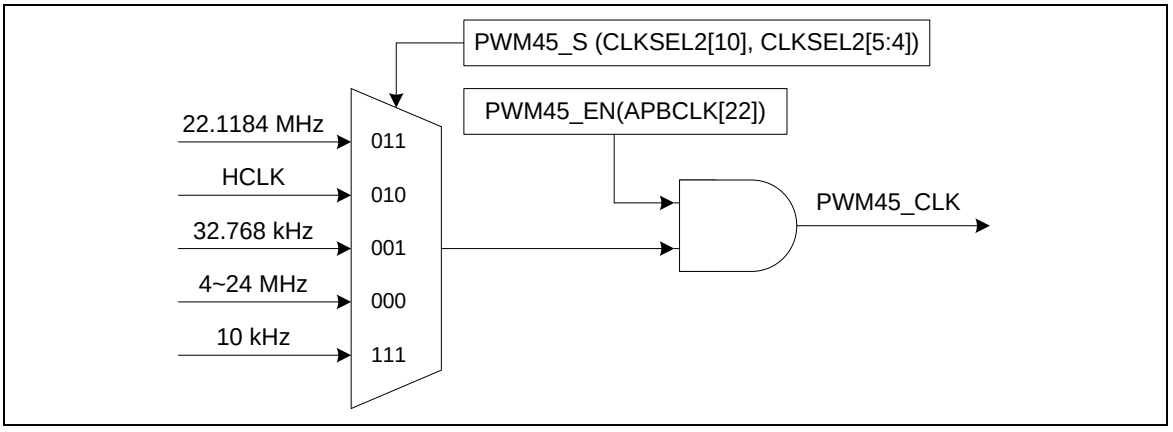


图 5-34 PWM 发生器 4 时钟源控制

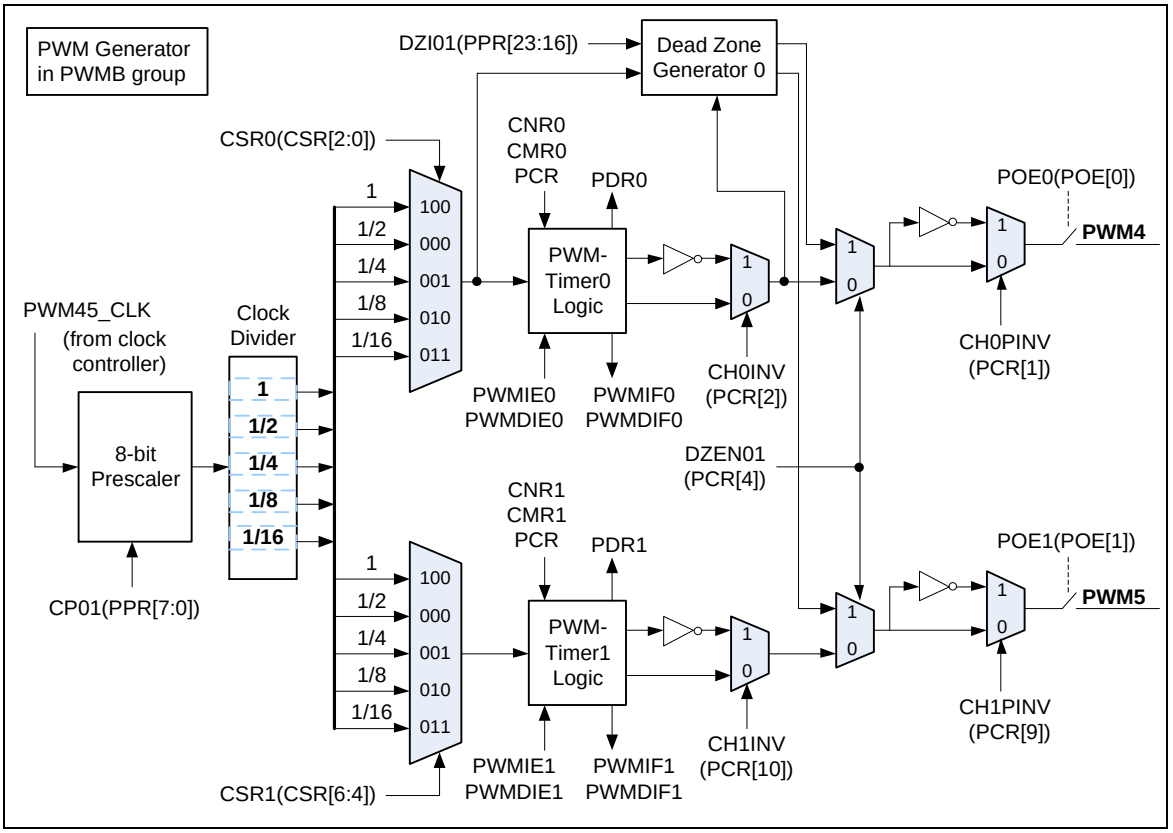


图 5-35PWM 发生器 4 结果框图

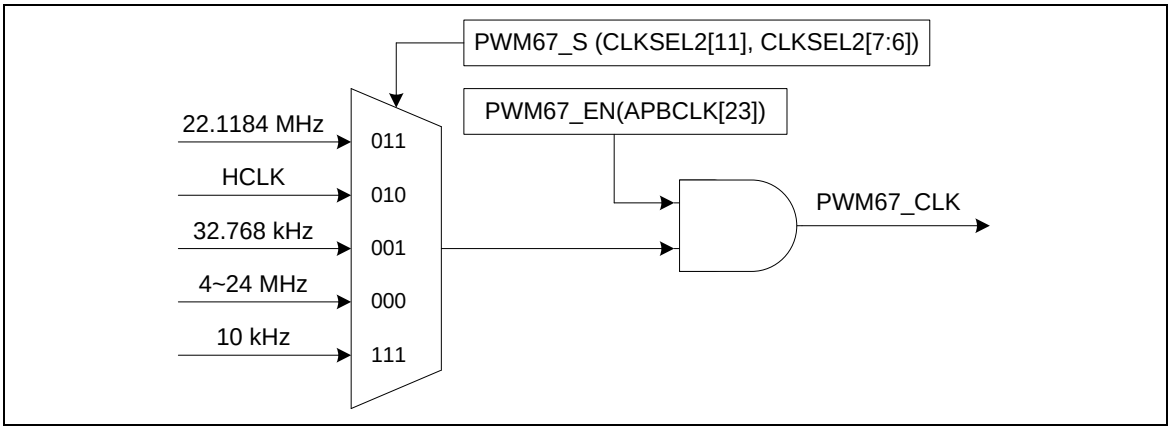


图 5-36 PWM 发生器 6 时钟控制

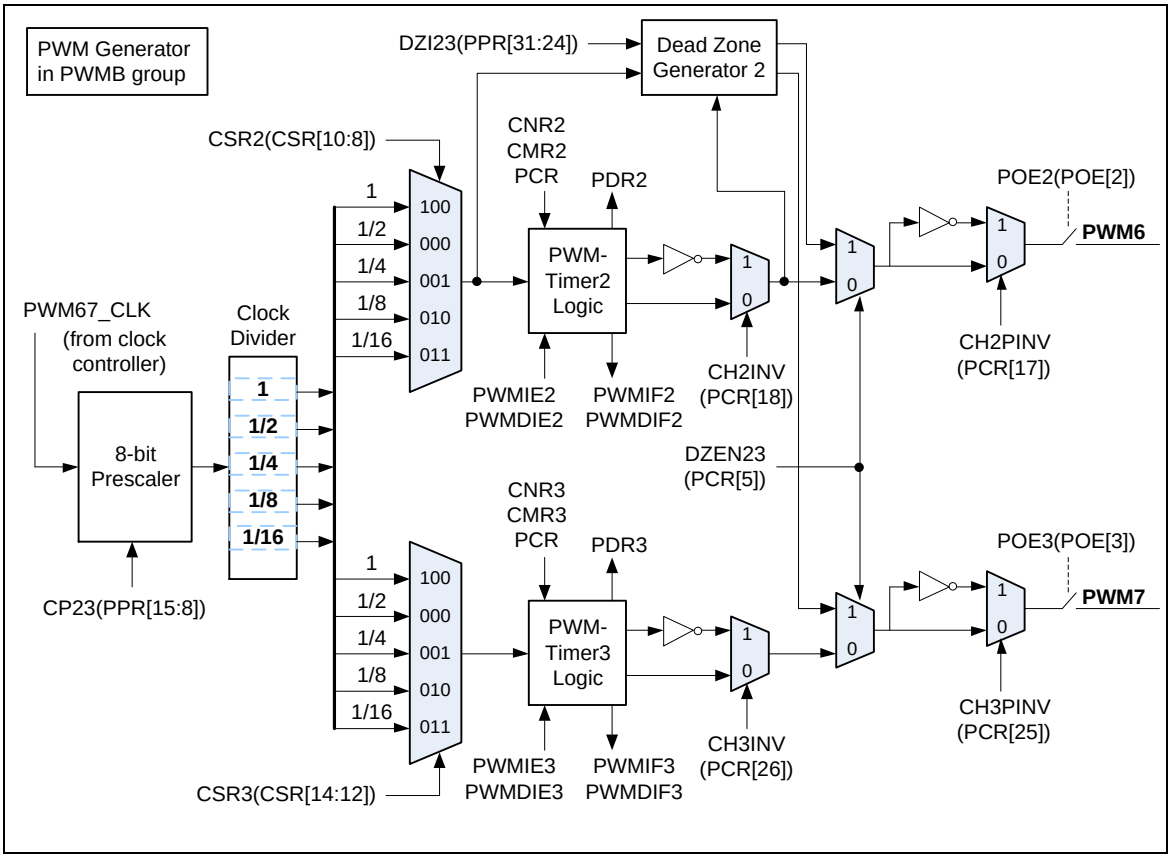


图 5-37 PWM 发生器 6 结构框图

5.9.4 基本设定

PWM引脚功能在寄存器GPA_MFP, GPB_MFP和GPE_MFP中设定。

PWM 时钟使能由APBCLK[23:20]来设定。PWM时钟资源由CLKSEL1[31:28], CLKSEL2[7:4] 和CLKSEL2[11:8]来选择。

5.9.5 功能描述

5.9.5.1 *PWM-定时器操作*

PWM 控制器支持两种操作模式: 边沿对齐和中心对齐

5.9.5.2 *边沿对齐PWM (向下计数)*

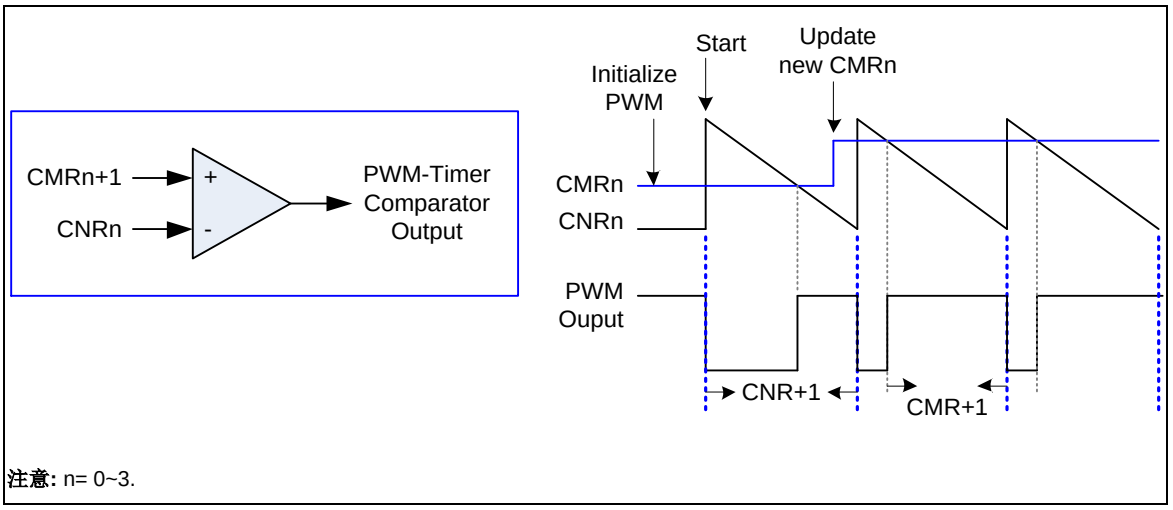
边沿对齐PWM 输出模式下，16位PWM 计数器从CNRn开始向下计数，当等于CMRn(旧)时，PWM发生器将输出低电平。计数器继续下数直到0，同时PWM发生器反转输出高电平。如果CHnMODE=1，CMRn(新)和CNRn(新)将被更新，如果PWM中断被使能(PWMIEn (PIER[3:0]) = 1)，PWM将产生中断

PWM 周期和占空比控制由向下计数的PWM寄存器(CNR)以及PWM比较寄存器(CMR) 控制。PWM定时器工作时序如图6-85. 脉宽宽度调制的公式如下，PWM定时器比较器图示如图6-84

注意：在PWM 功能使能前，相应GPIO管脚应配置成PWM功能(使能POE 并禁止CAPENR).

- PWM 频率 = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; XY代表01, 23, 45 , 67, 取决于所选择的PWM通道
- 占空比= (CMR+1)/(CNR+1)
- CMR >= CNR: PWM 输出高
- CMR < CNR: PWM 低脉宽= (CNR-CMR) unit[1]; PWM高脉宽 = (CMR+1) unit
- CMR = 0: PWM 低脉宽 = (CNR) unit; PWM 高脉宽= 1 unit

注意[1]: unit =一个 PWM 时钟周期



注意: n= 0~3.

图 5-38 PWM-定时器内部比较器输出图例

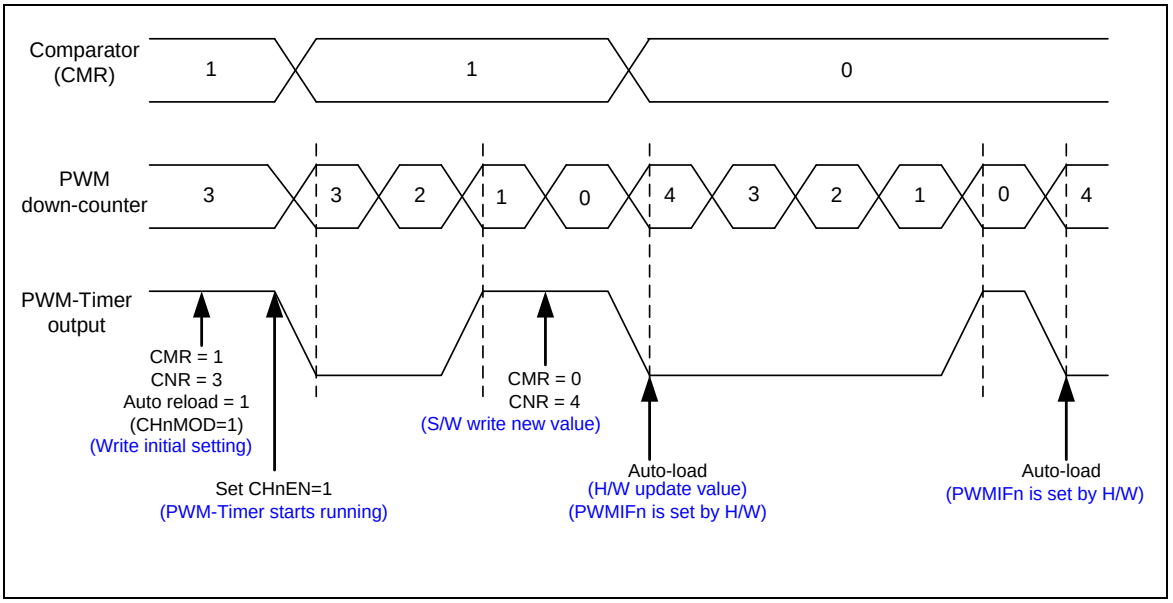


图 5-39 PWM-定时器操作时序

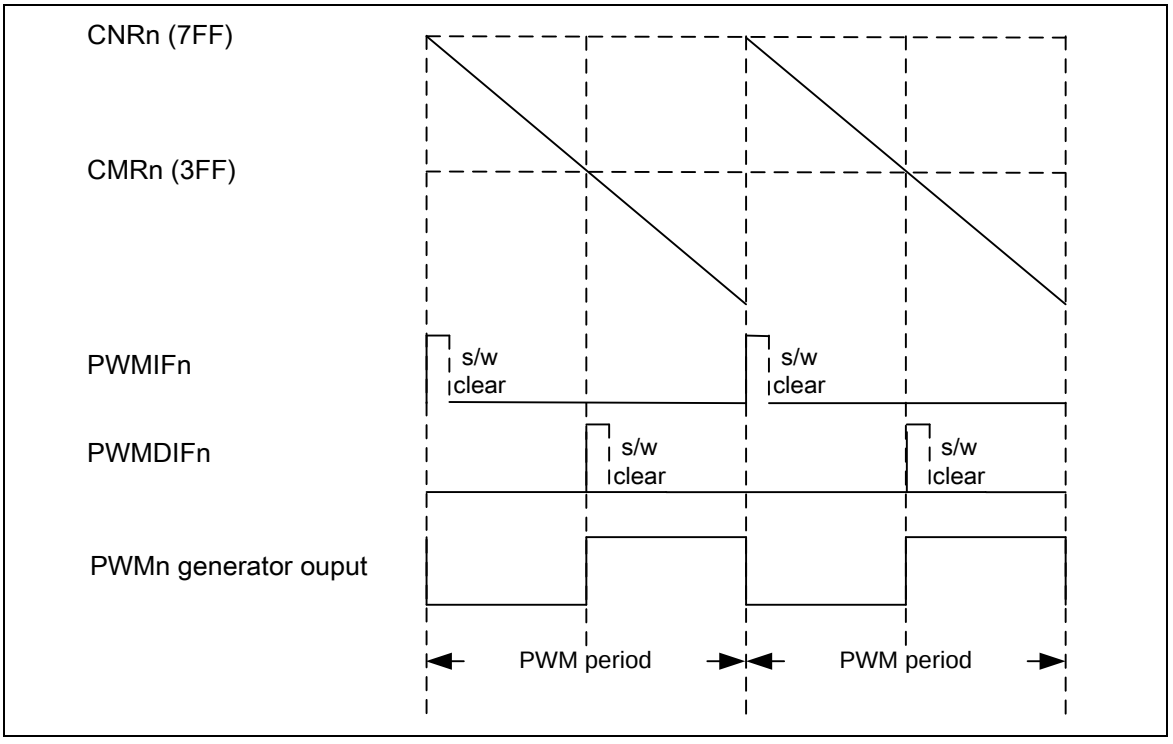


图 5-40 PWM 边缘对齐中断发生时序图

5.9.5.3 *中心对齐PWM (向上向下-计数)*

当PWM 定时器配置为向上计数/向下计数的计数器模式时，中心对齐PWM信号由其产生。PWM 计数器从0开始向上计数，直到等于CMRn(旧)的值，将触发PWM发生器使其输出低电平。计数器继续计数直到CNRn(旧)，之后计数器开始自动向下计数，当再次等于CMRn(旧)时,PWM 发生器输出为高电平。一旦PWM 计数器下溢，如果CHnMODE = 1; PWM 周期寄存器CNRn(新)和占空比寄存器CMRn(新) 将被更新。

中心对齐模式下，如果INTxxTYPE (PIER[17:16]) =0 ，向下计数的计数器下溢时，就在每个PWM 周期的开始（结束）， PWM 周期中断将发生；如果INTTYPExx (PIER[17:16]) =1 ，向上计数的计数器等于CNRn时，就是在每个PWM周期的中点处，PWM 周期中断将发生.

- PWM 频率= PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; xy,代表 01, 23, 45 ,67, 取决于所选的通道
- 占空比 = [(2 x CMR) + 1]/[2 x (CNR+1)]
- CMR > CNR: PWM 输出高
- CMR <= CNR: PWM 低脉宽= 2 x (CNR-CMR) + 1 unit[1]; PWM 高脉宽 = (2 x CMR) + 1 unit
- CMR = 0: PWM 低脉宽 = 2 x CNR + 1 unit; PWM 高脉宽 = 1 unit

  注意[1]: unit = 一个PWM时钟周期

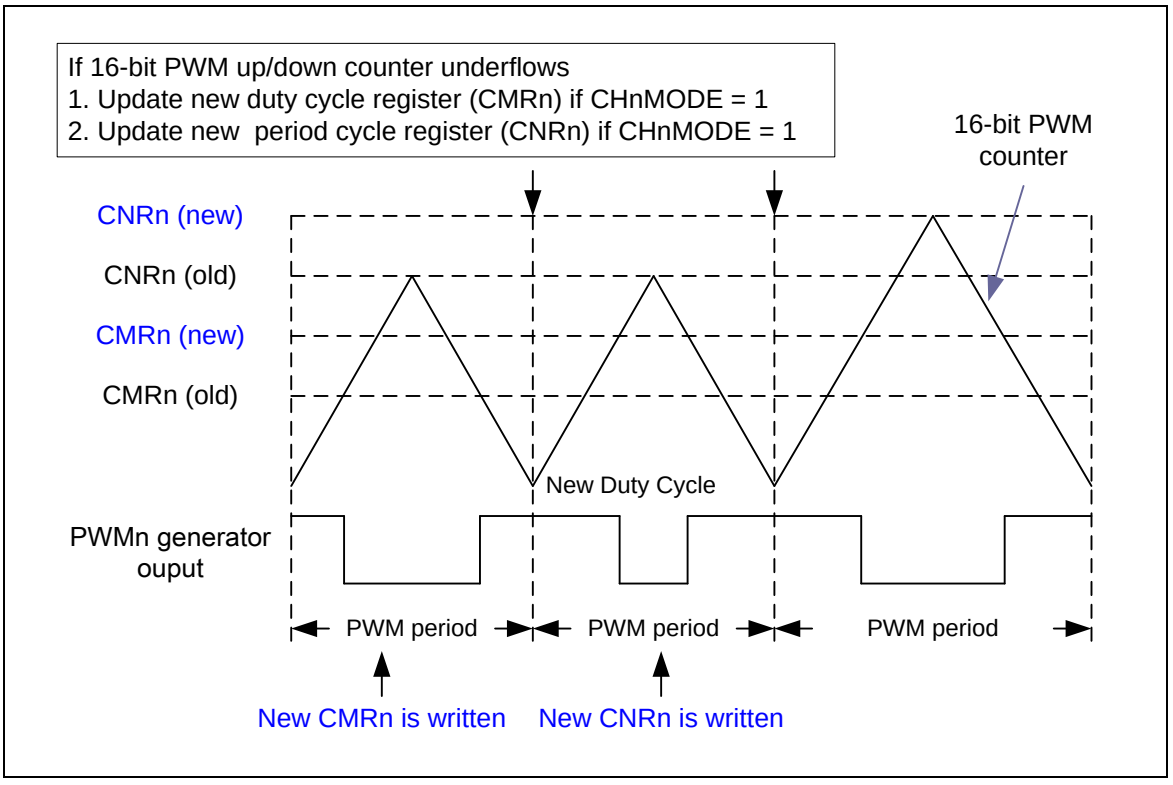


图 5-41 中心对齐模式输出波形

在中心对齐模式下，系统在两个特定时序下可产生PWM周期中断。如果INTxxTYPE (PIER[17:16]) = 0，向下计数的计数器等于0时，PWM周期中断将产生；如果INTxxTYPE (PIER[17:16]) = 1时，向上计数的计数器等于CNRx的值时，就是在每个PWM周期的中点处，PWM 周期中断将发生.

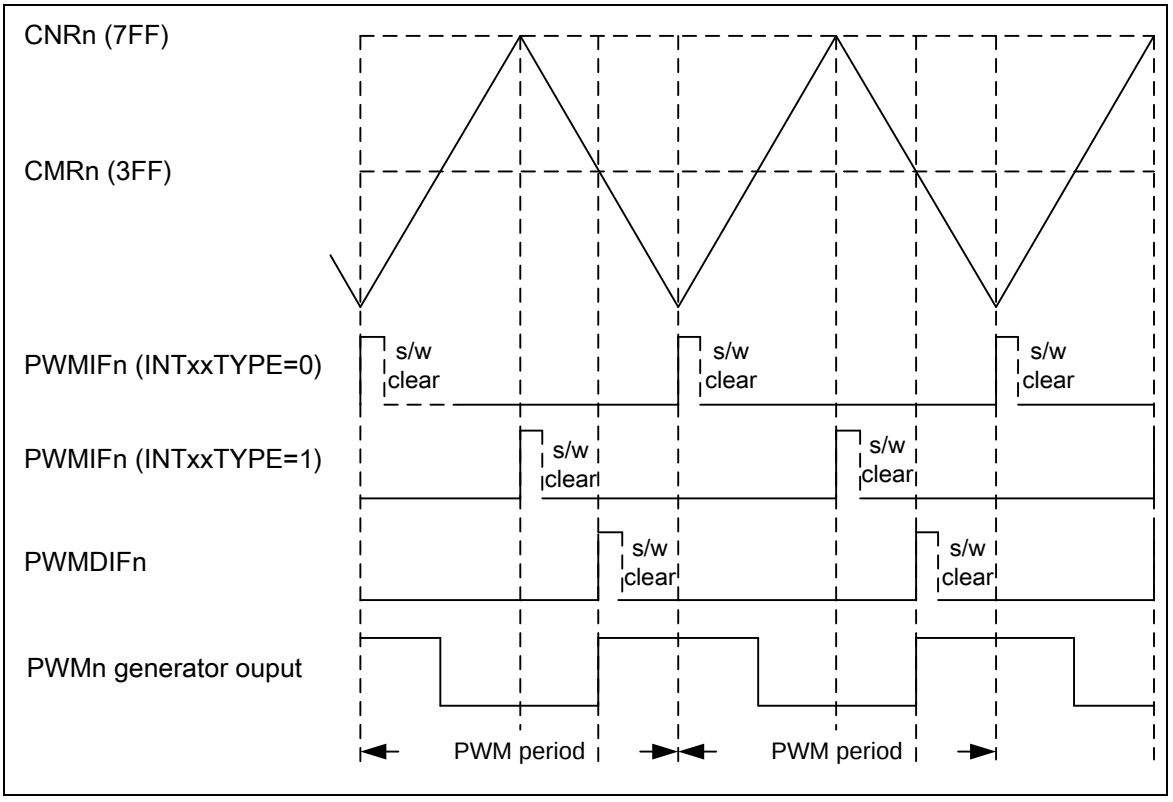


图 5-42 PWM 中心对齐中断发生时序图

5.9.5.4 *PWM 双缓存, 自动重载以及单触发模式*

PWM 定时器具有双缓存功能。寄存器新写入的值，在下一个周期加载，不会影响当前的定时器操作。PWM 计数器的值可以写入CNRx，当前PWM 计数器的值可从PDRx 读出.

如果CH0MOD (PCR[3])位被设定为0，PWM0将是单触发模式；如果CH0MOD (PCR[3])位被设定为1时，PWM0将是自动重载模式。推荐大家在使能PWM0计数器计数（通过设定CH0EN (PCR[0]) 位为1）之前切换PWM0的工作模式，因为在PWM0的工作模式更改时，CNR0和CMR0的内容将会被清0，而PWM0的周期和占空比也会被重置。当PWM0工作在单触发模式下，首先写值到CMR0和CNR0，然后通过设定CH0EN (PCR[0]) 位为1来使能PWM0的计数器，使其开始计数。在PWM0计数器的值从CNR0到0（向下计数）之后，CNR0和CMR0的内容将通过硬件清0，PWM的计数器将停止计数。软件需要写新的内容到CMR0和CNR0寄存器，来设定下一次的周期和占空比。当再一次开启单触发操作，CMR0的内容应该被先写，其原因为当CNR0被写入一个非0的值后，PWM0计数器将会自动地开始计数。当PWM0工作在自动重载模式下，CMR0和CNR0应该被先写值，然后通过设定CH0EN (PCR[0]) 位为1来使能PWM0计数器，使其开始计数。当PWM0计数器向下计数到0时，计数器将会重新装载CNR0的内容，如果CNR0的值被设定为0，计数器将停止计数。PWM1~PWM7具有和PWM0一样的功能。

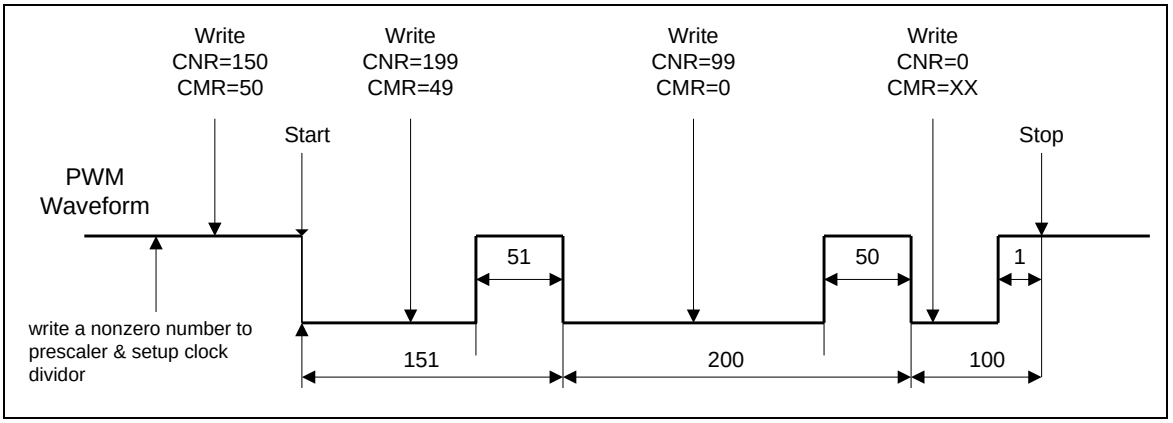


图 5-43 PWM 双缓存图解

5.9.5.5 *占空比调制*

双缓存允许CMRn寄存器的内容可以在任何时间被改写。写入的值在下一个周期才起作用.

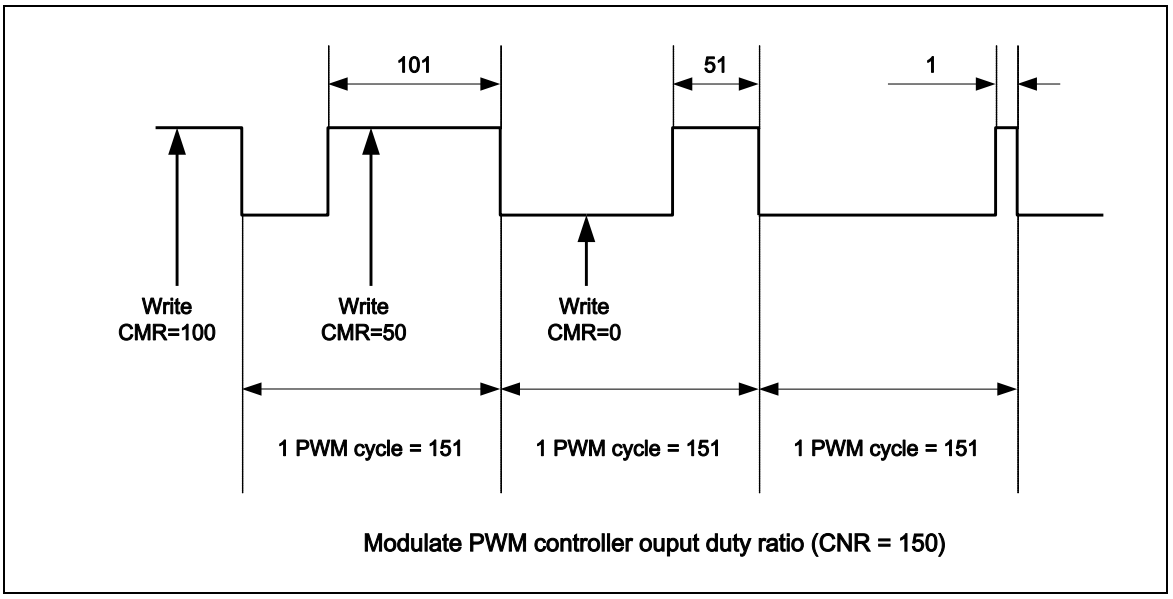


图 5-44 PWM 控制器输出占空比

5.9.5.6 *死区发生器*

PWM控制器提供死区发生器. 用于保护电源器件. 该功能在PWM上升沿产生可编程的延迟时间，用户通过编程DZIxx (PPR[31:16])来确定死区间隔.

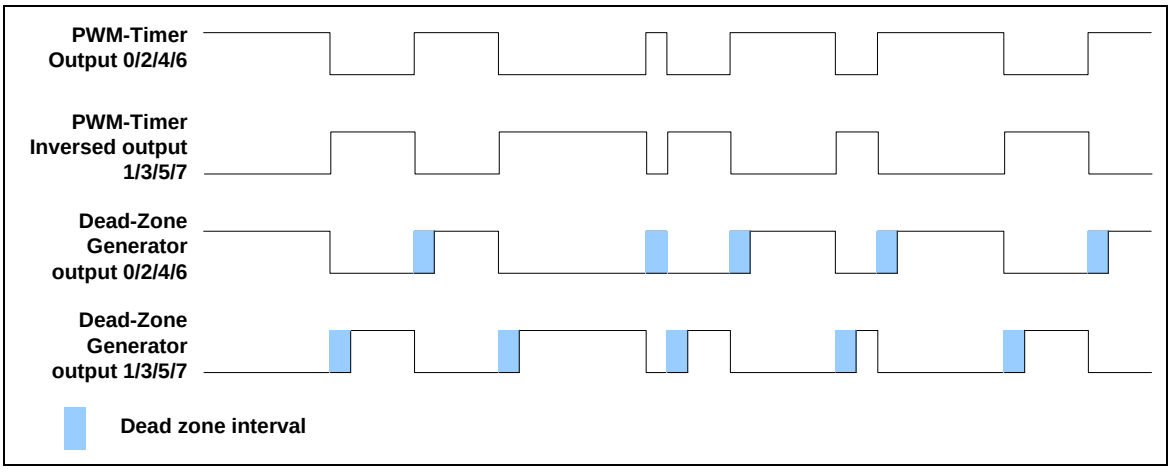


图 5-45 Paired-PWM 对带死区发生器操作输出

5.9.5.7 *PWM 中心对齐 触发ADC 功能*

通过设定PWMnTEN (TCON[3:0]) 为“1”，在中心对齐模式下，当PWM的计数器的达到CNR的值（向上计数）时，PWM能够触发ADC，使其开始转换。

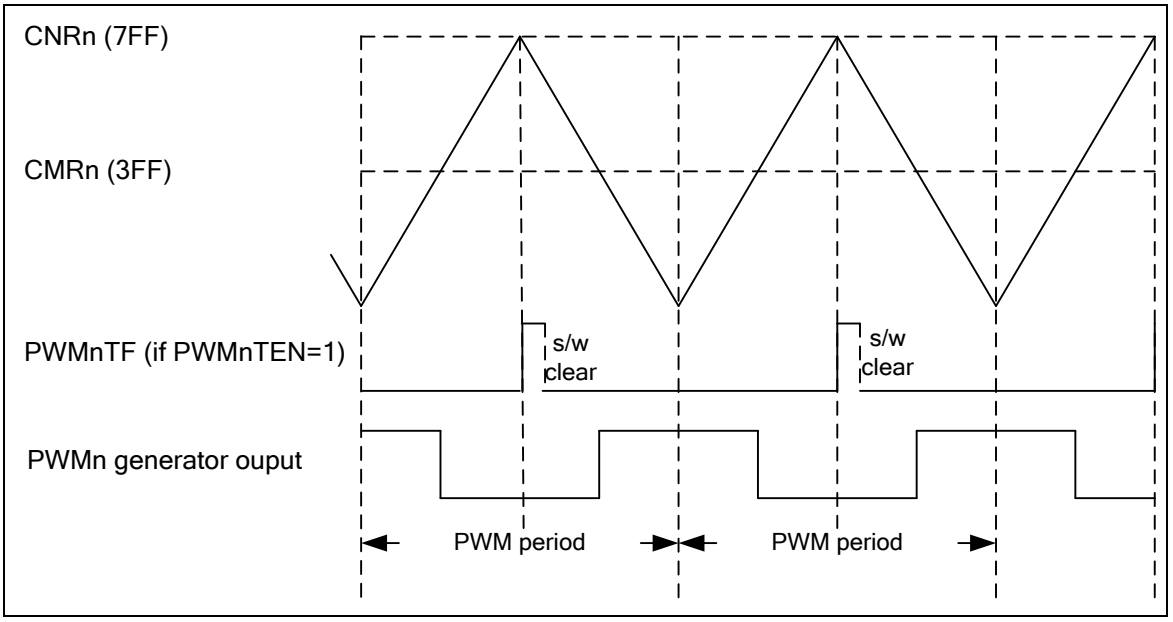


图 5-46 中心对齐模式下 PWM 触发 ADC 转换时序图

5.9.5.8 *捕捉模式的操作*

捕捉器0和PWM0 使用同一个定时器，捕捉器1和PWM1共同使用另一个定时器，以此类推。当输入信号有上升沿转变时，捕捉器将把PWM 计数器的值将存入CRLRn寄存器；当输入信号有下降沿转变时，捕捉器将把PWM 计数器的值将存入CFLRn 寄存器。捕捉器通道0通过软件设定CRL_IE0 (CCR0[1]) (使能上升沿锁存中断)和CFL_IE0 (CCR0[2]) (使能下降沿锁存中断)可以判定其中断是什么中断源导致。同样，捕捉器通道1通过设定CRL_IE1 (CCR0[17]) 和 CFL_IE1 (CCR0[18])具有同样的功能 。依此类推 。每当捕捉控制器触发捕捉中断时 ，相应的PWM 计数器会同时重新装载CNRn 的值。

注意: 在捕捉功能使能前，相应的管脚必须配置为捕获功能(禁止POE并使能CAPENR).

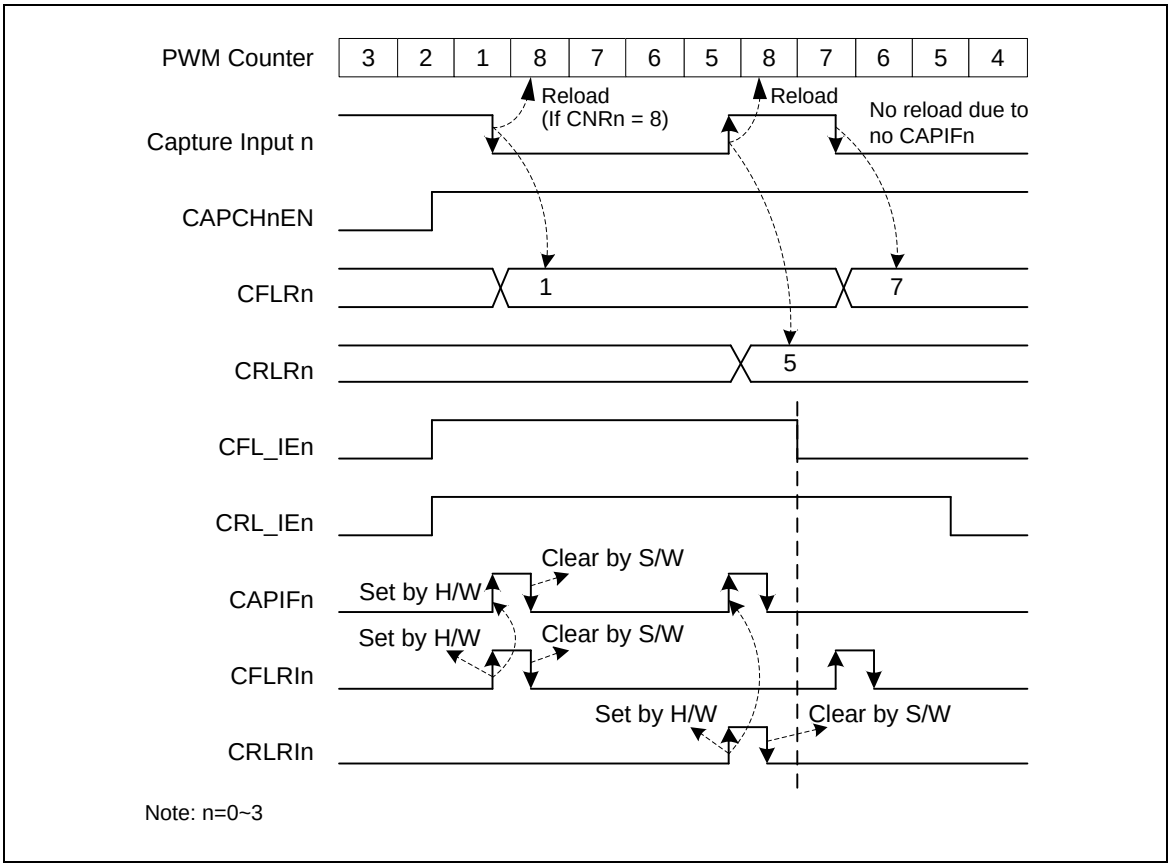


图 5-47 捕捉操作时序

在上述范例中，CNR 为8:

- 当捕捉中断标志(CAPIFn)置位时，PWM计数器将重载CNRn.
- 通道低脉冲宽度为(CNR + 1 - CRLR).
- 通道高脉冲宽度为(CNR + 1 - CFLR).

5.9.5.9 *PWM-定时器中断结构*

有8个PWM 中断，PWM0_INT~PWM7_INT，对高级中断控制器（AIC）来说，这些中断被分成PWMA_INT和PWMB_INT。PWM 0 与捕捉器0共用同一个中断。PWM1 与捕捉器1 共用同一个中断，

以此类推。因此，在同一个通道PWM功能和捕捉功能不能同时发生。图6-94 和图6-95描述了PWM 定时器中断的结构

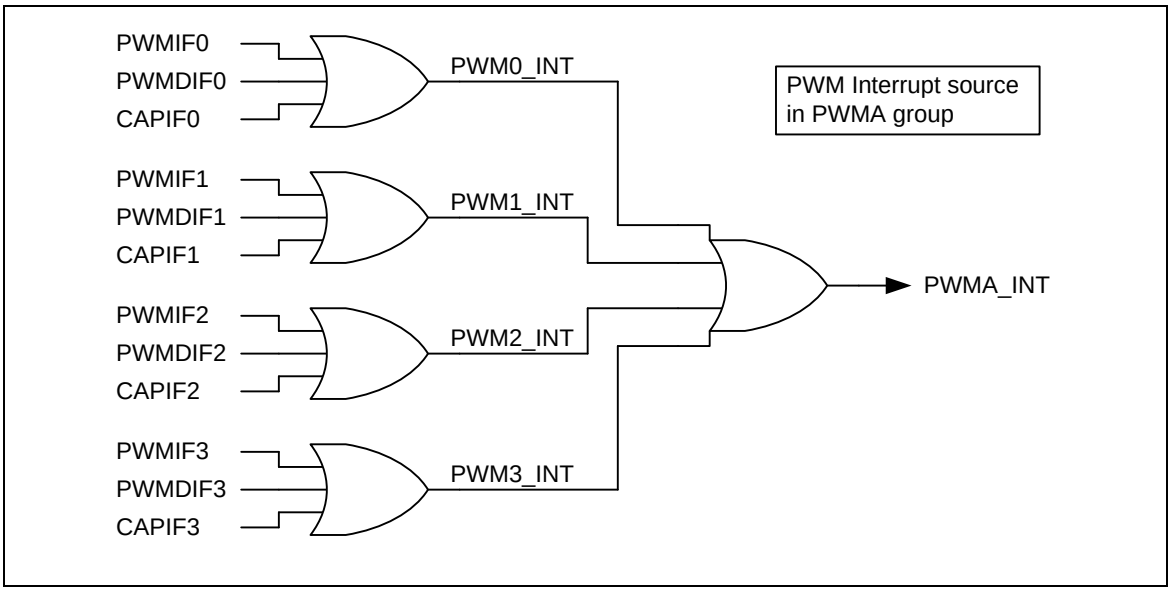


图 5-48 PWM A 组 PWM-定时器中断结构图

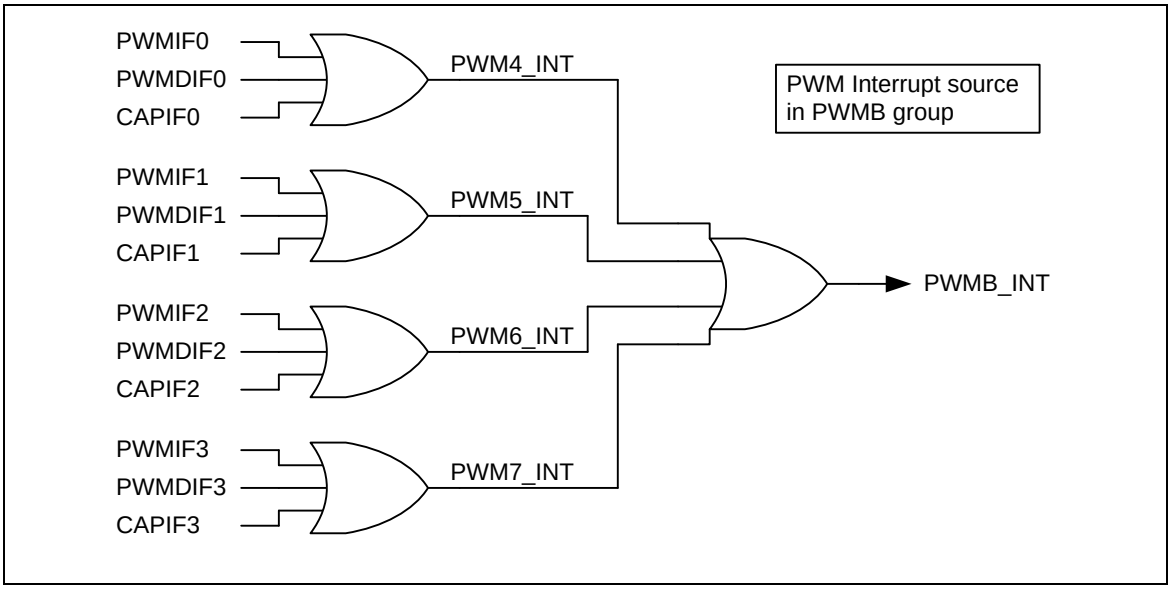


图 5-49 PWM B 组 PWM-定时器中断结构图

5.9.5.10 *PWM-定时器开启步骤*

启动PWM 推荐下列步骤：

1. 配置(CSR)寄存器来选择时钟资源、除频器
2. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
3. 配置预分频(PPR)
4. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
5. 配置反转打开/关闭，死区打开/关闭，自动重载/单触发模式以及停止PWM定时器(PCR)
6. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
7. 配置比较器寄存器(CMR) 设定PWM 占空比.
8. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
9. 配置PWM计数器寄存器(CNR) 设定PWM 周期.
10. 配置中断使能寄存器(PIER) (可选)
11. 配置相应PWM通道GPIO脚为PWM 功能(使能POE ， 关闭CAPENR) .
12. 使能PWM 定时器开始运行(PCR 中的CHnEN = 1)

5.9.5.11 *修改PWM 计数寄存器 (CNR), 比较器(CMR), 时钟预分频(CP01/CP23) 和 PWM 操作 模式(CHnMOD PCR寄存器3位) 步骤*

关于修改CNR/CMR/时钟预分频/PWM操作模式，推荐使用下面的步骤

1. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
2. 修改CMRn/CNRn/CHnMOD/CP01/CP23

5.9.5.12 *单次触发模式下PWM定时器重新启动的步骤*

在PWM单次触发模式下，当PWM波形产生一次后，PWM定时器将自动停止且CNR和CMR的值将会被硬件清0，软件必须重新填写CMR和CNR的值才能重新启动下一个PWM单次波形，关于产生PWM单次波形的步骤，建议按如下步骤进行：

1. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
2. 配置比较器寄存器(CMR)设定PWM占空比.
3. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
4. 配置PWM计数器寄存器(CNR)设定PWM周期.在设定CNR之后, PWM 波形将会产生

5.9.5.13 *PWM-定时器停止步骤*

方法1:

设定16-位计数器(CNR) 为0 ，并查看PDR 状态(计数器的当前值)。当PDR 达到0 ，关闭PWM 定时器(PCR 的CHnEN 位). (推荐)

方法2:

设定16位计数器(CNR) 为0，当中断发生时，关闭PWM定时器(PCR 的CHnEN 位). (推荐)

方法 3:

直接关闭PWM 定时器(PCR 的CHnEN 位). (不推荐)

不推荐方式3是因为禁止CHnEN 将立即停止PWM 输出信号，会导致PWM 输出的占空比改变，可能引起电机控制电路的损坏

5.9.5.14 *捕捉开始步骤*

1. 配置(CSR)寄存器来选择时钟及除频器
2. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
3. 配置预分频(PPR)
4. 配置通道使能，上升/下降沿中断使能以及输入信号反转打开/关闭(CCR0, CCR2)
5. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
6. 配置自动重载模式，边沿对齐，停止PWM定时器(PCR)
7. 如果PWM的时钟不是取HCLK,一直等待直到SYNCBUSYn被硬件设定为0
8. 配置PWM 计数器寄存器(CNR)（向下计数）
9. 使能PWM计数器，开始计数(设定PCR 寄存器的位CHnEN 为1)
10. 设定相应的GPIO引脚为PWM捕捉功能（禁用POE,使能CAPENR）
11. 配置相应的PWM通道所对应的GPIO为捕捉功能(禁用POE和使能CAPENR)

5.9.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
PWM 基地址 ● **PWM A 组** **PWMA_BA = 0x4004_0000** ● **PWM B 组** **PWMB_BA = 0x4014_0000**				
PPR	PWMA_BA+0x00 PWMB_BA+0x00	R/W	PWM 预分频寄存器	0x0000_0000
CSR	PWMA_BA+0x04 PWMB_BA+0x04	R/W	PWM 除频时钟选择寄存器	0x0000_0000
PCR	PWMA_BA+0x08 PWMB_BA+0x08	R/W	PWM控制寄存器	0x0000_0000
CNR0	PWMA_BA+0x0C PWMB_BA+0x0C	R/W	PWM 计数寄存器0	0x0000_0000
CMR0	PWMA_BA+0x10 PWMB_BA+0x10	R/W	PWM 比较寄存器0	0x0000_0000
PDR0	PWMA_BA+0x14 PWMB_BA+0x14	R	PWM 数据寄存器 0	0x0000_0000
CNR1	PWMA_BA+0x18 PWMB_BA+0x18	R/W	PWM 计数寄存器1	0x0000_0000
CMR1	PWMA_BA+0x1C PWMB_BA+0x1C	R/W	PWM 比较寄存器1	0x0000_0000
PDR1	PWMA_BA+0x20 PWMB_BA+0x20	R	PWM 数据寄存器1	0x0000_0000
CNR2	PWMA_BA+0x24 PWMB_BA+0x24	R/W	PWM 计数寄存器 2	0x0000_0000
CMR2	PWMA_BA+0x28 PWMB_BA+0x28	R/W	PWM比较寄存器2	0x0000_0000
PDR2	PWMA_BA+0x2C PWMB_BA+0x2C	R	PWM数据寄存器2	0x0000_0000
CNR3	PWMA_BA+0x30 PWMB_BA+0x30	R/W	PWM计数寄存器r 3	0x0000_0000
CMR3	PWMA_BA+0x34 PWMB_BA+0x34	R/W	PWM 比较寄存器 3	0x0000_0000
PDR3	PWMA_BA+0x38 PWMB_BA+0x38	R	PWM 数据寄存器 3	0x0000_0000
PBCR	PWMA_BA+0x3C PWMB_BA+0x3C	R/W	PWM 向后兼容寄存器	0x0000_0000
PIER	PWMA_BA+0x40	R/W	PWM 中断使能寄存器	0x0000_0000

	PWMB_BA+0x40			
PIIR	PWMA_BA+0x44 PWMB_BA+0x44	R/W	PWM 中断标志寄存器	0x0000_0000
CCR0	PWMA_BA+0x50 PWMB_BA+0x50	R/W	PWM 捕捉控制寄存器 0	0x0000_0000
CCR2	PWMA_BA+0x54 PWMB_BA+0x54	R/W	PWM 捕捉控制寄存器2	0x0000_0000
CRLR0	PWMA_BA+0x58 PWMB_BA+0x58	R	PWM 捕捉上升沿锁存寄存器(通道 0)	0x0000_0000
CFLR0	PWMA_BA+0x5C PWMB_BA+0x5C	R	PWM 捕捉下降沿锁存寄存器 (通道0)	0x0000_0000
CRLR1	PWMA_BA+0x60 PWMB_BA+0x60	R	PWM捕捉上升沿锁存寄存器(通道1)	0x0000_0000
CFLR1	PWMA_BA+0x64 PWMB_BA+0x64	R	PWM捕捉下降沿锁存寄存器(通道1)	0x0000_0000
CRLR2	PWMA_BA+0x68 PWMB_BA+0x68	R	PWM 捕捉上升沿锁存寄存器(通道2)	0x0000_0000
CFLR2	PWMA_BA+0x6C PWMB_BA+0x6C	R	PWM 捕捉下降沿锁存寄存器 (通道2)	0x0000_0000
CRLR3	PWMA_BA+0x70 PWMB_BA+0x70	R	PWM 捕捉上升沿锁存寄存器(通道3)	0x0000_0000
CFLR3	PWMA_BA+0x74 PWMB_BA+0x74	R	PWM捕捉下降沿锁存寄存器(通道3)	0x0000_0000
CAPENR	PWMA_BA+0x78 PWMB_BA+0x78	R/W	PWM 捕捉输入0~3 使能寄存器	0x0000_0000
POE	PWMA_BA+0x7C PWMB_BA+0x7C	R/W	PWM 使能通道0~3 输出	0x0000_0000
TCON	PWMA_BA+0x80 PWMB_BA+0x80	R/W	PWM 关于通道0~3的 触发控制	0x0000_0000
TSTATUS	PWMA_BA+0x84 PWMB_BA+0x84	R/W	PWM 触发状态寄存器	0x0000_0000
SYNCBUSY0	PWMA_BA+0x88 PWMB_BA+0x88	R	PWM0 同步忙状态寄存器	0x0000_0000
SYNCBUSY1	PWMA_BA+0x8C PWMB_BA+0x8C	R	PWM1同步忙状态寄存器	0x0000_0000
SYNCBUSY2	PWMA_BA+0x90 PWMB_BA+0x90	R	PWM2同步忙状态寄存器	0x0000_0000
SYNCBUSY3	PWMA_BA+0x94 PWMB_BA+0x94	R	PWM3同步忙状态寄存器	0x0000_0000

5.9.7 寄存器描述

PWM 预分频寄存器(PPR)

寄存器	偏移地址	R/W	描述	复位值
PPR	PWMA_BA+0x00 PWMB_BA+0x00	R/W	PWM 预分频寄存器	0x0000_0000

31	30	29	28	27	26	25	24
DZI23							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DZI01							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CP23							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CP01							

位	描述	
[31:24]	**DZI23**	**通道2与通道3的死区间隔(PWM A组的PWM2 和PWM3 对，PWM B组的PWM6 和PWM7 对)** 该8位决定死区长度. 死区长度的单位时间= [(prescale+1)*(clock source divider)]/ PWMxy_CLK (xy 可能是 23 或 67，取决于所选的PWM 通道).
[23:16]	**DZI01**	**通道0与通道1的死区间隔(PWM A组的PWM0 和PWM1 对，PWM B组的PWM4 和PWM5 对)** 该8位决定死区长度. 死区长度的单位时间= [(prescale+1)*(clock source divider)]/ PWMxy_CLK (xy 可能是 01 或 45，取决于所选的PWM 通道.)
[15:8]	**CP23**	**时钟预分频 2 (A组PWM-定时器2 / 3 和B组 PWM-定时器 6 / 7)** 时钟在输入到相应的PWM 定时器之前被(CP23 + 1) 除频 如果CP23=0，预分频器2输出时钟停止。相应PWM 定时器也停止.
[7:0]	**CP01**	**时钟预分频 0 (A组PWM-定时期0 / 1和B组PWM-定时器 4 / 5)** 时钟在输入到相应的PWM 定时器之前被(CP23 + 1) 除频 如果CP01=0，预分频器0输出时钟停止。相应PWM 定时器也停止.

PWM时钟除频选择寄存器 (CSR)

寄存器	偏移地址	R/W	描述	复位值
CSR	PWMA_BA+0x04 PWMB_BA+0x04	R/W	PWM 时钟除频选择寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	CSR3			Reserved	CSR2		
7	6	5	4	3	2	1	0
Reserved	CSR1			Reserved	CSR0		

位	描述	
[31:15]	**Reserved**	保留
[14:12]	**CSR3**	**PWM 定时器3 时钟除频选择(A组PWM 定时器3 和B组PWM 定时器7)** PWM 定时器3的时钟除频选择 000 = 2. 001 = 4. 010 = 8. 011 = 16. 100 = 1.
[11]	**Reserved**	保留
[10:8]	**CSR2**	**PWM 定时器 2时钟除频选择(A组PWM定时器2 和B组PWM定时器6)** PWM 定时器2的时钟除频选择 (和 CSR3一样)
[7]	**Reserved**	保留
[6:4]	**CSR1**	**PWM 定时器1 时钟除频选择 (A组PWM 定时器1 和B组 PWM 定时器5)** PWM 定时器1的时钟除频选择 (和 CSR3一样)
[3]	**Reserved**	保留
[2:0]	**CSR0**	**PWM定时器0时钟除频选择(A组PWM定时器0 和B组 PWM 定时器 4)** PWM 定时器0的时钟除频选择 (和 CSR3一样)

PWM控制寄存器 (PCR)

寄存器	偏移地址	R/W	描述	复位值
PCR	PWMA_BA+0x08 PWMB_BA+0x08	R/W	PWM控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
PWM23TYPE	PWM01TYPE	Reserved		CH3MOD	CH3INV	CH3PINV	CH3EN
23	22	21	20	19	18	17	16
Reserved				CH2MOD	CH2INV	CH2PINV	CH2EN
15	14	13	12	11	10	9	8
Reserved				CH1MOD	CH1INV	CH1PINV	CH1EN
7	6	5	4	3	2	1	0
Reserved		DZEN23	DZEN01	CH0MOD	CH0INV	CH0PINV	CH0EN

位	描述	
[31]	PWM23TYPE	**PWM23对齐类型选择 (A组PWM2 和 PWM3 对，B组的PWM6 和PWM7对)** 0 = 边沿对齐 1 = 中心对齐
[30]	PWM01TYPE	**PWM01对齐类型选择(A组PWM0 和 PWM1 对, B组PWM4 和PWM5 对)** 0 =边沿对齐 1 =中心对齐
[30:28]	Reserved	保留
[27]	CH3MOD	**PWM-定时器 3 自动重载/单次模式(A组PWM 定时器3 和B组 PWM 定时器7)** 0 = 单次模式 1 = 自动重载 **注意：**如果该位的值变化，会使CNR3 和CMR3 清0.
[26]	CH3INV	**PWM-定时器 3输出反转使能(A组PWM 定时器 3 和B组PWM 定时器 7)** 0 = 反转关闭 1 = 反转打开
[25]	CH3PINV	**PWM-定时器 3 输出极性反转使能(A组PWM 定时器3和B组 PWM 定时器 7)** 0 = PWM3输出极性不反转 1 = PWM3输出极性反转使能
[24]	CH3EN	**PWM-定时器3 使能(A组PWM 定时器3 和PWM 定时器 7)** 0 =相应的PWM 定时器停止运行 1 =相应的PWM 定时器开始运行
[23:20]	Reserved	保留
[19]	CH2MOD	**PWM-定时器2自动重载/单次模式(A组PWM 定时器 2 和B组PWM 定时器 6)**

		0 =单次模式 1 = 自动重载模式 **注意:** 如果该位的值变化，会使CNR2和CMR2 清0.
[18]	**CH2INV**	**PWM-定时器2 输出反转使能（A组PWM 定时器 2和B组 PWM 定时器 6)** 0 =反转关闭 1 =反转打开
[17]	**CH2PINV**	**PWM-定时器 2输出极性反转使能(A组PWM 定时器2和 PWM 定时器6)** 0 = PWM2输出极性不反转 1 = PWM2输出极性反转使能.
[16]	**CH2EN**	**PWM-定时器 2 使能(A组PWM 定时器2和B组PWM 定时器 6)** 0 =相应的PWM 定时器停止运行 1 =相应的PWM 定时器开始运行
[15:12]	**Reserved**	保留.
[11]	**CH1MOD**	**PWM-定时器1 自动重载/单次模式(A组PWM 定时器1 和 B组PWM 定时器 5)** 0 = 单次模式 1 = 自动重载 **注意:** 如果该位的值变化，会使CNR1和CMR1清0.
[10]	**CH1INV**	**PWM-定时器 1 输出反转使能(A组PWM 定时器1 和B组PWM 定时器 5)** 0 =反转关闭 1 =反转打开
[9]	**CH1PINV**	**PWM-定时器 1输出极性反转使能(A组PWM 定时器1和B组PWM 定时器5)** 0 = PWM1输出极性不反转. 1 = PWM1输出极性反转使能
[8]	**CH1EN**	**PWM-定时器1 使能(A组PWM 定时器 1 和B组PWM 定时器5)** 0 =相应的PWM 定时器停止运行 1 =相应的PWM 定时器开始运行
[7:6]	**Reserved**	保留.
[5]	**DZEN23**	**死区 2 发生器使能(A组PWM2 和PWM3 对, B组PWM6 和PWM7 对)** 0 = 禁用. 1 = 使能 **注意:** 当死区发生器使能，PWM A 组的PWM2 和PWM3 将成为一对，输出互补信号，PWM B 组的PWM6 和PWM7将成为一对，输出互补信号，
[4]	**DZEN01**	**死区 0 发生器使能(A组PWM0 和 PWM1 对, B组PWM4 和 PWM5对)** 0 =禁用 1 =使能 **注意:** 当死区发生器使能，PWM A 组的PWM0 和PWM1 将成为一对，输出互补信号，PWM B 组的PWM4 和PWM5将成为一对，输出互补信号，
[3]	**CH0MOD**	**PWM-定时器 0 自动重载/单次模式 (A组PWM 定时器 0 和B组PWM 定时器 4)** 0 = 单次模式. 1 = 自动重载 **注意:** 如果该位的值变化，会使CNR0和CMR0 清0.

[2]	**CH0INV**	**PWM-定时器 0 输出反转使能(A组PWM 定时器0和B组PWM 定时器 4)** 0 =反转关闭 1 =反转打开
[1]	**CH0PINV**	**PWM-定时器 0 输出极性反转使能 (A组PWM 定时器 0 和B组 PWM 定时器4)** 0 = PWM0输出极性不反转. 1 = PWM0输出极性反转使能.
[0]	**CH0EN**	**PWM-定时器 0 使能 (A组PWM 定时器0 和B组PWM 定时器4)** 0 =相应的PWM 定时器停止运行. 1 =相应的PWM 定时器开始运行.

PWM 计数寄存器3-0 (CNR3-0)

寄存器	偏移地址	R/W	描述	复位值
CNR0	PWMA_BA+0x0C PWMB_BA+0x0C	R/W	PWM 计数寄存器0	0x0000_0000
CNR1	PWMA_BA+0x18 PWMB_BA+0x18	R/W	PWM计数寄存器 1	0x0000_0000
CNR2	PWMA_BA+0x24 PWMB_BA+0x24	R/W	PWM计数寄存器 2	0x0000_0000
CNR3	PWMA_BA+0x30 PWMB_BA+0x30	R/W	PWM计数寄存器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**
CNRx [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNRx [7:0]							

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**CNRx**	**PWM 定时器载入值** CNR决定了PWM周期. PWM频率= PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; xy可能为01, 23, 45 或67, 取决于所选择的PWM 通道 边沿对齐模式 ● 占空比= (CMR+1)/(CNR+1). ● CMR >= CNR: PWM 输出高 ● CMR < CNR: PWM 低脉宽= (CNR-CMR) unit; PWM 高脉宽= (CMR+1) unit. ● CMR = 0: PWM低脉宽= (CNR) unit; PWM高脉宽= 1 unit. 中心对齐模式: ● 占空比= [(2 x CMR) + 1]/[2 x (CNR+1)]. ● CMR > CNR: PWM输出高. ● CMR <= CNR: PWM低脉宽= 2 x (CNR-CMR) + 1 unit; PWM高脉宽= (2 x CMR) + 1 unit. ● CMR = 0: PWM 低脉宽 = 2 x CNR + 1 unit; PWM 高脉宽 = 1 unit. (Unit =一个PWM时钟周期) . **注意:** CNR写入数据后将在下一个PWM 周期生效 **注意:**当PWM 工作在中心对齐模式时，CNR 的值应该被设在0x0000 到0xFFFE 之间。如

		果CNR 等于0xFFFF， PWM 工作将不正常 **注意:** 当CNR 等于0时，PWM 输出总是高电平

PWM 比较寄存器 3-0 (CMR3-0)

寄存器	偏移地址	R/W	描述	复位值
CMR0	PWMA_BA+0x10 PWMB_BA+0x10	R/W	PWM 比较器0	0x0000_0000
CMR1	PWMA_BA+0x1C PWMB_BA+0x1C	R/W	PWM比较器1	0x0000_0000
CMR2	PWMA_BA+0x28 PWMB_BA+0x28	R/W	PWM比较器2	0x0000_0000
CMR3	PWMA_BA+0x34 PWMB_BA+0x34	R/W	PWM比较器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**
CMRx [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMRx [7:0]							

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**CMRx**	**PWM 比较寄存器** CMR 决定PWM占空比 PWM 频率 = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; xy, 可能 01, 23, 45或67, 取决于所选择的PWM 通道. 边沿对齐模式 ● 占空比 = (CMR+1)/(CNR+1). ● CMR >= CNR: PWM 输出高. ● CMR < CNR: PWM低脉宽= (CNR-CMR) unit; PWM 高脉宽 = (CMR+1) unit. ● CMR = 0: PWM 低脉宽= (CNR) unit; PWM高脉宽 = 1 unit. 中心对齐模式: ● 占空比= [(2 x CMR) + 1]/[2 x (CNR+1)]. ● CMR > CNR: PWM 输出高 ● CMR <= CNR: PWM低脉宽= 2 x (CNR-CMR) + 1 unit; PWM高脉宽= (2 x CMR) + 1 unit. ● CMR = 0: PWM低脉宽= 2 x CNR + 1 unit; PWM高脉宽= 1 unit. (Unit = 一个PWM 周期). **注意:** CMR写入数据后将在下一个PWM 周期生效

PWM 数据寄存器 3-0 (PDR 3-0)

寄存器	偏移地址	R/W	描述	复位值
PDR0	PWMA_BA+0x14 PWMB_BA+0x14	R	PWM 数据寄存器 0	0x0000_0000
PDR1	PWMA_BA+0x20 PWMB_BA+0x20	R	PWM 数据寄存器1	0x0000_0000
PDR2	PWMA_BA+0x2C PWMB_BA+0x2C	R	PWM 数据寄存器 2	0x0000_0000
PDR3	PWMA_BA+0x38 PWMB_BA+0x38	R	PWM 数据寄存器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**
PDR[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDR[7:0]							

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**PDRx**	**PWM 数据寄存器** 用户可以监控PDR来查询16位计数器的当前值.

PWM 向后兼容寄存器r(PBCR)

寄存器	偏移地址	R/W	描述	复位值
PBCR	PWMA_BA+0x3C PWMB_BA+0x3C	R/W	PWM 向后兼容寄存器	0x0000_0000

31	30	29	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							**BCn**

位	描述	
[31:1]	**Reserved**	保留
[0]	**BCn**	**PWM 向后兼容寄存器** 0 =确定写0清CFLRI0~3 和CRLRI0~3 1 =确定写1清CFLRI0~3 和CRLRI0~3 参考寄存器CCR0/CCR2 的第6, 7, 22, 23 位描述 **注意:** 推荐该位设定为1，当写值到CCR0/CCR2，可以避免CFLRIx 和 CRLRIx被当前的清掉

PWM 中断使能寄存器(PIER)

寄存器	偏移地址	R/W	描述	复位值
PIER	PWMA_BA+0x40 PWMB_BA+0x40	R/W	PWM 中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						INT23TYPE	INT01TYPE
15	14	13	12	11	10	9	8
Reserved				PWMDIE3	PWMDIE2	PWMDIE1	PWMDIE0
7	6	5	4	3	2	1	0
Reserved				PWMIE3	PWMIE2	PWMIE1	PWMIE0

位	描述	
[31:18]	**Reserved**	保留
[17]	**INT23TYPE**	**PWM23周期中断类型选择位（A组PWM2 和 PWM3 对，B组PWM6 和 PWM7对)** 0 =如果PWM 计数器的值下溢，PWMIFn 将被置 1 =如果PWM 计数器的值等于CNRn 寄存器的值，PWMIFn 将被置. **注意:** 该位只有在PWM 中心对齐模式才有效.
[16]	**INT01TYPE**	**PWM01 周期中断类型选择位 (A组PWM0 和 PWM1 对, B组PWM4 和 PWM5 对)** 0 =如果PWM 计数器的值下溢，PWMIFn 将被置 1 =如果PWM 计数器的值等于CNRn 寄存器的值，PWMIFn 将被置. **注意:** 该位只有在PWM 中心对齐模式才有效.
[11]	**PWMDIE3**	**PWM 通道3占空比中断使能位** 0 =禁止 1 =使能.
[10]	**PWMDIE2**	**PWM通道2占空比中断使能位** 0 =禁止 1 =使能.
[9]	**PWMDIE1**	**PWM通道1占空比中断使能位** 0 =禁止 1 =使能.
[8]	**PWMDIE0**	**PWM 通道0占空比中断使能位** 0 =禁止 1 =使能..
[7:4]	**Reserved**	保留

[3]	**PWMIE3**	**PWM 通道 3 周期中断使能位** 0 =禁止 1 =使能. .
[2]	**PWMIE2**	**PWM通道 2周期中断使能位** 0 =禁止 1 =使能.
[1]	**PWMIE1**	**PWM 通道1周期中断使能位** 0 =禁止 1 =使能.
[0]	**PWMIE0**	**PWM 通道 0周期中断使能位** 0 =禁止 1 =使能.

PWM 中断标志寄存器(PIIR)

寄存器	偏移地址	R/W	描述	复位值
PIIR	PWMA_BA+0x44 PWMB_BA+0x44	R/W	PWM 中断标志寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				PWMDIF3	PWMDIF2	PWMDIF1	PWMDIF0
7	6	5	4	3	2	1	0
Reserved				PWMIF3	PWMIF2	PWMIF1	PWMIF0

位	描述	
[31:12]	**Reserved**	保留
[11]	**PWMDIF3**	**PWM通道3占空比中断标志** 当PWM 通道3计数器向下计数，其值等于CMR3 寄存器的值时，这个标志将由硬件设置， 软件可以写1清除 **注意:** 在边沿对齐模式下， 如果CMR等于CNR,这个标志不工作
[10]	**PWMDIF2**	**PWM 通道2占空比中断标志** 当PWM 通道2计数器向下计数，其值等于CMR2 寄存器的值时，这个标志将由硬件设置， 软件可以写1清除 **注意:** 在边沿对齐模式下， 如果CMR等于CNR,这个标志不工作
[9]	**PWMDIF1**	**PWM 通道1占空比中断标志** 当PWM 通道1计数器向下计数，其值等于CMR1 寄存器的值时，这个标志将由硬件设置， 软件可以写1清除 **注意:** 在边沿对齐模式下， 如果CMR等于CNR,这个标志不工作
[8]	**PWMDIF0**	**PWM通道0占空比中断标志** 当PWM 通道0计数器向下计数，其值等于CMR0寄存器的值时，这个标志将由硬件设置， 软件可以写1清除 **注意:** 在边沿对齐模式下， 如果CMR等于CNR,这个标志不工作
[7:4]	**Reserved**	保留.
[3]	**PWMIF3**	**PWM 通道3周期中断状态** 当PWM3 计数器的值达到中断要求(依靠PIER寄存器INT23 TYPE位的设定)， 硬件自动将该位置1。软件写1清除该位
[2]	**PWMIF2**	**PWM 通道2周期中断状态** 当PWM2计数器的值达到中断要求(依靠PIER寄存器INT23 TYPE位的设定)，硬件自动将该位置1。软件写1清除该位

[1]	**PWMIF1**	**PWM 通道 1周期中断状态** 当PWM1 计数器的值达到中断要求(依靠PIER寄存器INT23 TYPE位的设定)，硬件自动将该位置1。软件写1清除该位
[0]	**PWMIF0**	**PWM 通道0周期中断状态** 当PWM0 计数器的值达到中断要求(依靠PIER寄存器INT23 TYPE位的设定)，硬件自动将该位置1。软件写1清除该位

注意:通过将PIIR 寄存器相应位写1，用户可以清除每个中断标志.

捕捉控制寄存器(CCR0)

寄存器	偏移地址	R/W	描述	复位值
CCR0	PWMA_BA+0x50 PWMB_BA+0x50	R/W	PWM 捕捉控制寄存器0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CFLRI1	CRLRI1	Reserved	CAPIF1	CAPCH1EN	CFL_IE1	CRL_IE1	INV1
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CFLRI0	CRLRI0	Reserved	CAPIF0	CAPCH0EN	CFL_IE0	CRL_IE0	INV0

位	描述	
[31:24]	**Reserved**	保留
[23]	**CFLRI1**	**CFLR1锁定标志位** 当PWM 组输入通道1发生下降沿变化时，CFLR1 锁存PWM 向下计数器的值，该位由硬件置位. 如果BCn 为0，软件写0清该位，如果BCn 为1，软件写1清该位.
[22]	**CRLRI1**	**CRLR1锁定标志位** 当PWM 组输入通道1发生上升沿变化时，CRLR1 锁存PWM 向下计数器的值，该位由硬件置位. 如果BCn 为0，软件写0清该位，如果BCn 为1，软件写1清该位.
[5]	**Reserved**	保留
[20]	**CAPIF1**	**通道 1捕捉中断指示标志** 如果PWM 组通道1上升锁存中断使能(CRL_IE1=1)，PWM 组通道1发生上升沿转变会使CAPIF1 为高；同样， 如果PWM 组通道1下降锁定中断使能(CFL_IE1=1)，下降沿会使CAPIF1 置高. 写1 清该位为0
[19]	**CAPCH1EN**	**通道1捕捉功能使能位** 0 = 禁用PWM 组通道1的捕捉功能 1 = 使能PWM 组通道1的捕捉功能. 当使能时， 捕捉功能将锁存PWM 计数器的值，并存储到CRLR（上升锁存）或CFLR（下降锁存). 当禁用时，捕捉器不更新CRLR 或CFLR，并关闭通道1中断.
[18]	**CFL_IE1**	**通道1下降沿锁存中断使能** 0 = 禁用下降沿锁存中断 1 = 使能下降沿锁存中断

		使能时，如果检测到PWM 组通道1有下降沿转变，捕捉中断产生.
[17]	**CRL_IE1**	**通道1上升沿锁存中断使能** 0 = 禁用上升沿锁存中断 1 = 使能上升沿锁存中断 使能时，如果检测到PWM 组通道1有上升沿转变，捕捉中断产生.
[16]	**INV1**	**通道1反转使能** 0 = 反转关闭 1 = 反转打开 来自GPIO 引脚的输入信号反转之后再输入到捕捉定时器
[15:8]	**Reserved**	保留.
[7]	**CFLRI0**	**CFLR0 锁存指示** 当PWM 组输入通道0发生下降沿转变时，CFLR0 锁存PWM 向下计数器的值，该位由硬件置位. 如果BCn 为0，软件写0清该位，如果BCn 为1，软件写1清该位
[6]	**CRLRI0**	**CRLR0 锁存指示** 当 PWM 组输入通道 0 发生上升沿变化时，CRLR0 锁定 PWM 向下计数器的值，硬件设置该位。 如果 BCn 位为 0，软件写 0 清该位；如果 BCn 位 为 1，软件写 1 清该位。
[5]	**Reserved**	保留.
[4]	**CAPIF0**	**通道0捕捉中断标志位** 如果PWM 组通道0 上升沿锁定中断使能 (CRL_IE0=1)，PWM 通道 0 发生上升沿转变会使得 CAPIF0 为高；同样，如果PWM 组通道0 下降沿锁定中断使能 (CFL_IE0=1)，下降沿转变会使得 CAPIF0 为高。 写 1 清除该位为 0。
[3]	**CAPCH0EN**	**通道 0 捕捉功能使能** 0 =禁用 PWM 组通道0的捕捉功能。 1 =使能 PWM 组通道0的捕捉功能 当使能时，捕捉功能将锁定 PWM 计数器，并存储到 CRLR （上升锁存）和 CFLR（下降锁存）。 当禁用时，捕捉器不更新 CRLR 和 CFLR，并禁用 PWM 组通道 0 中断。
[2]	**CFL_IE0**	**通道 0 下降锁定中断使能** 0 = 禁用下降沿锁存中断 1 = 使能下降沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 0 有下降沿转变，捕捉产生中断
[1]	**CRL_IE0**	**通道 0 上升锁定中断使能** 0 = 禁用上升沿锁存中断 1 = 使能上升沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 0 有上升沿转变，捕捉产生中断。
[0]	**INV0**	**通道 0 反向使能** 0 = 反向禁用 1 = 反向使能。将GPIO 的输入信号反向后再给捕捉定时器

捕捉控制寄存器(CCR2)

寄存器	偏移地址	R/W	描述	复位值
CCR2	PWMA_BA+0x54 PWMB_BA+0x54	R/W	PWM 捕捉控制寄存器 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CFLRI3	CRLRI3	Reserved	CAPIF3	CAPCH3EN	CFL_IE3	CRL_IE3	INV3
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CFLRI2	CRLRI2	Reserved	CAPIF2	CAPCH2EN	CFL_IE2	CRL_IE2	INV2

位	描述	
[31:24]	**Reserved**	保留
[23]	**CFLRI3**	**CFLR3锁定标志位** 当 PWM 组输入通道 3 发生下降沿变化时，CFLR3 锁定 PWM 向下计数器的值，硬件设置该位。 如果 BCn 位为 0，软件写 0 清该位；如果 BCn 位 为 1，软件写 1 清该位。
[22]	**CRLRI3**	**CRLR3 锁定标志位** 当 PWM 组输入通道 3 发生上升沿变化时，CRLR3 锁定 PWM 向下计数器的值，硬件设置该位。 如果 BCn 位为 0，软件写 0 清该位；如果 BCn 位 为 1，软件写 1 清该位。
[21]	**Reserved**	保留
[20]	**CAPIF3**	**通道 3 捕捉中断标志位** 如果PWM 组通道3 上升沿锁定中断使能 (CRL_IE3=1)，PWM 通道 3 发生上升沿转变会使得 CAPIF3 为高；同样，如果PWM 组通道3 下降沿锁定中断使能 (CFL_IE3=1)，下降沿转变会使得 CAPIF3 为高。 写 1 清除该位为 0。
[19]	**CAPCH3EN**	**通道 3 捕捉功能使能** 0 = 禁用 PWM 组通道3的捕捉功能 1 = 使能 PWM 组通道3的捕捉功能 当使能时，捕捉功能将锁定 PWM 计数器，并存储到 CRLR （上升锁存）和 CFLR（下降锁存）。 当禁用时，捕捉器不更新 CRLR 和 CFLR，并禁用 PWM 组通道 3 中断。
[18]	**CFL_IE3**	**通道 3 下降锁存中断使能位** 0 = 禁用下降沿锁存中断 1 = 使能下降沿锁存中断

		当使能时，如果捕捉器检测到 PWM 组通道 3 有下降沿转变，捕捉产生中断
[17]	**CRL_IE3**	**通道 3 上升锁存中断使能位** 0 = 禁用上升沿锁存中断 1 = 使能上升沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 3 有上升沿转变，捕捉产生中断
[16]	**INV3**	**通道 3 反向使能** 0 = 反向禁用 1 = 反向使能. 将GPIO 的输入信号反向后再给到捕捉定时器
[15:8]	**Reserved**	保留
[7]	**CFLRI2**	**CFLR2锁存标志位** 当 PWM 组输入通道 2 发生下降沿变化时，CFLR2 锁定 PWM 向下计数器的值，硬件设置该位。 如果 BCn 位为 0，软件写 0 清该位；如果 BCn 位 为 1，软件写 1 清该位。
[6]	**CRLRI2**	**CRLR2锁存标志位** 当 PWM 组输入通道 2 发生上升沿变化时，CRLR2 锁定 PWM 向下计数器的值，硬件设置该位。 如果 BCn 位为 0，软件写 0 清该位；如果 BCn 位 为 1，软件写 1 清该位。
[5]	**Reserved**	保留
[4]	**CAPIF2**	**通道 2 捕捉中断标志位** 如果PWM 组通道2 上升沿锁定中断使能 (CRL_IE2=1)，PWM 通道 2 发生上升沿转变会使得 CAPIF2 为高；同样，如果PWM 组通道2 下降沿锁定中断使能 (CFL_IE2=1)，下降沿转变会使得 CAPIF2 为高。 写 1 清除该位为 0。
[3]	**CAPCH2EN**	**通道 2 捕捉功能使能位** **0 =** 禁用 PWM 组通道2的捕捉功能 **1 =** 使能 PWM 组通道2的捕捉功能 当使能时，捕捉功能将锁定 PWM 计数器，并存储到 CRLR （上升锁存）和 CFLR（下降锁存）。 当禁用时，捕捉器不更新 CRLR 和 CFLR，并禁用 PWM 组通道 2 中断。
[2]	**CFL_IE2**	**通道 2 下降锁存中断使能位** 0= 禁用下降沿锁存中断 1 = 使能下降沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 2 有下降沿转变，捕捉产生中断
[1]	**CRL_IE2**	**通道 2 上升锁存中断使能位** 0 = 禁用上升沿锁存中断 1 = 使能上升沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 2 有上升沿转变，捕捉产生中断.
[0]	**INV2**	**通道 2 反向使能位** 0 = 反向禁用 1 = 反向使能 将GPIO 的输入信号反向后再给捕捉定时器

捕捉上升沿锁存寄存器3-0 (CRLR3-0)

寄存器	偏移地址	R/W	描述	复位值
CRLR0	PWMA_BA+0x58 PWMB_BA+0x58	R	PWM 捕捉上升沿锁存寄存器 (通道 0)	0x0000_0000
CRLR1	PWMA_BA+0x60 PWMB_BA+0x60	R	PWM 捕捉上升沿锁存寄存器 (通道1)	0x0000_0000
CRLR2	PWMA_BA+0x68 PWMB_BA+0x68	R	PWM 捕捉上升沿锁存寄存器 (通道2)	0x0000_0000
CRLR3	PWMA_BA+0x70 PWMB_BA+0x70	R	PWM 捕捉上升沿锁存寄存器 (通道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**
CRLRx [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CRLRx [7:0]							

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**CRLRx**	**捕捉上升沿锁存寄存器** 当通道 0/1/2/3 有上升沿转变时，锁存 PWM 计数。

捕捉下降沿锁存寄存器3-0 (CFLR3-0)

寄存器	偏移地址	R/W	描述	复位值
CFLR0	PWMA_BA+0x5C PWMB_BA+0x5C	R	PWM捕捉下降沿锁存寄存器（通道 0）	0x0000_0000
CFLR1	PWMA_BA+0x64 PWMB_BA+0x64	R	PWM捕捉下降沿锁存寄存器（通道 1）	0x0000_0000
CFLR2	PWMA_BA+0x6C PWMB_BA+0x6C	R	PWM捕捉下降沿锁存寄存器（通道 2）	0x0000_0000
CFLR3	PWMA_BA+0x74 PWMB_BA+0x74	R	PWM捕捉下降沿锁存寄存器（通道 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**
CFLRx [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFLRx [7:0]							

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**CFLRx**	**捕捉下降沿锁存寄存器** 当通道 0/1/2/3 有下降沿转变时，锁存 PWM 计数

捕捉输入使能寄存器(CAPENR)

寄存器	偏移地址	R/W	描述	复位值
CAPENR	PWMA_BA+0x78 PWMB_BA+0x78	R/W	PWM 捕捉输入 0~3 使能寄存器	0x0000_0000

31	30	29	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				**CINEN3**	**CINEN2**	**CINEN1**	**CINEN0**

位	描述	
[31:4]	**Reserved**	保留
[3]	**CINEN3**	**通道 3 捕捉输入使能位** 0 = PWM 通道 3捕捉输入路径禁止。PWM通道3捕捉功能输入总当作是0。 1 = PWM 通道 3捕捉输入路径使能。 PWM通道3捕捉功能输入使能，需要相关GPIO的多功能设置成PWM3。
[2]	**CINEN2**	**通道 2捕捉输入使能位** 0 = PWM 通道 2捕捉输入路径禁止。PWM通道2捕捉功能输入总当作是0。 1 = PWM 通道 2捕捉输入路径使能。 PWM通道2捕捉功能输入使能，需要相关GPIO的多功能设置成PWM2。
[1]	**CINEN1**	**通道 1捕捉输入使能位** 0 = PWM 通道 1捕捉输入路径禁止。PWM通道1捕捉功能输入总当作是0。 1 = PWM 通道 1捕捉输入路径使能。 PWM通道1捕捉功能输入使能，需要相关GPIO的多功能设置成PWM1。
[0]	**CINEN0**	**通道 0捕捉输入使能位** 0 =PWM 通道 0捕捉输入路径禁止。PWM通道0捕捉功能输入总当作是0 1 = PWM 通道 0捕捉输入路径使能。 PWM通道0捕捉功能输入使能，需要相关GPIO的多功能设置成PWM0。

PWM输出使能寄存器(POE)

寄存器	偏移地址	R/W	描述	复位值
POE	PWMA_BA+0x7C PWMB_BA+0x7C	R/W	PWM 通道 0~3输出使能寄存器	0x0000_0000

31	30	29	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				**POE3**	**POE2**	**POE1**	**POE0**

位	描述	
[31:4]	**Reserved**	保留
[3]	**POE3**	**通道 3 输出使能位** 0 =禁止 PWM 通道 3 输出 1 =使能 PWM 通道 3 输出 **注意**：相应的 GPIO 管脚必须切换到 PWM 功能
[2]	**POE2**	**通道 2 输出使能位** 0 =禁止 PWM 通道 2 输出 1 =使能 PWM 通道 2 输出 **注意**：相应的 GPIO 管脚必须切换到 PWM 功能
[1]	**POE1**	**通道 1 输出使能位** 0 =禁止 PWM 通道1 输出 1 =使能 PWM 通道1 输出 **注意**：相应的 GPIO 管脚必须切换到 PWM 功能
[0]	**POE0**	**通道 0输出使能位** 0 =禁止 PWM 通道 0 输出 1 =使能 PWM 通道 0 输出 **注意**：相应的 GPIO 管脚必须切换到 PWM 功能

PWM 触发控制寄存器 (TCON)

寄存器	偏移地址	R/W	描述	复位值
TCON	PWMA_BA+0x80 PWMB_BA+0x80	R/W	PWM 通道0~3触发控制寄存器	0x0000_0000

31	30	29	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				PWM3TEN	PWM2TEN	PWM1TEN	PWM0TEN

位	描述	
[31:4]	**Reserved**	保留.
[3]	**PWM3TEN**	**通道 3 中心对齐触发使能位** 0 = PWM通道 3触发 ADC功能禁止 1 = PWM通道 3触发 ADC功能使能 如果该位置1，当PWM计数器上数到CNR时，PWM可以触发ADC开始转换 **注意:** 该功能仅支持PWM工作在中心对齐模式
[2]	**PWM2TEN**	**通道 2中心对齐触发使能位** 0 = PWM通道 2触发 ADC功能禁止 1 = PWM通道 2触发 ADC功能使能 如果该位置1，当PWM计数器上数到CNR时，PWM可以触发ADC开始转换 **注意:** 该功能仅支持PWM工作在中心对齐模式.
[1]	**PWM1TEN**	**通道 1中心对齐触发使能位** 0 = PWM通道 1触发 ADC功能禁止 1 = PWM通道 1触发 ADC功能使能 如果该位置1，当PWM计数器上数到CNR时，PWM可以触发ADC开始转换 **注意:** 该功能仅支持PWM工作在中心对齐模式..
[0]	**PWM0TEN**	**通道 0中心对齐触发使能位** 0 = PWM通道 0触发 ADC功能禁止. 1 = PWM通道 0触发 ADC功能使能 如果该位置1，当PWM计数器上数到CNR时，PWM可以触发ADC开始转换 **注意:** 该功能仅支持PWM工作在中心对齐模式...

PWM 触发状态寄存器 (TSTATUS)

寄存器	偏移地址	R/W	描述	复位值
TSTATUS	PWMA_BA+0x84 PWMB_BA+0x84	R/W	PWM 触发状态寄存器	0x0000_0000

31	30	29	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				PWM3TF	PWM2TF	PWM1TF	PWM0TF

位	描述	
[3]	PWM3TF	**通道 3中心对齐触发标志** 在中心对齐方式，如果PWM3TEN置1， 当PWM计数器上数到CNR，该位被硬件置1。如果ADC的触发源选择PWM，在该位置1后，ADC开始转换。 软件写1清0该位
[2]	PWM2TF	**通道 2中心对齐触发标志** 在中心对齐方式，如果PWM2TEN置1， 当PWM计数器上数到CNR，该位被硬件置1。如果ADC的触发源选择PWM，在该位置1后，ADC开始转换。 软件写1清0该位
[1]	PWM1TF	**通道 1中心对齐触发标志** 在中心对齐方式，如果PWM1TEN置1， 当PWM计数器上数到CNR，该位被硬件置1。如果ADC的触发源选择PWM，在该位置1后，ADC开始转换。 软件写1清0该位
[0]	PWM0TF	**通道 0中心对齐触发标志** 在中心对齐方式，如果PWM0TEN置1， 当PWM计数器上数到CNR，该位被硬件置1。如果ADC的触发源选择PWM，在该位置1后，ADC开始转换。 软件写1清0该位

PWM0 同步忙状态寄存器 (SYNCBUSY0)

寄存器	偏移地址	R/W	描述	复位值
SYNCBUSY0	PWMA_BA+0x88 PWMB_BA+0x88	R	PWM0 同步忙状态寄存器	0x0000_0000

31	30	29	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							S_BUSY

位	描述	
[31:1]	Reserved	保留.
[0]	S_BUSY	**PWM 同步忙** 当软件写CNR0/CMR0/PPR或切换PWM0 工作模式 (PCR[3])，PWM会有一个忙的时间来完全更新这些值，因为PWM时钟可能和系统时钟范围不一样，软件需要写CNR0/CMR0/PPR或切换PWM0 工作模式 (PCR[3])之前检查忙状态，确保之前的设置完全更新。 当软件写CNR0/CMR0/PPR或切换PWM0 工作模式 (PCR[3]) 时，该位会置1。当PWM完全更新这些值，该位会被硬件自动清0。

PWM1同步忙状态寄存器(SYNCBUSY1)

寄存器	偏移地址	R/W	描述	复位值
SYNCBUSY1	PWMA_BA+0x8C PWMB_BA+0x8C	R	PWM1同步忙状态寄存器	0x0000_0000

31	30	29	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							S_BUSY

位	描述	
[31:1]	Reserved	保留
[0]	S_BUSY	**PWM 同步忙** 当软件写CNR1/CMR1/PPR或切换PWM1 工作模式 (PCR[11])， PWM会有一个忙的时间来完全更新这些值，因为PWM时钟可能和系统时钟范围不一样，软件需要写CNR1/CMR1/PPR或切换PWM1 工作模式 (PCR[11])之前检查忙状态，确保之前的设置完全更新。 当软件写CNR1/CMR1/PPR或切换PWM1 工作模式 (PCR[11]) 时，该位会置1。当PWM完全更新这些值，该位会被硬件自动清0。.

PWM2同步忙状态寄存器(SYNCBUSY2)

寄存器	偏移地址	R/W	描述	复位值
SYNCBUSY2	PWMA_BA+0x90 PWMB_BA+0x90	R	PWM2 同步忙状态寄存器	0x0000_0000

31	30	29	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							S_BUSY

位	描述	
[31:1]	Reserved	保留
[0]	S_BUSY	**PWM 同步忙** 当软件写CNR2/CMR2/PPR或切换PWM2 工作模式 (PCR[19])， PWM会有一个忙的时间来完全更新这些值，因为PWM时钟可能和系统时钟范围不一样，软件需要写CNR2/CMR2/PPR或切换PWM2 工作模式 (PCR[19])之前检查忙状态，确保之前的设置完全更新。 当软件写CNR2/CMR2/PPR或切换PWM2 工作模式 (PCR[19]) 时，该位会置1。当PWM完全更新这些值，该位会被硬件自动清0。

PWM3同步忙状态寄存器(SYNCBUSY3)

寄存器	偏移地址	R/W	描述	复位值
SYNCBUSY3	PWMA_BA+0x94 PWMB_BA+0x94	R	PWM3 同步忙状态寄存器	0x0000_0000

31	30	29	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							S_BUSY

位	描述	
[31:1]	Reserved	保留
[0]	S_BUSY	**PWM 同步忙** 当软件写CNR3/CMR3/PPR或切换PWM3 工作模式 (PCR[27])， PWM会有一个忙的时间来完全更新这些值，因为PWM时钟可能和系统时钟范围不一样，软件需要写CNR3/CMR3/PPR或切换PWM3 工作模式 (PCR[27])之前检查忙状态，确保之前的设置完全更新。 当软件写CNR3/CMR3/PPR或切换PWM3 工作模式 (PCR[27]) 时，该位会置1。当PWM完全更新这些值，该位会被硬件自动清0。.

5.10 看门狗定时器 (WDT)

5.10.1 概述

设计看门狗定时器的目的是，当系统运行到一个未知状态时，通过它来使系统复位。这种做法可以预防系统进入到无限期的死循环。此外，该看门狗定时器支持系统从Idle/Power-down模式唤醒功能。

5.10.2 特性

- 18位的看门狗定时器可满足用户溢出时间间隔要求
- 溢出时间间隔(24 ~ 218)个WDT_CLK时钟周期可选，如WDT_CLK = 10 kHz，那么溢出时间间隔是104 ms ~ 26.3168 s
- 系统复位保持时间(1 / WDT_CLK) * 63
- 支持看门狗定时器复位延时周期
  - 可选的复位延时周期包括(1026、130、18 or 3) * WDT_CLK个复位延时周期
- 当CWDTEN (CONFIG0[31] 看门狗使能)位被置为0，支持芯片上电或复位条件下看门狗强制打开。
- 支持看门狗定时器溢出唤醒功能，此时时钟源必须选择内部低速10k时钟源

5.10.3 框图

看门狗定时器时钟控制和框图如下：

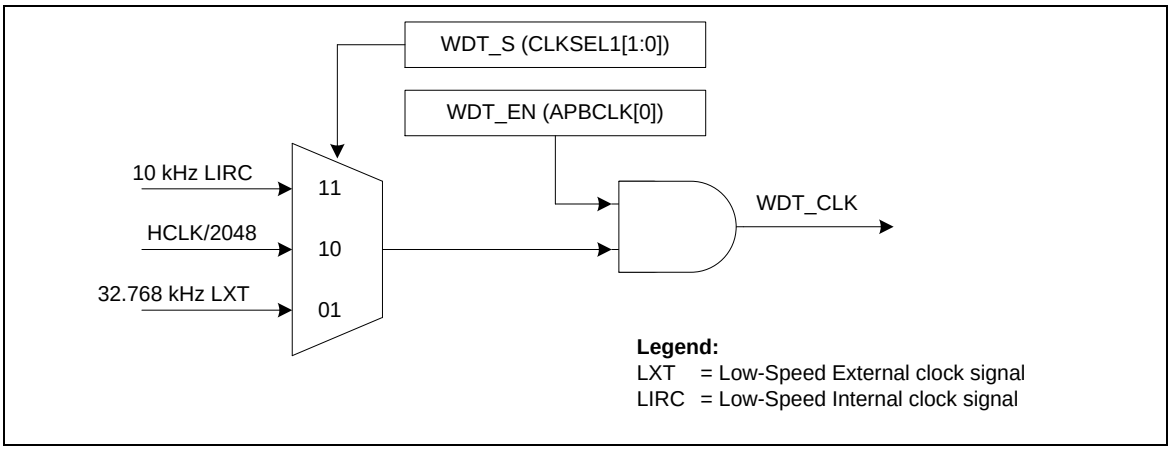


图 5-50 看门狗定时器时钟控制

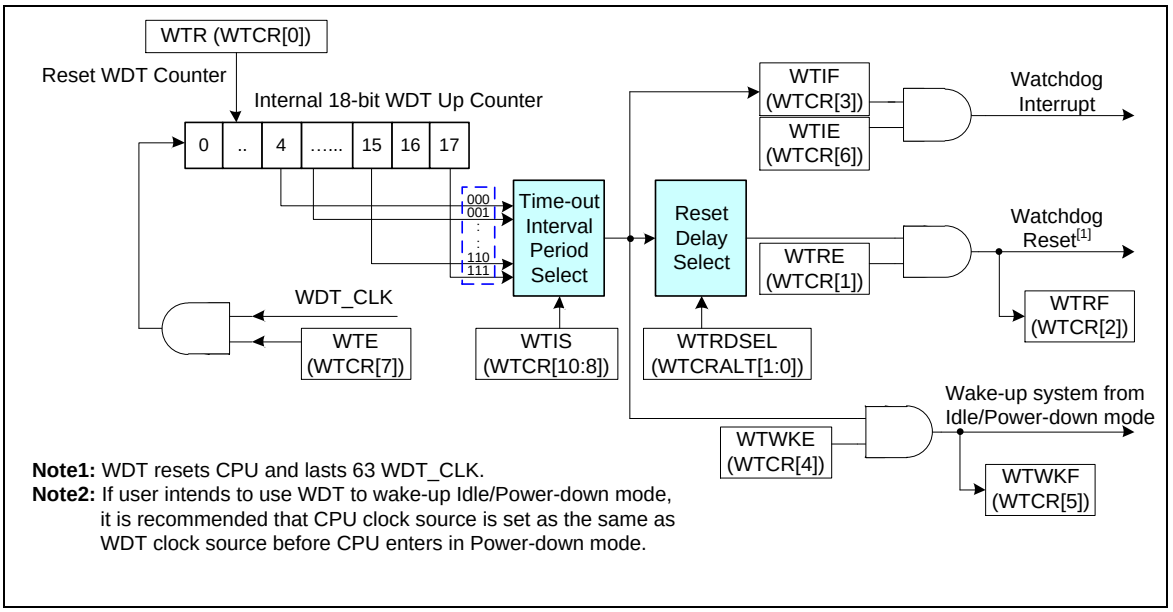


图 5-51 看门狗定时器框图

5.10.4 基本配置

看门狗定时器的时钟在APBCLK[0]位使能，时钟源选择在CLKSEL1[1:0]设置

或者用户也可以烧录时把CONFIG0[31]配置为0，从而在芯片上电或RESET后强制看门狗定时器和10k时钟源使能。

5.10.5 功能描述

看门狗定时器(WDT)包含一个18位的可设置溢出时间间隔的向上计数器。表5-29显示了WDT溢出时间间隔选择，图6-154看门狗定时器溢出时间间隔和复位周期时序显示了WDT溢出时间间隔和复位周期时序。

- WDT定时溢出中断

对WTE置位可以使能WDT功能，然后WDT计数器向上计数。通过设置WTIS有8个不同的溢出时间间隔可以选择。当WDT计数器计到WTIS设定值，WDT产生溢出中断，然后WTIF标志立即被置1.

- WDT复位延时周期和复位系统

WTIF标志被置位后，还会有一个固定的延时周期T_{RSTD}, 用户应在T_{RSTD} 延时计完之前对WTR置1来复位18位的WDT向上计数器的值，以防止产生WDT时间溢出复位信号。如T_{RSTD} 时间计满以后，WDT向上计数器的值仍没有被清除，若WTRE位为1，则WTRF标志会被置1，然后芯片立即复位。请参考图6-154看门狗定时器时间溢出间隔和复位周期时序， T_{RST}复位周期将保持至少63个WDT时钟周期，然后芯片从复位向量地址(0x0000_0000)重新开始执行程序。芯片由WDT定时溢出复位后，WTRF标志将会保持1。用户可以通过检查该标志来确认系统是否因WDT定时溢出复位。

- 看门狗唤醒

如果看门狗时钟源选择为内部10k, 如果WTWKE位被使能，看门狗定时溢出中断产生后，系统能从Power-down模式下被唤醒。同时，WTWKF标志会被自动置1。用户可以通过查询WTWKF标志知道系统是否是被看门狗定时溢出中断唤醒。

WTIS	定时溢出中断间隔 T_{TIS}	复位延时周期 T_{RSTD}
000	$2^4 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
001	$2^6 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
010	$2^8 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
011	$2^{10} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
100	$2^{12} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
101	$2^{14} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
110	$2^{16} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
111	$2^{18} * T_{WDT}$	(3/18/130/1026) * T_{WDT}

表 5-8 看门狗定时器定时溢出间隔周期

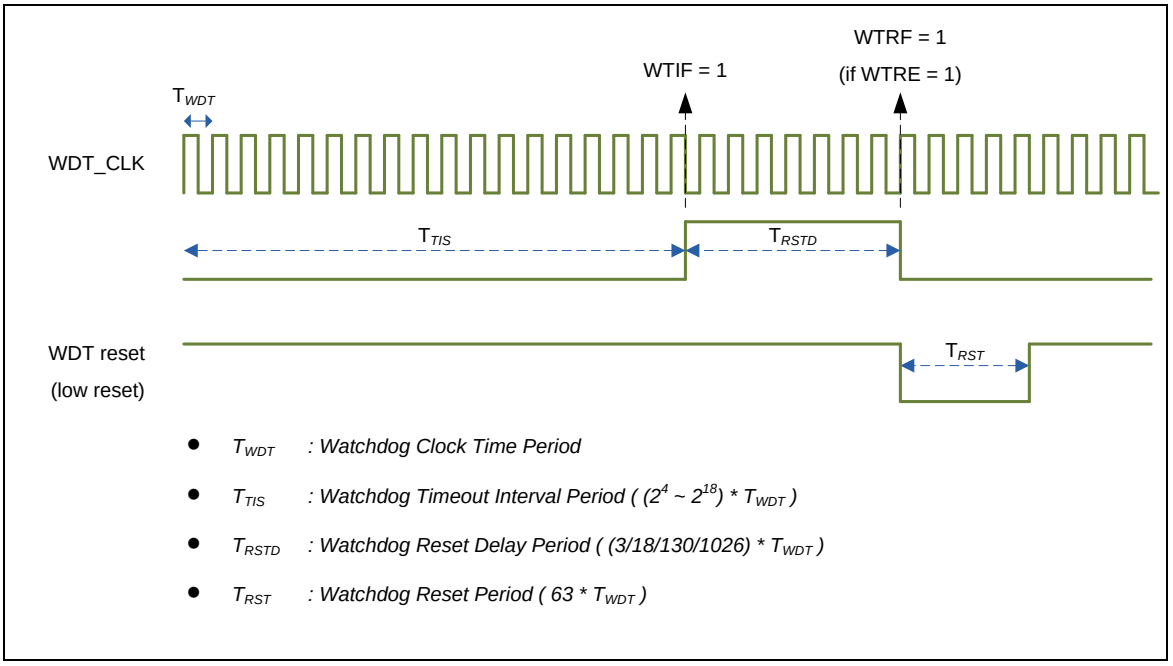


图 5-52 看门狗定时器定时溢出间隔和复位周期时序图

5.10.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移地址	R/W	描述	复位值
WDT 基地址: **WDT_BA = 0x4000_4000**				
WTCR	WDT_BA+0x00	R/W	看门狗定时器控制寄存器	0x0000_0700
WTCRALT	WDT_BA+0x04	R/W	看门狗定时器选择控制寄存器	0x0000_0000

5.10.7 寄存器描述

看门狗定时器控制寄存器(WTCR)

寄存器	偏移地址	R/W	描述	复位值
WTCR	WDT_BA+0x00	R/W	看门狗定时器控制寄存器	0x0000_0700

31	30	29	28	27	26	25	24
DBGACK_WDT	保留						
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留					WTIS		
7	6	5	4	3	2	1	0
WTE	WTIE	WTWKF	WTWKE	WTIF	WTRF	WTRE	WTR

位	描述	
[31]	**DBGACK_WDT**	**ICE 调试模式下看门狗计数控制位(写保护)** 0 = ICE 调试模式影响看门狗计数，当ICE让CPU运行停止，看门狗计数器也停止. 1 = ICE 调试模式不影响看门狗计数。不论CPU受ICE停止还是运行，看门狗继续计数
[30:11]	**保留**	**保留**.
[10:8]	**WTIS**	**看门狗定时器溢出时间间隔选择 (写保护)** 以下三个位用于选择看门狗定时溢出周期 000 = 2^4 *T_{WDT}. 001 = 2^6 * T_{WDT}. 010 = 2^8 * T_{WDT}. 011 = 2^{10} * T_{WDT}. 100 = 2^{12} * T_{WDT}. 101 = 2^{14} * T_{WDT}. 110 = 2^{16} * T_{WDT}. 111 = 2^{18} * T_{WDT}.
[7]	**WTE**	**看门狗定时器使能位 (写保护)** 0 =看门狗定时器禁止. (该操作会恢复看门狗计数器的值.) 1 =看门狗定时器使能. **注意:**如果CWDTEN (CONFIG0[31] 看门狗使能位) 位被清零, WTE位会被强制置1，并且用户不能把它改为0.
[6]	**WTIE**	**看门狗定时器定时溢出中断使能(写保护)** 如果该位使能,看门狗定时溢出后会产生中断信号，并告知CPU 0 = 看门狗定时溢出中断禁止.

		1 = 看门狗定时溢出中断使能.
[5]	**WTWKF**	**看门狗定时器溢出唤醒标志** 该位表明看门狗定时器中断唤醒标志的状态 0 = 看门狗定时器没有导致芯片唤醒 1 = 芯片从Idle或Power-down 模式被唤醒 **注意:** 该位为写1清零.
[4]	**WTWKE**	**看门狗定时器定时溢出唤醒功能控制位 (写保护)** 如果此位被置1，当WTIF被置1且WTIE被使能，看门狗定时溢出中断信号将会触发唤醒MCU. 0 = 唤醒触发事件禁止，即便看门狗定时器定时溢出中断发生 1 = 唤醒触发事件使能，如果看门狗定时溢出中断产生，将唤醒MCU **注意: MCU能被看门狗定时器定时溢出中断信号唤醒的条件是看门狗定时器时钟源必须为内部10k时钟**
[3]	**WTIF**	**看门狗定时器定时溢出标志** 当看门狗定时器计数器的值达到溢出时间间隔，该位将被置1 0 = 没有发生看门狗定时溢出中断 1 = 有看门狗定时溢出中断发生. **注意:** 该位写1清零
[2]	**WTRF**	**看门狗定时器复位标志** 该位用来指示系统是否因看门狗定时器定时溢出发生复位 0 = 没有发生看门狗定时溢出复位 1 = 发生了看门狗定时溢出复位. **注意:** 该位写1清零.
[1]	**WTRE**	**看门狗定时器复位使能位 (写保护)** 如果看门狗计数器的值达到设定值，且固定的WDT复位时间已经计完，还没有被清零，如果该位被设置，则会使能看门狗定时溢出复位 0 = 看门狗定时溢出复位功能禁止. 1 =看门狗定时溢出复位功能使能
[0]	**WTR**	**清看门狗计数器 (写保护)** 0 = 不影响. 1 =18位看门狗定时器计数值复位清零. **注意:** 该位硬件自动清零.

看门狗定时器选择控制寄存器 (WTCRALT)

寄存器	偏移地址	R/W	描述	复位值
WTCRALT	WDT_BA+0x04	R/W	看门狗定时器选择控制寄存器	0x0000_0000

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

位	描述	
[31:2]	**保留**	**保留**.
[1:0]	**WTRDSEL**	**看门狗定时器复位延时选择 (写保护)** 当看门狗定时器发生溢出，在看门狗复位延时的时段内可以将看门狗计数器清零，以防止看门狗溢出复位. 对不同的看门狗溢出时间，用户也可以选择适当的看门狗复位延时时间 这两个位是保护位.写这两个位需要对地址0x5000_0100写“59h”, “16h”, “88h” 来解锁。详细请参考寄存器REGWRPROT，地址GCR_BA+0x100. 00 = 看门狗定时器复位延时时间是1026 * WDT_CLK. 01 =看门狗定时器复位延时时间是130 * WDT_CLK. 10 =看门狗定时器复位延时时间是18 * WDT_CLK. 11 =看门狗定时器复位延时时间是3 * WDT_CLK. **注意:** 如果看门狗定时器超时复位发生，该寄存器的值将被清零

5.11 窗口看门狗定时器(WWDT)

5.11.1 预览

窗口看门狗定时器用于在一个窗口时间内执行系统复位，以防止程序在不可预知条件下跑到一个不可控的状态

5.11.2 特性

- 6位向下计数值(WWDTVAL[5:0]) 和6位比较窗口值(WWDTCR[21:16]) ，使得窗口周期更加灵活
- 用4位值可以选择16种看门狗预分频值，预分频器由11位组成，因此最大可除以2048分配

5.11.3 框图

窗口看门狗定时器 的时钟控制和框图如下

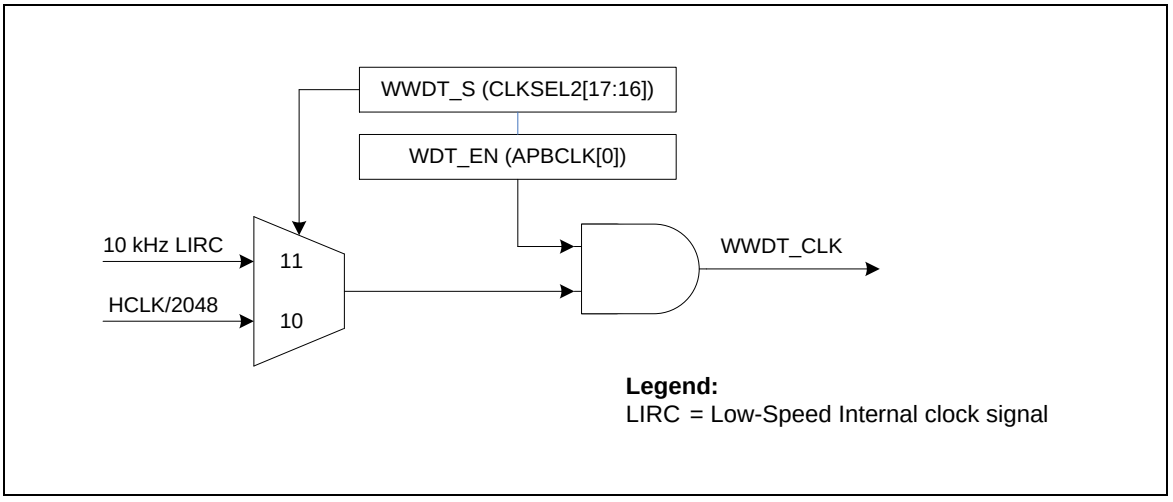


图 5-53 窗口看门狗定时器时钟控制

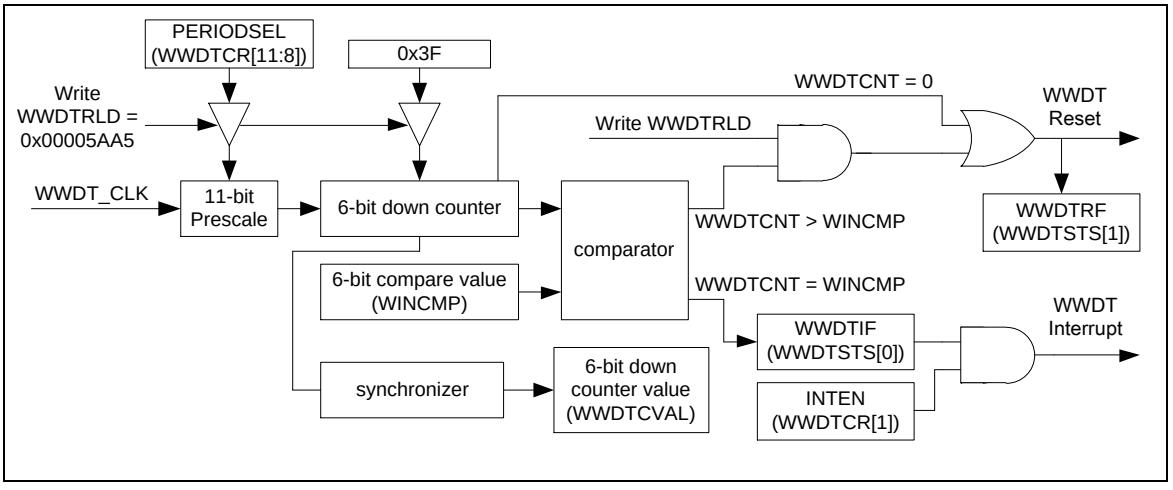


图 5-54 窗口看门狗定时器时钟框图

5.11.4 基本配置

窗口看门狗定时器外设时钟源通过APBCLK[0]来使能，通过CLKSEL2[17:16]来选择.

5.11.5 功能描述

窗口看门狗定时器(WWDT)是一个 6位向下计数器，该计数器带一个可选择预分频值，不同的预分频值对应不同的看门狗定时溢出时间

6位窗口看门狗定时器的时钟源可以是系统时钟经2048 (HCLK/2048)分频的时钟，也可以是内部10k低速RC振荡时钟源，看门狗的时钟源带一个可选择的11位预分频值，该值可通过PERIODSEL (WWDTCRL[11:8]) 位来设置选择，对应预分频值如下表

周期选择	预分频值	最大定时溢出间隔	最大定时溢出间隔 (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

表 5-9 窗口看门狗定时器 预分频值选择

- 窗口看门狗定时器的计数

当WWDTEN位被使能，窗口看门狗向下计数器将会从0x3F向下递减计数到0. 为了防止程序在非用户指定位置关闭窗口看门狗定时器，窗口看门狗定时器控制寄存器WWDTCR在芯片上电或复位后仅可写一次。当WWDTEN (WWDTCR[0])位被软件使能以后，用户不能禁止窗口看门狗定时器(WWDTEN)，修改计数器预分频周期(PERIODSEL)，或修改窗口比较值 (WINCMP)，除非芯片复位。

- 窗口看门狗定时器比较中断

窗口看门狗定时器向下计数过程中，当窗口看门狗定时器计数值(WWDTCVAL) 等于比较值WINCMP，WWDTIF会被置1，并且WWDTIF 可以被软件清零;如果WWDTIE位被使能，当WWDTIF位被硬件置1，就会产生窗口看门狗比较匹配中断。

- 窗口看门狗定时器复位系统

当WWDTIF产生后，用户必须通过对WWDTRLD写0x00005AA5进行重载回到初始值0x3F，可以防止窗口看门狗定时器的值继续向下计数到0，从而产生窗口看门狗定时器复位系统信号通知系统复位。

如果寄存器WWDTCVAL的当前值大于比较寄存器WINCMP的值，用户对WWDTRLD寄存器写0x00005AA5，窗口看门狗定时器复位系统信号将会产生，并导致芯片复位。

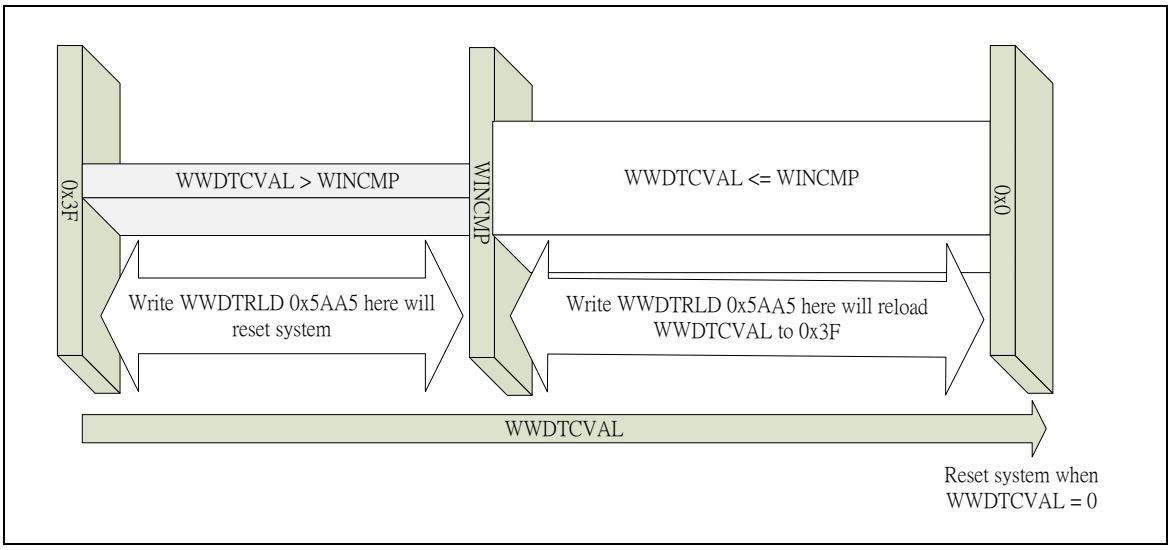


图 5-55 窗口看门狗定时器复位和重载过程

- 窗口看门狗定时器的窗口设置限制

当用户对WWDTRLD寄存器写0x00005AA5来重载WWDT的值到0x3F的时候，大概需要3个窗口看门狗定时器时钟来同步重载命令到实际执行重载操作。这意味着如果用户设置PERIODSEL的值为0000，即计数器预分频值设置为1，WINCMP寄存器的值必须大于2；否则，当WWDTIF标志产生后，写WWDTRLD来重载窗口看门狗定时器的值为0x3F操作无效，将会一直出现窗口看门狗定时器引起的系统复位。

周期选择	预分频值	有效的WINCMP比较值值
0000	1	0x3 ~ 0x3F
0001	2	0x2 ~ 0x3F
其它	其它	0x0 ~ 0x3F

表 5-10 WINCMP 寄存器设置限制

5.11.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移地址	读/写	描述	复位值

WWDT 基地址: WWDT_BA = 0x4000_4100				
WWDTRLD	WWDT_BA+0x00	W	窗口看门狗定时器装载计数寄存器	0x0000_0000
WWDTCR	WWDT_BA+0x04	R/W	窗口看门狗定时器控制寄存器	0x003F_0800
WWDTSR	WWDT_BA+0x08	R/W	窗口看门狗定时器状态寄存器	0x0000_0000
WWDTCVR	WWDT_BA+0x0C	R	窗口看门狗定时器计数器值寄存器	0x0000_003F

5.11.7 寄存器描述

窗口看门狗定时器重载计数寄存器(WWDTRLD)

寄存器	偏移地址	读/写	描述	复位值
WWDTRLD	WWDT_BA+0x00	W	窗口看门狗定时器重载计数器寄存器	0x0000_0000

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

位	描述	
[31:0]	**WWDTRLD**	窗口看门狗定时器重载计数器寄存器 对此寄存器写0x00005AA5 将会重载窗口看门狗定时器计数器的值到0x3F. **注意:**用户只能是当窗口看门狗计数器的值在0到WINCMP之间才可以写WWDTRLD寄存器来重载窗口看门狗计数器。如果窗口看门狗计数器的值大于WINCMP时写WWDTRLD寄存器，窗口看门狗定时器复位信号仍会立即产生

窗口看门狗定时器控制寄存器 (WWDTCR)

寄存器	偏移地址	读/写	描述	复位值
WWDTCR	WWDT_BA+0x04	R/W	窗口看门狗定时器控制寄存器	0x003F_0800

注意:此寄存器仅当芯片上电或复位后写一次

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

位	描述	
[31]	**DBGACK_WWDT**	ICE 调试模式下窗口看门狗计数控制位 0 =ICE 调试模式下影响窗口看门狗定时器计数，当CPU受ICE停止后，窗口看门狗向下计数器计数值将保持 1 = ICE 调试模式下不影响窗口看门狗定时器计数，窗口看门狗定时器向下计数器将会保持继续计数，不管CPU是否受ICE保持。
[30:22]	**保留**	**保留**.
[21:16]	**WINCMP**	**窗口看门狗定时器窗口比较寄存器** 设置此寄存器来调整有效重载窗口 注意：当当前窗口看门狗定时器计数器的值在0到WINCMP之间时，用户才可以写WWDTRLD来重载窗口看门狗定时器计数器的值。如果当前窗口看门狗定时器计数器的值大于WINCMP时用户写WWDTRLD寄存器，窗口看门狗定时器复位信号仍会立即产生。
[15:12]	**保留**	**保留**.
[11:8]	**PERIODSEL**	**窗口看门狗定时器计数器预分频周期选择** 0000 = 预分频为1; 最大定时溢出周期是 1 * 64 * TWWDT. 0001 =预分频为2; 最大定时溢出周期是2 * 64 * TWWDT. 0010 =预分频为4; 最大定时溢出周期是4 * 64 * TWWDT. 0011 =预分频为8; 最大定时溢出周期是8 * 64 * TWWDT. 0100 =预分频为16; 最大定时溢出周期是16 * 64 * TWWDT. 0101 =预分频为32; 最大定时溢出周期是32 * 64 * TWWDT. 0110 =预分频为64; 最大定时溢出周期是64 * 64 * TWWDT. 0111 =预分频为128; 最大定时溢出周期是128 * 64 * TWWDT. 1000 =预分频为192; 最大定时溢出周期是192 * 64 * TWWDT. 1001 =预分频为256; 最大定时溢出周期是256 * 64 * TWWDT. 1010 =预分频为384; 最大定时溢出周期是384 * 64 * TWWDT.

		1011 =预分频为512; 最大定时溢出周期是512 * 64 * TWWDT. 1100 =预分频为768; 最大定时溢出周期是768 * 64 * TWWDT. 1101 =预分频为1024; 最大定时溢出周期是1024 * 64 * TWWDT. 1110 =预分频为1536; 最大定时溢出周期是1536 * 64 * TWWDT. 1111 =预分频为2048; 最大定时溢出周期是2048 * 64 * TWWDT.
[7:2]	**保留**	**保留**.
[1]	**WWDTIE**	**窗口看门狗定时器中断使能位** 如果该位被使能，如果有窗口看门狗定时器计数器比较匹配中断信号产生，就会通知CPU 0 =窗口看门狗定时器计数器比较匹配中断禁止 1 =窗口看门狗定时器计数器比较匹配中断使能
[0]	**WWDTEN**	**窗口看门狗定时器使能位** 设置该位来使能窗口看门狗定时器计数器计数 0 =窗口看门狗定时器计数器停止. 1 =窗口看门狗定时器计数器计数开始

窗口看门狗定时器状态寄存器 (WWDTSR)

寄存器	偏移地址	读/写	描述	复位值
WWDTSR	WWDT_BA+0x08	R/W	窗口看门狗定时器状态寄存器	0x0000_0000

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

位	描述	
[31:2]	**保留**	**保留**.
[1]	**WWDTRF**	**窗口看门狗定时器溢出复位标志** 该位指示系统是否已被窗口看门狗定时器溢出复位 0 =窗口看门狗定时器没有发生复位 1 =窗口看门狗定时器发生了复位 注意：该位是写1清零
[0]	**WWDTIF**	**窗口看门狗定时器比较匹配中断标志** 该位指示窗口看门狗定时器的比较匹配中断状态标志 0 =窗口看门狗定时器计数器的值与WINCMP寄存器的值不匹配 1 =窗口看门狗定时器计数器的值与WINCMP寄存器的值匹配 注意：该位是写1清零

窗口看门狗定时器计数器值寄存器 (WWDTCVR)

寄存器	偏移地址	读/写	描述	复位值
WWDTCVR	WWDT_BA+0x0C	R	窗口看门狗定时器计数器值寄存器	0x0000_003F

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

位	描述	
[31:6]	**保留**	**保留**.
[5:0]	**WWDTCVAL**	**窗口看门狗定时器计数器值** WWDTCVAL将会被持续更新，以指示6位向下计数器的值

5.12 实时时钟（RTC）

5.12.1 概述

实时时钟 (RTC) 控制器用于记录实时时间及日历等信息。RTC控制器支持可配置的时间节拍和闹钟定时中断。时间及日历等信息由BCD 码格式进行表示。可对外接晶振的频率精度进行数字频率补偿。

RTC控制器也提供80字节的寄存器用于存储用户的重要信息

5.12.2 特征

- 支持时间计数（秒，分，时）和日历计数（日，月，年），用户可以通过访问TLR寄存器用来查看时间及通过访问CLR寄存器查看日历
- 支持闹钟时间（秒，分，时）和日历（日，月，年）设定
- 12-小时或 24-小时模式可选择
- 闰年自动识别
- 一周天数计数器
- 频率补偿寄存器 (FCR)
- 所有时间日期由 BCD 码表示

- 支持周期时间节拍中断，提供 8个周期选项供选择1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 及 1 秒
- 支持 RTC 定时节拍和闹钟定时中断
- 支持RTC中断从空闲模式或掉电模式下唤醒芯片
- 提供80字节的寄存器用于存储用户信息

5.12.3 框图

RTC 模块框图如下：

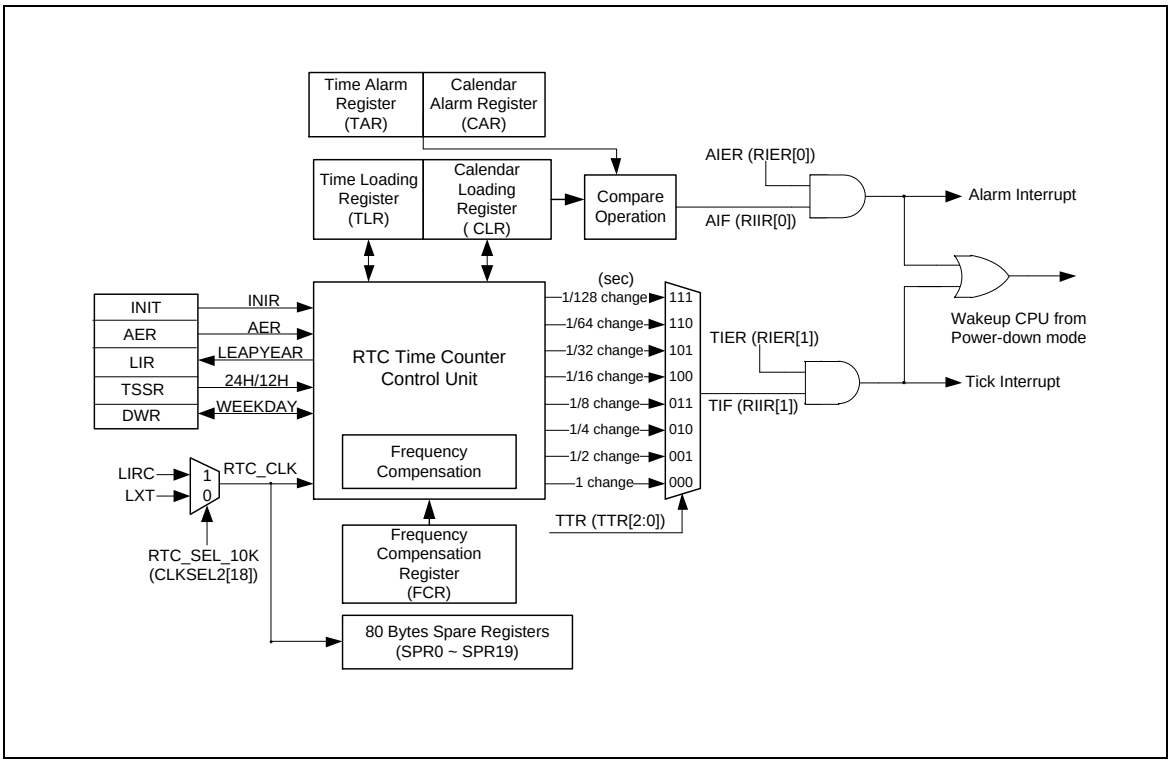


图 5-56 RTC 框图

5.12.4 基本配置

RTC控制器时钟使能（在APBCLK[1]的RTC_EN位）和低速32KHZ晶振使能（在PWRCON[1]的XTL32K_EN位）

5.12.5 功能描述

5.12.5.1 *RTC 初始化*

当 RTC 模块上电后，RTC 处于 reset 状态。用户需要写入数据 (0x a5eb1357) 到 INIR寄存器 (INIR [31:0] RTC 初始化) 让 RTC 离开 reset 状态。一旦 INIR 被写入 0xa5eb1357，RTC 将会永远处于非复位状态。用户可以读计时(INIR[0] RTC 计时状态) 位来判断RTC是否在复位状态。

5.12.5.2 *访问RTC 寄存器*

由于RTC 时钟和系统时钟的不同，当用户对任一 RTC 寄存器进行写入时，必须等待2个 RTC 时钟周期 (大约60us)，寄存器内的值才会被更新。

此外，用户必须保证 RTC 控制器的寄存器（TLR, CLR, TAR 和 CAR）载入数据的合理性。RTC不会对DWR和CLR的内容的合理性进行检查。

5.12.5.3 *RTC读/写使能*

寄存器 AER 位15~0 作为保护RTC 寄存器允许其读/写的密码用来解锁RTC寄存器读/写保护功能。如果AER 位15~0被设置为 0xA965，用户可以读ENF(AER [16] RTC 寄存器访问使能标志)的状态来查询RTC寄存器是否可以访问。一旦ENF位使能，RTC访问使能功能将会在1024个RTC时钟（大约30ms）内有效。ENF位会在1024个RTC时钟之后自动清0。

当ENF为1或0时，RTC控制寄存器的访问属性如下表：

寄存器	ENF=1	ENF=0
INIR	读/写	读/写
AER	读/写	读/写
FCR	读/写	不可访问
TLR	读/写	读
CLR	读/写	读
TSSR	读/写	读/写
DWR	读/写	读
TAR	读/写	不可访问
CAR	读/写	不可访问
LIR	读	读
RIER	读/写	读/写
RIIR	读/写	读/写
TTR	读/写	不可访问
SPRCTL	读/写	不可访问

SPR0-SPR19	读/写	不可访问

5.12.5.4 *频率补偿*

RTC的时钟源可能不是精确的32768 Hz， FCR(频率补偿寄存器) 允许软件对时钟输入进行数字补偿。时钟输入的频率必须在 32776 Hz 到 32761 Hz 范围内。

检测到的整数部分	整数 (FCR[11:8])	检测到的整数部分	整数 (FCR[11:8])
32776	1111	32768	0111
32775	1110	32767	0110
32774	1101	32766	0101
32773	1100	32765	0100
32772	1011	32764	0011
32771	1010	32763	0010
32770	1001	32762	0001
32769	1000	32761	0000

下面为对高于或低于32768 Hz的RTC时钟输入进行补偿的示例。

例子1: (RTC 源时钟> 32768 Hz) **RTC 源时钟测量: 32773.65 Hz (> 32768 Hz)** **整数部分: 32773 => 0x8005** **INTEGER (FCR [11:8] 整数部分) = 0x05 – 0x01 + 0x08 = 0x0c** **小数部分: 0.65** **FRACTION (FCR [5:0] 小数部分) = 0.65 X 60 = 39 = 0x27** **RTC FCR 寄存器应该是 0xC27.**
例子2: (RTC 源时钟≦ 32768 Hz) RTC源时钟测量**:** 32765.27 Hz (≦ 32768 Hz) 整数部分: 32765 => 0x7FFD INTEGER (FCR [11:8] 整数部分) = 0x0D – 0x01 – 0x08 = 0x04 小数部分: 0.27 FRACTION (FCR [5:0] 小数部分) = 0.27 x 60 = 16.2 = 0x10 RTC FCR寄存器应该是0x410.

5.12.5.5 *时间和日历计数*

TLR 和 CLR 用于载入时间和日历。TAR和CAR用于闹钟（时间和日历）设定。

5.12.5.6 *12/24 小时时标格式选择*

根据 TSSR 位0 选择12/24小时时标格式。

24小时制式 (24H_12H = 1)	12小时制式 (24H_12H = 0)	24小时制式 (24H_12H = 1)	12小时制式(PM Time + 20) (24H_12H = 0)
00	12 (AM12)	12	32 (PM12)
01	01 (AM01)	13	21 (PM01)
02	02 (AM02)	14	22 (PM02)
03	03 (AM03)	15	23 (PM03)
04	04 (AM04)	16	24 (PM04)
05	05 (AM05)	17	25 (PM05)
06	06 (AM06)	18	26 (PM06)
07	07 (AM07)	19	27 (PM07)
08	08 (AM08)	20	28 (PM08)
09	09 (AM09)	21	29 (PM09)
10	10 (AM10)	22	30 (PM10)
11	11 (AM11)	23	31 (PM11)

5.12.5.7 *一周日期计数器*

RTC控制器提供一周计日格式寄存器DWR(时间节拍寄存器TTR[2：0])。内部的值由0至6，用于表示周日至周六。

5.12.5.8 *时间周期节拍中断*

通过设置 TTR（TTR[2:0]）来选择周期中断，周期中断有8个选项 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 以及 1 秒。当TIER（RIER[1]时间节拍中断使能）被置 1，周期中断使能后，MCU根据TTR寄存器内设定的值周期性的发生中断。.

5.12.5.9 *闹钟中断*

当 TLR 和CLR中的值等于TAR和CAR中的设定值，如果此时闹钟中断已设为使能 (AIER位RIER[0]=1)，则闹钟中断标志AIF(RIIR[0])被置位同时发出闹钟中断请求。

5.12.5.10 *应用指南*

1. TAR, CAR, TLR 和 CLR 寄存器为BCD 格式。
2. 用户必须保证载入值为有效合理的。例如，CLR = 201a (年), 13 (月), 00 (日)，或 CLR 与 DWR 不匹配，等等。
3. 上电或复位后寄存器的值

寄存器	复位值
AER	0

CLR	05/1/1 (年/月/日)
TLR	00:00:00 (时: 分 : 秒)
CAR	00/00/00(年/月/日)
TAR	00:00:00(时: 分 : 秒)
TSSR	1 (24 小时制)
DWR	6 (星期六)
RIER	0
RIIR	0
LIR	0
TTR	0

4. 在 TLR 和 TAR 中，仅 2 位 BCD 码用于指示“年”。目前仅支持20xy年份，不支持19xy 或21xy年。

5.12.5.11 *特定寄存器*

RTC模块提供了80字节的特定寄存器用于存储用户的重要信息，这些寄存器的时钟来自于RTC时钟，在向20个特定寄存器（SPR0 ~ SPR19)写值前，用户必须使能SPREN（SPRCTL[2] SPR寄存器使能），用户可以通过读SPRRDY（SPRCTL[7]）来检查是否有数据被写到那些寄存器中。当SPRRDY为1时，特定寄存器的值已经被更新并准备好再次被更新；当SPRRDY为0时，特定寄存器的数据正在更新。

如果外部的32khz时钟（LXT）没准备好，用户通过设定RTC_SEL_10K(CLKSEL2[18])为1来选择芯片内部的10kHZ时钟(LIRC)代替LXT，来完成特定寄存器的读写操作

5.12.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
RTC 基地址: **RTC_BA = 0x4000_8000**				
INIR	RTC_BA+0x00	R/W	RTC初始寄存器	0x0000_0000
AER	RTC_BA+0x04	R/W	RTC访问使能寄存器	0x0000_0000
FCR	RTC_BA+0x08	R/W	RTC频率补偿寄存器	0x0000_0700
TLR	RTC_BA+0x0C	R/W	时间载入寄存器	0x0000_0000
CLR	RTC_BA+0x10	R/W	日历载入寄存器	0x0005_0101
TSSR	RTC_BA+0x14	R/W	时间格式（时标）选择寄存器	0x0000_0001
DWR	RTC_BA+0x18	R/W	一周日期寄存器	0x0000_0006
TAR	RTC_BA+0x1C	R/W	时间闹钟寄存器	0x0000_0000
CAR	RTC_BA+0x20	R/W	日历闹钟寄存器	0x0000_0000
LIR	RTC_BA+0x24	R	闰年指示寄存器	0x0000_0000
RIER	RTC_BA+0x28	R/W	RTC中断使能寄存器	0x0000_0000
RIIR	RTC_BA+0x2C	R/W	RTC中断指示寄存器	0x0000_0000
TTR	RTC_BA+0x30	R/W	RTC时钟节拍寄存器	0x0000_0000
SPRCTL	RTC_BA+0x3C	R/W	RTC特定功能控制寄存器	0x0000_0080
SPR0	RTC_BA+0x40	R/W	RTC特定寄存器0	0x0000_0000
SPR1	RTC_BA+0x44	R/W	RTC特定寄存器1	0x0000_0000
SPR2	RTC_BA+0x48	R/W	RTC 特定寄存器2	0x0000_0000
SPR3	RTC_BA+0x4C	R/W	RTC 特定寄存器3	0x0000_0000
SPR4	RTC_BA+0x50	R/W	RTC 特定寄存器4	0x0000_0000
SPR5	RTC_BA+0x54	R/W	RTC 特定寄存器5	0x0000_0000
SPR6	RTC_BA+0x58	R/W	RTC 特定寄存器6	0x0000_0000
SPR7	RTC_BA+0x5C	R/W	RTC 特定寄存器7	0x0000_0000
SPR8	RTC_BA+0x60	R/W	RTC 特定寄存器8	0x0000_0000
SPR9	RTC_BA+0x64	R/W	RTC 特定寄存器 9	0x0000_0000
SPR10	RTC_BA+0x68	R/W	RTC 特定寄存器10	0x0000_0000
SPR11	RTC_BA+0x6C	R/W	RTC 特定寄存器 11	0x0000_0000
SPR12	RTC_BA+0x70	R/W	RTC 特定寄存器12	0x0000_0000

SPR13	RTC_BA+0x74	R/W	RTC 特定寄存器13	0x0000_0000
SPR14	RTC_BA+0x78	R/W	RTC特定寄存器14	0x0000_0000
SPR15	RTC_BA+0x7C	R/W	RTC 特定寄存器15	0x0000_0000
SPR16	RTC_BA+0x80	R/W	RTC 特定寄存器 16	0x0000_0000
SPR17	RTC_BA+0x84	R/W	RTC 特定寄存器17	0x0000_0000
SPR18	RTC_BA+0x88	R/W	RTC特定寄存器18	0x0000_0000
SPR19	RTC_BA+0x8C	R/W	RTC特定寄存器 19	0x0000_0000

5.12.7 寄存器描述

RTC 初始化寄存器(INIR)

寄存器	偏移地址	R/W	描述	复位值
INIR	RTC_BA+0x00	R/W	RTC 初始化寄存器	0x0000_0000

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

位	描述	
[31:1]	**INIR[31:1]**	**RTC 初始化** 当 RTC 模块上电后，RTC 处于复位状态。用户需要写入数据 (0xa5eb1357) 到 INIR 使得 RTC 离开复位状态。一旦 INIR 被写入 0xa5eb1357，RTC 将会一直处于非复位状态。 INIR寄存器的这些位只能被写，如果读的话，总是为0
[0]	**INIR[0]/Active**	**RTC 激活状态 （只读）** 0 = RTC 在复位状态 1 = RTC 在正常激活状态

RTC访问使能寄存器(AER)

寄存器	偏移地址	R/W	描述	复位值
AER	RTC_BA+0x04	R/W	RTC 访问使能寄存器	0x0000_0000

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

位	描述	
[31:17]	**Reserved**	保留
[16]	**ENF**	**RTC 寄存器访问使能标志 (只读)** 0 = 禁止RTC寄存器读/写访问 1 = 使能RTC寄存器读/写访问 **注意**：当AER[15:0] 被写入0xA965后，该位将被置1，在1024个RTC 时钟后将被自动清0
[15:0]	**AER**	**RTC 寄存器访问使能密码（只写)** 写0xA965 到这个寄存器，在1024个RTC 时钟内可访问RTC

RTC频率补偿寄存器(FCR)

寄存器	偏移地址	R/W	描述	复位值
FCR	RTC_BA+0x08	R/W	RTC频率补偿寄存器	0x0000_0700

31	30	29	28	27	26	25	24
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**					

位	描述	
[31:12]	**Reserved**	保留
[11:8]	**INTEGER**	**整数部分** 请参考5.14.5.4 .
[5:0]	**FRACTION**	**小数部分** 公式= (检测到的值的小数部分) x 60. **注意:** FCR 的值必须按照 Hex 格式表示

注意:.在 RTC 访问使能位 ENF (AER[16]) 激活后，该寄存器的值可以被读回。.

RTC时间载入寄存器(TLR)

寄存器	偏移地址	R/W	描述	复位值
TLR	RTC_BA+0x0C	R/W	时间载入寄存器	0x0000_0000

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

位	描述	
[31:22]	**Reserved**	保留
[21:20]	**10HR**	十位小时 时间数字(0~2)
[19:16]	**1HR**	个位小时时间数字（0~9)
[15]	**Reserved**	保留
[14:12]	**10MIN**	十位分钟时间数字(0~5)
[11:8]	**1MIN**	个位分钟时间数字 (0~9)
[7]	**Reserved**	保留
[6:4]	**10SEC**	十位秒时间数字(0~5)
[3:0]	**1SEC**	个位秒时间数字 (0~9)

注意：

1. TLR 为 BCD 计数方式，RTC 不会对载入值的合理性进行检测.
2. 括号内列出的为可接受的值。.
3. 当 RTC运行在 12-小时制范围模式， 10HR 的高位是AM/PM的意思

RTC日历载入寄存器(CLR)

寄存器	偏移地址	R/W	描述	复位值
CLR	RTC_BA+0x10	R/W	日历载入寄存器	0x0005_0101

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

位	描述	
[31:24]	**Reserved**	保留
[23:20]	**10YEAR**	十位 年日历数字 (0~9)
[19:16]	**1YEAR**	个位 年日历数字(0~9)
[15:13]	**Reserved**	保留
[12]	**10MON**	十位 月日历数字 (0~1)
[11:8]	**1MON**	个位 月日历数字(0~9)
[7:6]	**Reserved**	保留
[5:4]	**10DAY**	十位 天日历数字 (0~3)
[3:0]	**1DAY**	个位 天日历数字(0~9)

注意:

1. CLR 为 BCD 计数格式，RTC 不会检测载入值的合理性。
2. 括号内列出的为可接受的值。

RTC时间格式选择寄存器(TSSR)

寄存器	偏移地址	R/W	描述	复位值
TSSR	RTC_BA+0x14	R/W	时间格式选择寄存器	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							**24H_12H**

位	描述	
[31:1]	**Reserved**	保留
[0]	**24H_12H**	**24-小时/ 12-小时 模式选择** 用于表示 TLR 和 TAR 为 24-小时 或 12-小时模式 请参考5.14.5.6 . 0 =选择 12-小时制 1 =选择 24-小时制

RTC一周日期寄存器(DWR)

寄存器	偏移地址	R/W	描述	复位值
DWR	RTC_BA+0x18	R/W	一周日期寄存器	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					DWR		

位	描述	
[31:3]	Reserved	保留
[2:0]	DWR	一周日期寄存器 000 = 星期日 001 = 星期一 010 = 星期二 011 = 星期三 100 = 星期四 101 = 星期五. 110 = 星期六 111 = 保留

RTC时间闹钟寄存器(TAR)

寄存器	偏移地址	R/W	描述	复位值
TAR	RTC_BA+0x1C	R/W	时间闹钟寄存器	0x0000_0000

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

位	描述	
[31:22]	Reserved	保留
[21:20]	10HR	十位 小时时间数字闹钟设定(0~2)
[19:16]	1HR	个位 小时时间数字闹钟设定(0~9)
[15]	Reserved	保留
[14:12]	10MIN	十位 分钟时间数字闹钟设定(0~5)
[11:8]	1MIN	个位 分钟时间数字闹钟设定(0~9)
[7]	Reserved	保留
[6:4]	10SEC	十位 秒时间数字闹钟设定(0~5)
[3:0]	1SEC	个位 秒时间数字闹钟设定(0~9)

注意:

1. 在 RTC 访问使能位 ENF (AER[16]) 激活后，该寄存器的值可以被读回。
2. TAR 为 BCD 计数方式，RTC 不会对载入值的合理性进行检测。.
3. 括号内列出的为可接受的值。.
4. 当 RTC运行在 12-小时制式， 10HR 的高位是AM/PM的意思。

RTC日历闹钟寄存器(CAR)

寄存器	偏移地址	R/W	描述	复位值
CAR	RTC_BA+0x20	R/W	日历闹钟寄存器	0x0000_0000

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

位	描述	
[31:24]	**Reserved**	保留
[23:20]	**10YEAR**	十位 年日历数字闹钟设定 (0~9)
[19:16]	**1YEAR**	个位 年日历数字闹钟设定(0~9)
[15:13]	**Reserved**	保留
[12]	**10MON**	十位 月日历数字闹钟设定 (0~1)
[11:8]	**1MON**	个位 月日历数字闹钟设定(0~9)
[7:6]	**Reserved**	保留
[5:4]	**10DAY**	十位 天日历数字闹钟设定(0~3)
[3:0]	**1DAY**	个位 天日历数字闹钟设定(0~9)

注意:

1. 在 RTC 访问使能位 ENF (AER[16]) 激活后，该寄存器的值可以被读回。
2. CAR 为 BCD 计数方式，RTC 不会对载入值的合理性进行检测。
3. 括号内列出的为可接受的值。.

RTC闰年指示寄存器(LIR)

寄存器	偏移地址	R/W	描述	复位值
LIR	RTC_BA+0x24	R	闰年指示寄存器	0x0000_0000

31	30	29	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							LIR

位	描述	
[31:1]	Reserved	保留
[0]	LIR	闰年指示寄存器（只读） 0 = 表示该年非闰年 1 = 表示该年为闰年

RTC中断使能寄存器(RIER)

寄存器	偏移地址	R/W	描述	复位值
RIER	RTC_BA+0x28	R/W	RTC中断使能寄存器	0x0000_0000

31	30	29	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						**TIER**	**AIER**

位	描述	
[31:2]	**Reserved**	保留
[1]	**TIER**	**时钟节拍中断使能位** 该位用来使能/禁止RTC时间节拍中断，如果TIF(RIIR [1] RTC 时钟节拍中断标志)置1，会产生一个中断信号。. 0 = RTC 时钟节拍中断禁用 1 = RTC 时钟节拍中断使能 **注意：** 当系统在空闲/掉电模式并且RTC时钟节拍中断信号产生，该位也会触发一个唤醒事件。
[0]	**AIER**	**闹钟中断使能位** 该位用来使能/禁止RTC闹钟中断，如果AIF (RIIR [0] RTC 闹钟中断标志)置1，会产生一个中断信号。 0 = RTC 闹钟中断禁用 1 = RTC 闹钟中断使能 **注意：** 当系统在空闲/掉电模式并且RTC闹钟中断信号产生，该位也会触发一个唤醒事件

RTC中断指示寄存器(RIIR)

寄存器	偏移地址	R/W	描述	复位值
RIIR	RTC_BA+0x2C	R/W	RTC中断指示寄存器	0x0000_0000

31	30	29	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						**TIF**	**AIF**

位	描述	
[31:2]	**Reserved**	保留
[1]	**TIF**	**RTC时钟节拍中断标志** 如果RTC 时钟节拍中断使能 TIER (RIER[1])被置1，当 RTC 时钟节拍发生，该位将置1并产生一个中断信号;当芯片在掉电模式下，如果RTC节拍中断使能，该位置1将会唤醒IC 0 = 节拍情况未发生 1 = 节拍情况发生 **注意:** 写1清该位
[0]	**AIF**	**RTC 闹钟中断标志** 当 RTC 闹钟中断使能位AIER (RIER[0])被置1，一旦 RTC TLR 和 CLR 的值达到了闹钟设定寄存器 TAR 和 CAR 的值，RTC 控制器将设置 AIF 为1并产生一个中断信号；当芯片工作在掉电模式下，如果RTC闹钟中断使能，将会唤醒IC. 0 = 闹钟条件未匹配 1 = 闹钟条件匹配 **注意：**写1清该位

RTC时钟节拍寄存器(TTR)

寄存器	偏移地址	R/W	描述	复位值
TTR	RTC_BA+0x30	R/W	RTC时钟节拍寄存器	0x0000_0000

31	30	29	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					TTR[2:0]		

位	描述	
[31:3]	Reserved	保留
[2:0]	TTR	**时钟节拍寄存器** 该位用来设定产生时钟节拍中断的周期. 000 = 时钟节拍 为1秒 001 = 时钟节拍 为1/2 秒 010 = 时钟节拍 为1/4 秒 011 = 时钟节拍 为1/8 秒 100 = 时钟节拍 为1/16 秒 101 = 时钟节拍 为1/32 秒 110 = 时钟节拍 为1/64 秒. 111 = 时钟节拍 为1/28 秒 **注意**：在 RTC 访问使能位 ENF (AER[16]) 激活后，该寄存器的值可以被读回。.

RTC 特定功能控制寄存器 (SPRCTL)

寄存器	偏移地址	R/W	描述	复位值
SPRCTL	RTC_BA+0x3C	R/W	RTC 特定功能控制寄存器	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**
SPRRDY	**Reserved**				**SPREN**	**Reserved**	

位	描述	
[31:8]	**Reserved**	保留
[7]	**SPRRDY**	**SPR 寄存器准备** 该位表明寄存器SPRCTL, SPR0 ~ SPR19 是否准备好被访问 在用户写数据到寄存器SPRCTL, SPR0 ~ SPR19，通过读该位可以检查这些寄存器是否更新完成 0 = SPRCTL, SPR0 ~ SPR19 更新正在进行 1 = SPRCTL, SPR0 ~ SPR19 更新完成并准备接收访问. **注意:** 该位只读，向该位写数据不会有任何影响
[6:3]	**Reserved**	保留
[2]	**SPREN**	**SPR 寄存器使能位** 0 = 特定寄存器禁用 1 = 特定寄存器使能 **注意:** 当特定寄存器禁用后，RTC SPR0 ~ SPR19将不可访问
[1:0]	**Reserved**	保留

RTC 特定寄存器(SPRx)

寄存器	偏移地址	R/W	描述	复位值
SPR0	RTC_BA+0x40	R/W	RTC 特定寄存器 0	0x0000_0000
SPR1	RTC_BA+0x44	R/W	RTC 特定寄存器 1	0x0000_0000
SPR2	RTC_BA+0x48	R/W	RTC 特定寄存器 2	0x0000_0000
SPR3	RTC_BA+0x4C	R/W	RTC 特定寄存器 3	0x0000_0000
SPR4	RTC_BA+0x50	R/W	RTC 特定寄存器 4	0x0000_0000
SPR5	RTC_BA+0x54	R/W	RTC 特定寄存器 5	0x0000_0000
SPR6	RTC_BA+0x58	R/W	RTC 特定寄存器 6	0x0000_0000
SPR7	RTC_BA+0x5C	R/W	RTC 特定寄存器 7	0x0000_0000
SPR8	RTC_BA+0x60	R/W	RTC 特定寄存器 8	0x0000_0000
SPR9	RTC_BA+0x64	R/W	RTC 特定寄存器 9	0x0000_0000
SPR10	RTC_BA+0x68	R/W	RTC 特定寄存器 10	0x0000_0000
SPR11	RTC_BA+0x6C	R/W	RTC 特定寄存器 11	0x0000_0000
SPR12	RTC_BA+0x70	R/W	RTC 特定寄存器 12	0x0000_0000
SPR13	RTC_BA+0x74	R/W	RTC 特定寄存器 13	0x0000_0000
SPR14	RTC_BA+0x78	R/W	RTC 特定寄存器 14	0x0000_0000
SPR15	RTC_BA+0x7C	R/W	RTC 特定寄存器 15	0x0000_0000
SPR16	RTC_BA+0x80	R/W	RTC 特定寄存器 16	0x0000_0000
SPR17	RTC_BA+0x84	R/W	RTC 特定寄存器 17	0x0000_0000
SPR18	RTC_BA+0x88	R/W	RTC 特定寄存器 18	0x0000_0000
SPR19	RTC_BA+0x8C	R/W	RTC 特定寄存器 19	0x0000_0000

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

位	描述	
[31:0]	**SPARE**	**特定寄存器** 这个区域用于存储和备份用户的信息. 一旦snooper引脚事件被侦测到，这部分区域的内容将会被硬件自动清除.在向特定寄存器存储备份信息之前，用户应该写0xA965到AER[15:0]确保寄存器的读/写使能位ENF (AER[16])处于激活状态

5.13 UART 接口控制器(UART)

5.13.1 概述

NuMicro NUC200系列提供了多达3个异步串行接口. UART0为高速串口，UART1~2为普通串口。此外，只有UART0 和 UART1支持硬件流控功能。UART控制器的接收过程是把外设的串行数据转为并行数据，发送过程是把CPU的并行数据转成串行数据发送出去。UART控制器支持IrDA串行功能、LIN主/从功能，和RS-485功能模式。每个UART通道支持七种类型的中断。

5.13.2 特性

- 全双工，异步通讯口
- UART0/UART1/UART2分别有64/16/16个字节的收和发FIFO缓冲区
- 支持硬件自动流控功能(CTS, RTS)，RTS自动流控触发电平可设(UART0 和 UART1 支持该功能)
- 接收FIFO区域触发等级的数据长度可设
- 每个通道波特率可单独设置
- 支持CTS引脚触发唤醒功能(仅UART0 和 UART1 支持此功能)
- 支持 7位接收缓存定时溢出检测功能
- UART0/UART1两通道支持DMA数据收发
- 可通过设置UA_TOR [DLY]寄存器的相应位来设置两个数据间（从上一个stop 位到下一个start位之间）的时间间隔
- 支持break error, frame error, parity error和收发缓冲区溢出检测等功能
- 可编程串行接口特性
  - 数据位长度可设为5~8位
  - 校验位可设为，奇、偶校验、无校验或 固定校验位的产生和检测
  - 可设置停止位长度为，1位,1.5位或2位。
- IrDA SIR 功能模式
  - 支持正常模式下3/16位宽功能
- LIN 功能模式
  - 支持LIN 主/从模式
  - 支持传输中产生break功能可设
  - 支持接收器break检测功能
- RS-485模式
  - 支持RS-485 9位模式
  - 支持 RTS 软硬件控制使能(仅 UART0 和 UART1)

5.13.3 框图.

串口时钟控制和内部模块框图分别如图 6-128 和 图 6-129所示

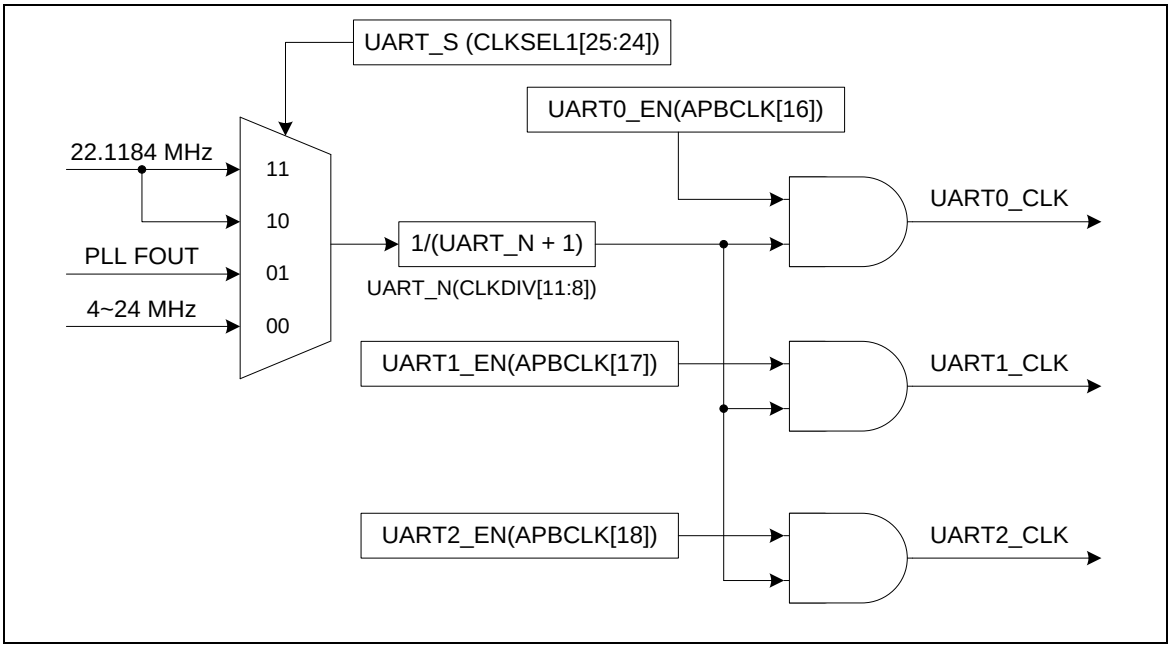
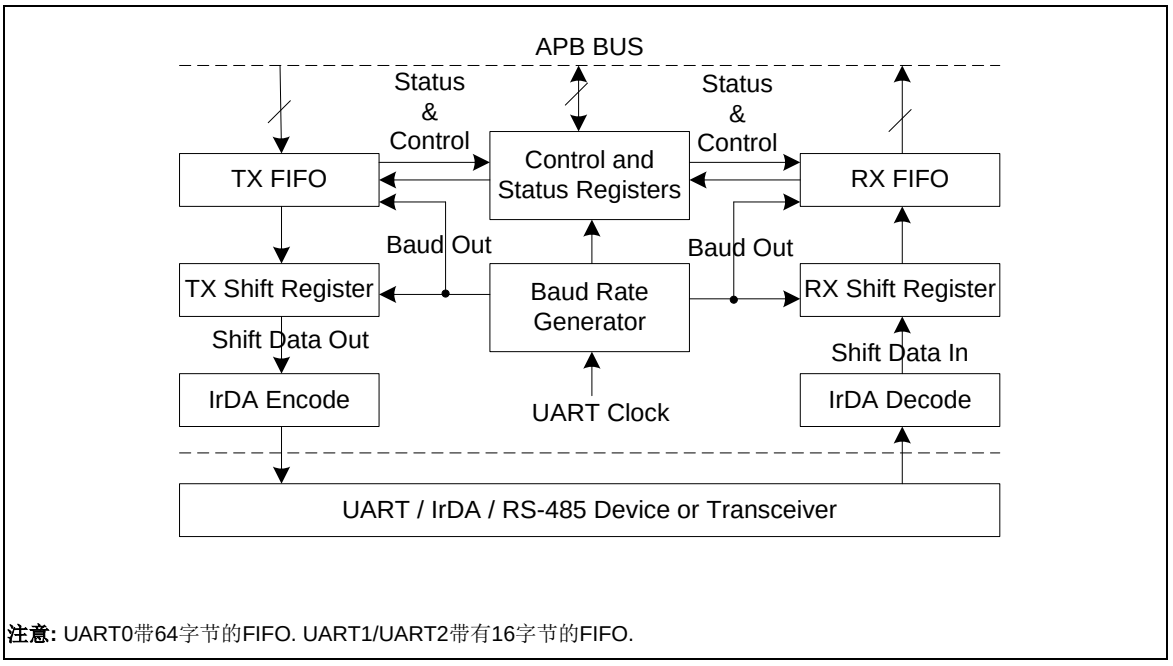


图 5-57 串口时钟控制框图



注意: UART0带64字节的FIFO. UART1/UART2带有16字节的FIFO.

图 5-58 串口模块框图

每个模块功能详细描述如下:

TX_FIFO

发送口带有一个64/16 字节的 FIFO缓冲区以减少CPU中断的频率

RX_FIFO

接收口带一个64/16 字节的 FIFO缓冲区（每个字节加三个错误位）以减少CPU的中断频率

发送移位寄存器

该模块用于控制把并行数据串行输出

接收移位寄存器

该模块用于控制把串行数据并行输入

Modem控制寄存器.

该寄存器用于控制到Modem或数传机（或类似于Modem的外设）的接口

波特率发生器.

通过把输入的外部时钟除频得到期望的波特率。详情请参考波特率公式

IrDA编码.

该模块是IrDA编码控制模块

IrDA解码.

该模块是IrDA解码控制模块

控制和状态寄存器.

该区域是包含FIFO控制寄存器(UA_FCR)，FIFO状态寄存器(UA_FSR),和收发线控寄存器(UA_LCR)的寄存器集合。时间溢出控制寄存器(UA_TOR)表明时间溢出中断的条件。该寄存器集也包括中断使能寄存器(UA_IER)和中断状态寄存器(UA_ISR)，以使能或禁止相应中断,中断源查询。一共有7种类型的中断，包括发送FIFO空中断(THRE_INT),接收达到阈值(RDA_INT)中断，线状态中断(parity error 或 framing error 或 break interrupt) (RLS_INT)，超时中断(TOUT_INT)，MODEM/唤醒状态中断(MODEM_INT)，缓存错误中断 (BUF_ERR_ INT) 和 LIN接收器break域检测中断(LIN_INT) 。

5.13.4 基本配置

UART控制器功能脚在PB_MFP, PD_MFP, ALT_MFP, ALT_MFP1 和 ALT_MFP2寄存器中被配置

UART 控制器时钟， UART0 和 UART1 分别在 UART0_EN(APBCLK[16]) 和 UART1_EN (APBCLK[17])中被使能

UART控制器时钟源通过UART_S(CLKSEL[25:24])位来选择.

UART控制器时钟预分频通过UART_N(CLKDIV[11:8])位来设置

UART接口控制器引脚描述如下：

引脚	输入输出类型	描述
UART_TXD	输出	UART 发送
UART_RXD	输入	UART 接收
UART_nCTS	输入	UART modem 清零发送
UART_nRTS	输出	UART modem 请求发送

表 5-11 UART 接口控制脚

5.13.5 功能描述

UART控制器支持四个功能模式，包括UART，IrDA，LIN和RS-485模式。用户可以通过对UA_FUN_SEL设置选择功能。四种功能模式将在下面的章节中描述

5.13.5.1 *UART控制器波特率发生器.*

UART控制器包含一个可编程波特率发生器，其通过分频器对输入时钟源分频而得到收发数据所需的串行时钟。波特率计算公式：波特率= UART_CLK / M * [BRD + 2]，这里M和BRD在波特率分频器寄存器(UA_BAUD).中有所定义。下表列出了各种条件下的UART波特率计算公式和参数的设置。通过设置相应的参数和寄存器可以得到零误差的波特率。IrDA功能模式，波特率发生器必须是模式0.

模式	DIV_X_EN	DIV_X_ONE	除数 X	BRD	波特率公式
0	0	0	无关	A	UART_CLK / [16 * (A+2)].
1	1	0	B	A	UART_CLK / [(B+1) * (A+2)] , B must >= 8.
2	1	1	无关	A	UART_CLK / (A+2), 如果UART 外围时钟<= HCLK, A 必须 >=9. 如果HCLK <UART 外围时钟<= 2*HCLK, A 必须 >=15. 如果 2*HCLK <UART 外围时钟<= 3*HCLK, A 必须 >=21. 如果 UART 外围时钟> 3*HCLK, 不支持. UART 外围时钟= UART 时钟源/(UART 时钟分频数+1).

表 5-12 UART 波特率公式

UART外围时钟 = 22.1184 MHz			
波特率	模式 0	模式1	模式2
921600	不支持	A=0, B=11	A=22
460800	A=1	A=1, B=15 A=2, B=11	A=46
230400	A=4	A=4, B=15 A=6, B=11	A=94
115200	A=10	A=10, B=15 A=14, B=11	A=190
57600	A=22	A=22, B=15 A=30, B=11	A=382
38400	A=34	A=62, B=8 A=46, B=11 A=34, B=15	A=574
19200	A=70	A=126, B=8 A=94, B=11 A=70, B=15	A=1150
9600	A=142	A=254, B=8 A=190, B=11 A=142, B=15	A=2302
4800	A=286	A=510, B=8 A=382, B=11 A=286, B=15	A=4606

表 5-13 UART 控制器波特率参数设置表

UART 外围时钟 = 22.1184 MHz

波特率	模式0	模式1	模式2
921600	Not support	0x2B00_0000	0x3000_0016
460800	0x0000_0001	0x2F00_0001 0x2B00_0002	0x3000_002E
230400	0x0000_0004	0x2F00_0004 0x2B00_0006	0x3000_005E
115200	0x0000_000A	0x2F00_000A 0x2B00_000E	0x3000_00BE
57600	0x0000_0016	0x2F00_0016 0x2B00_001E	0x3000_017E
38400	0x0000_0022	0x2800_003E 0x2B00_002E 0x2F00_0022	0x3000_023E
19200	0x0000_0046	0x2800_007E 0x2B00_005E 0x2F00_0046	0x3000_047E
9600	0x0000_008E	0x2800_00FE 0x2B00_00BE 0x2F00_008E	0x3000_08FE
4800	0x0000_011E	0x2800_01FE 0x2B00_017E 0x2F00_011E	0x3000_11FE

表 5-14 UART 控制器波特率寄存器(UA_BAUD)设置表

5.13.5.2 *UART控制器发送延时时间*

UART控制器可以通过设置DLY (UA_TOR [15:8])位来控制传输过程两个数据帧之间即上一个数据帧的停止位和下一个数据帧的起始位之间的间隔。单位是位。延时操作如下图所示

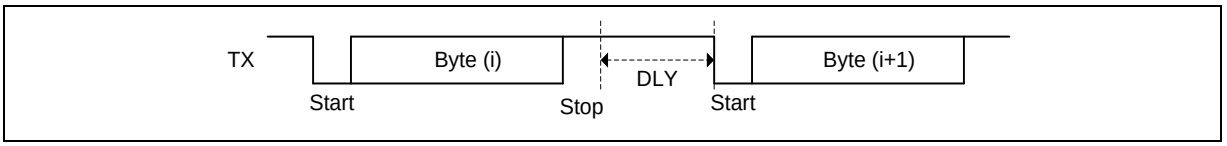


图 5-59 发送延时操作

5.13.5.3 *UART控制器FIFO控制和状态*

UART0内置一个64字节的发送FIFO (TX_FIFO)和一个64字节的接收FIFO (RX_FIFO), 通讯中,使用这些收发FIFO缓冲寄存器可以减少对CPU的中断次数。UART1~2仅有16字节的发送FIFO (TX_FIFO)和16字节的接收FIFO (RX_FIFO)。CPU在任何时候都可以读到UART的状态。状态信息包括UART传输操作的类型和条件，以及接收过程有可能发生的3个错误状态（奇偶校验错误，帧错误，打断错误）。FIFO控制和状态也支持所有的功能模式，包括UART, IrDA, LIN和RS-485模式。

5.13.5.4 *UART控制器唤醒功能*

当芯片在Power-down模式下，外部CTS脚上电平变化可以唤醒芯片。该功能在所有功能模式下都有效，但仅支持UART0 和 UART1。用户要使用唤醒功能还必须使能MODEN_INT中断。

5.13.5.5 *UART控制器中断和状态*

每个UART控制器支持7种类型的中断，它包括：

- 接收阈值水平达到后的中断(RDA_INT)
- 发送FIFO空中断(THRE_INT)
- Line状态中断（奇偶校验错误，帧错误，打断中断）(RLS_INT)
- MODEM/唤醒状态中断(MODEM_INT)
- 接收缓冲区定时溢出中断(TOUT_INT)
- 缓冲区错误中断(BUF_ERR_INT)
- LIN总线中断(LIN_ INT)

下表描述了中断源和中断标志。当中断使能且有中断标志时就会产生中断。用户必须在中断后清中断标志

中断源	中断指示	中断使能位	中断标志	清中断标志方法
接收数据有效中断	RDA_INT	RDA_IEN	**RDA_IF**	读 UA_RBR
发送保持寄存器空中断	THRE_INT	THRE_IEN	**THRE_IF**	写 UA_THR
接收Line状态中断	RLS_INT	RLS_IEN	**RLS_IF** = BIF	**写 “1” 到**BIF
			RLS_IF = FEF	**写 “1” 到**FEF
			RLS_IF = PEF	**写 “1”到**PEF
			RLS_IF = RS485_ADD_DETF	**写 ‘1’ 到** RS485_ADD_DETF
Modem状态中断	MODEM_INT	MODEM_IEN	**MODEM_IF** = DCTSF	**写 “1” 到** DCTSF
RX定时溢出中断	TOUT_INT	RTO_IEN	**TOUT_IF**	读 UA_RBR
缓冲区错误中断	BUF_ERR_INT	BUF_ERR_IEN	**BUF_ERR_IF** =TX_OVER_IF	**写 “1” 到** TX_OVER_IF
			BUF_ERR_IF = RX_OVER_IF	**写 “1” 到** RX_OVER_IF
LIN 总线中断	LIN_ _INT	LIN _IEN	**LIN_IF =** LIN_BKDET_F	**写 “1” 到** LIN_IF **及** **写 “1” 到** LIN_BKDET_F
			LIN_IF = BIT_ERR_F	**写 “1” 到** BIT_ERR_F
			LIN_IF = LIN_IDPERR_F	**写 “1” 到 o** LIN_IDPERR_F
			LIN_IF = LINS_HERR_F	**写 “1” 到** LINS_HERR_F
			LIN_IF = LINS_HDET_F	**写 “1” 到** LINS_HDET_F

表 5-15 UART 控制器中断源和标志列表

UART 中断源	中断使能位	中断指示	中断标志	清标志方式
接收数据有效中断	RDA_INT	RDA_IEN	**HW_RDA_IF**	读 UA_RBR
发送保持寄存器空中断	THRE_INT	THRE_IEN	**HW_THRE_IF**	写 UA_THR
接收 Line状态中断	RLS_INT	RLS_IEN	**HW_RLS_IF** = BIF	**写 '1' 到** RFR
			HW_RLS_IF = FEF	
			HW_RLS_IF = PEF	
			HW_RLS_IF=RS485_ADD_DET	
Modem 状态中断	MODEM_INT	MODEM_IEN	**MODEM_IF** = DCTSF	**写 "1" 到** DCTSF
RX 定时溢出中断	TOUT_INT	RTO_IEN	**HW_TOUT_IF**	读 UA_RBR
缓冲区错误中断	BUF_ERR_INT	BUF_ERR_IEN	**HW_BUF_ERR_IF** = TX_OVER_IF	**写 "1" 到** RFR
			HW_BUF_ERR_IF = RX_OVER_IF	

表 5-16 DMA 模式下的控制器中断源和标志列表

5.13.5.6 *UART功能模式*

UART控制器提供了UART功能(用户须设置UA_FUN_SEL [1:0] 为"00"设置为UART功能)UART 波特率最高速度是1 Mbps.

UART为全双工异步通讯接口。收发各包含一个16字节的FIFO缓冲区。用户可以设置接收时的FIFO触发阈值以及定时溢出检测时间。发送数据帧间（即从上一帧停止位到下一帧起始位）时间间隔通过DLY (UA_TOR [15:8])位可设。UART支持硬件自动流控功能(CTS, RTS),且RTS流控触发电平可设，全双工串行接口通讯参数可设。

UART 线控制功能

UART控制器通过设置UA_LCR寄存器支持串行接口全部特性。可以通过对UA_LCR寄存器设置数据位和停止位长度以及奇偶校验设置。下表列出了UART数据位和停止位长度的设置以及UART奇偶校验位的设置。

NSB (UA_LCR[2])	WLS (UA_LCR[1:0])	数据位长度 (Bit)	停止位长度 (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

表 5-17 UART 线的数据位和停止位长度设置

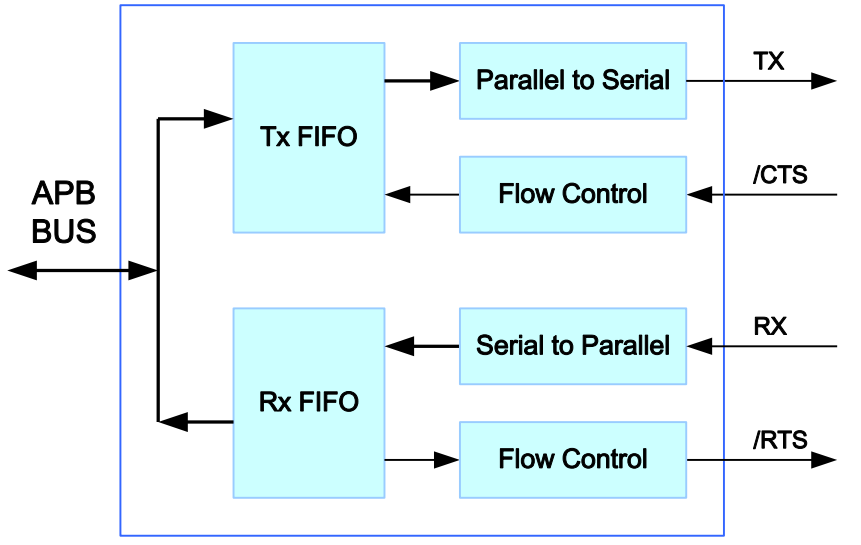
奇偶校验类型	SPE (UA_LCR[5])	EPE (UA_LCR[4])	PBE (UA_LCR[3])	描述
无校验	x	x	0	无奇偶校验位输出
奇校验	0	0	1	奇偶校验位的计算方法是把数据流中的所有的1相加，使得包括校验位中的1的总数为奇数个.
偶校验	0	1	1	奇偶校验位的计算方法是把数据流中的所有的1相加，使得包括校验位中的1的总数为偶数个.
奇偶校验位强制置1	1	0	1	奇偶校验位总是逻辑1 不管数据位中1的个数是多少（奇偶计数），奇偶校验位永远都是逻辑1
奇偶校验位强制为0	1	1	1	奇偶校验位总是逻辑0 不管数据位中1的个数是多少（奇偶计数），奇偶校验位永远都是逻辑0

表 5-18 UART 线奇偶校验设置

UART自动流控功能

UART支持自动流控功能，该功能用到两根信号线CTS (清零发送) 和RTS (请求发送)来控制UART与外部设备（如Modem）间的数据传输。当自动流控使能后，只有等到UART对外部设备发出有效的RTS信号后才允许接收数据，否则不接收。当RX FIFO接收到数据的数量达等于RTS_TRI_LEV (UA_FCR [19:16])位的值后,RTS信号会被取消。当UART检测到外部设备给CTS信号脚有效信号后，UART开始发送数据。否则UART不会发送数据。

以下为自动流控功能模块框图



注意: 该功能仅支持UART0和UART1

图 5-60 自动流控模块框图

以下框图展示了UART CTS自动流控功能模型。用户必须要先设置AUTO_CTS_EN (UA_IER [13])以使能CTS自动流控功能。LEV_CTS (UA_MCR [8])位可以设置CTS脚输入的有效状态。当CTS脚上任何电平变化将导致DCTSF (UA_MSR [0])位被置1，然后TX脚将从TX FIFO自动发出数据。

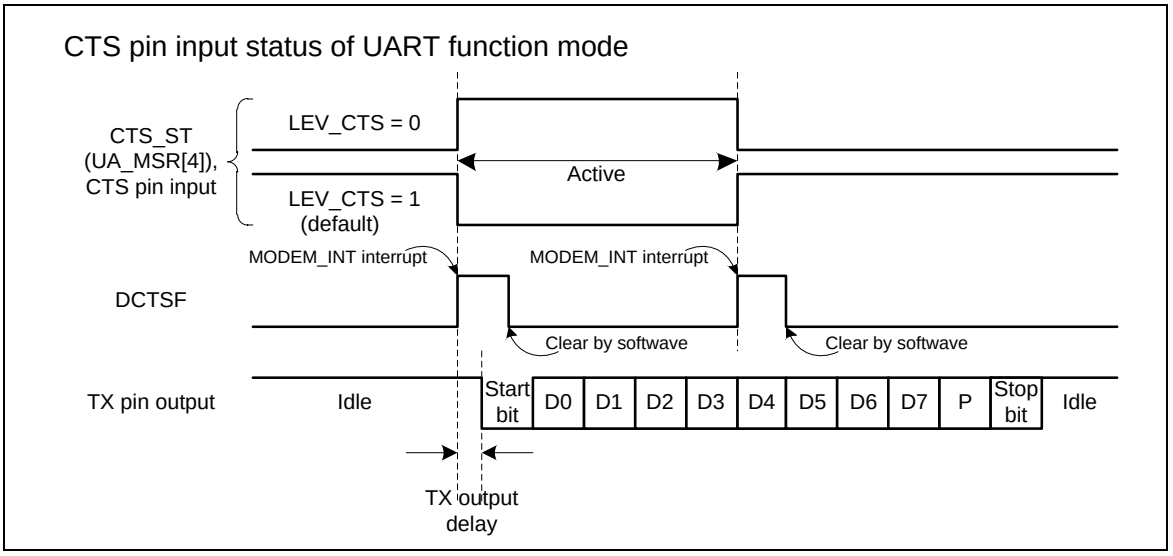


图 5-61 UART CTS 自动流控使能

如下图所示，UART RTS自动流控模式(AUTO_RTS_EN (UA_IER[12])=1)中，RTS的触发阈值由UART FIFO控制寄存器的 RTS_RTI_LEV(UA_FCR[19:16])位来控制。

设置LEV_RTS(UA_MCR[9])位可以控制RTS 脚的反向或非反向。用户可以读RTS_ST(UA_MCR[13])位来知道真实RTS脚输出电压的逻辑状态。

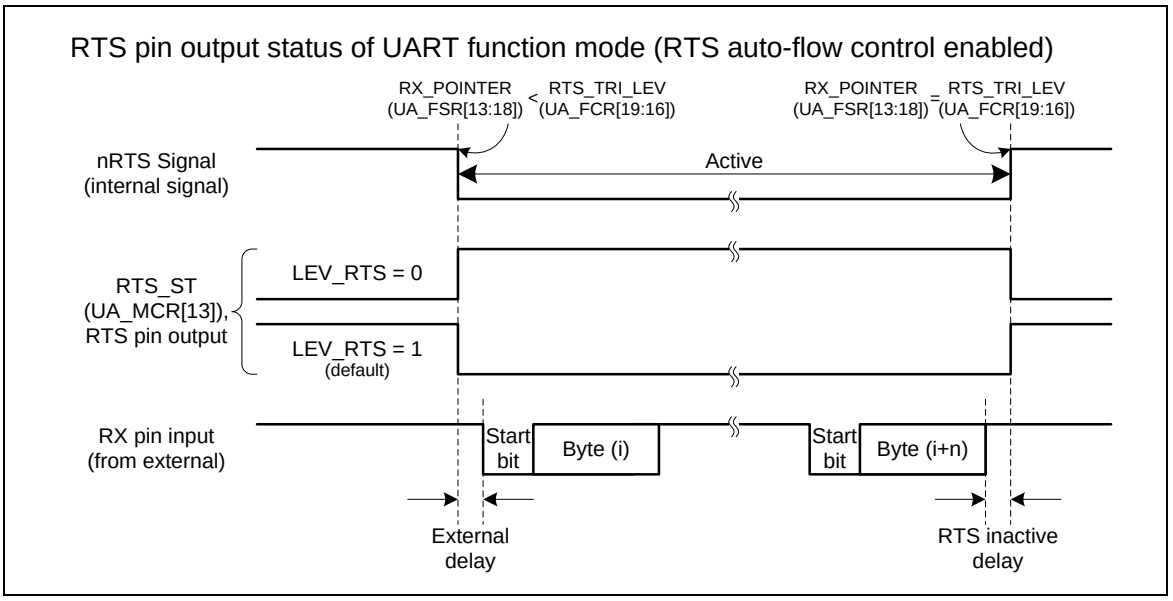


图 5-62 UART RTS 自动流控使能

如下图所示，在软件模式下(AUTO_RTS_EN(UA_IER[12])=0)，RTS流控直接由RTS(UA_MCR[1])位软件来控制。

设置LEV_RTS(UA_MCR[9])位可以控制RTS输出脚状态与RTS(UA_MCR[1])控制状态是同向还是反向。用户可以读RTS_ST(UA_MCR[13])位来获取RTS脚真实输出电平状态。

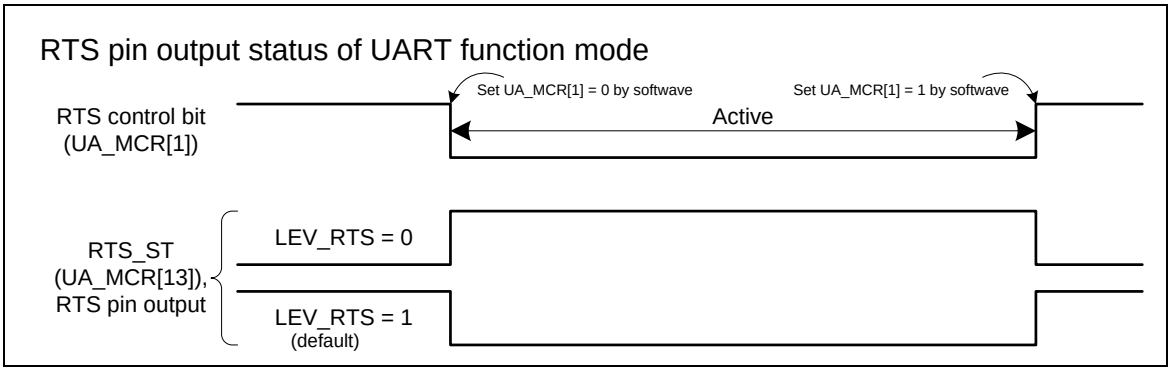


图 5-63 UART RTS 软件控制的流控

5.13.5.7 *IrDA 功能模式*

UART控制器也提供Serial IrDA (SIR, 串行红外)功能(用户必须设置UA_FUN_SEL [1:0] 为'10')来使能IrDA功能。SIR规范定义了一个短距离红外异步串行传输模式，包括一个起始位，8个数据位，一个停止位。最大速率115.2kbps. IrDA SIR模块包含一个IrDA SIR协议编/解码器。IrDA SIR协议是半双工工作模式。所以它不能同时收发数据。IrDA SIR物理层规定了发送与接收数据的时间上至少10ms的时间间隔，该延迟特性需通过软件来完成。

IrDA 模式下，DIV_X_EN (UA_BAUD [29])位需被禁止。

波特率= Clock / (16 * BRD)，这里BRD是UA_BAUD寄存器中定义的波特率分频器。

以下框图展示了IrDA控制模块框图

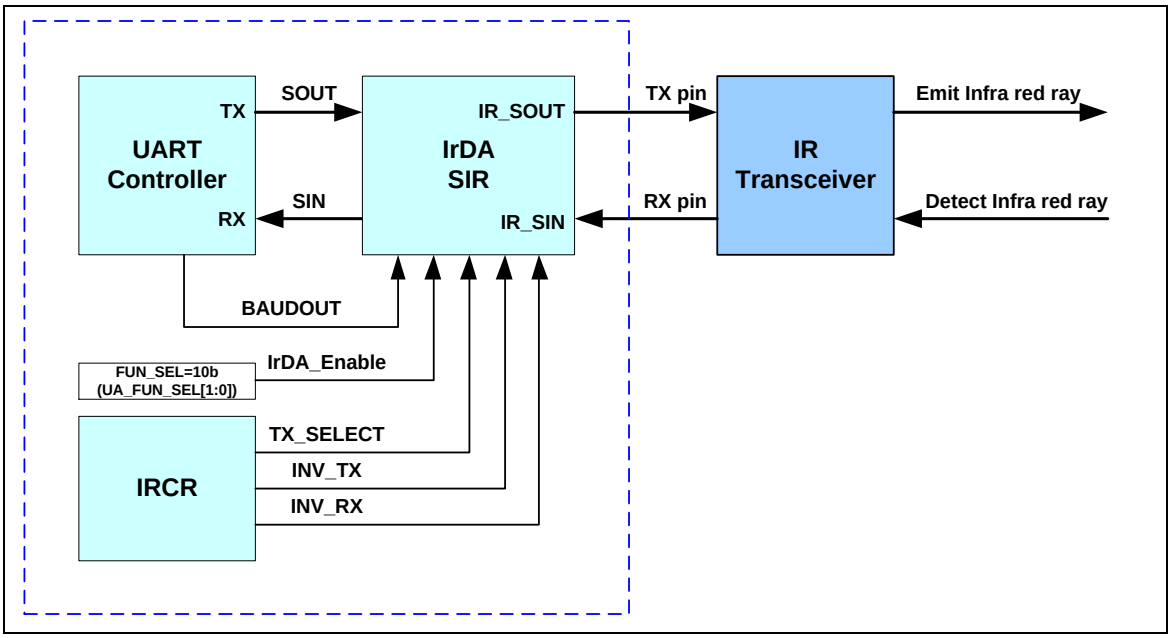


图 5-64 IrDA 控制模块框图

IrDA SIR 发送编码

IrDA SIR 传送编码调制采用 Non-Return-to Zero (NRZ)编码将数据流从 UART 输出。IrDA SIR 物理层指定使用归零反向调制编码 (Return-to-Zero, Inverted (RZI))，用一个红外光脉冲代表逻辑 0，被调整的脉冲输出到外部输出驱动器和红外线发光二极管。

在正常模式下，传输脉冲的宽度为 3/16 波特率周期。.

IrDA SIR 接收解码

IrDA SIR 接收解码器对输入管脚的(Return-to-Zero, Inverted (RZI))串行位流进行解调，并输出NRZ串行位流到 UART接收数据输入端。在空闲状态里，解码器输入端通常为高。（因此，IRCR (INV_RX [6])位默认设为1）。

当解码器输入端为低时，表明接收到一个起始位。

IrDA SIR 操作

IrDA SIR 编码/解码提供 UART 数据流和半双工串行 SIR 之间的转换。IrDA编码/解码波形图如下：

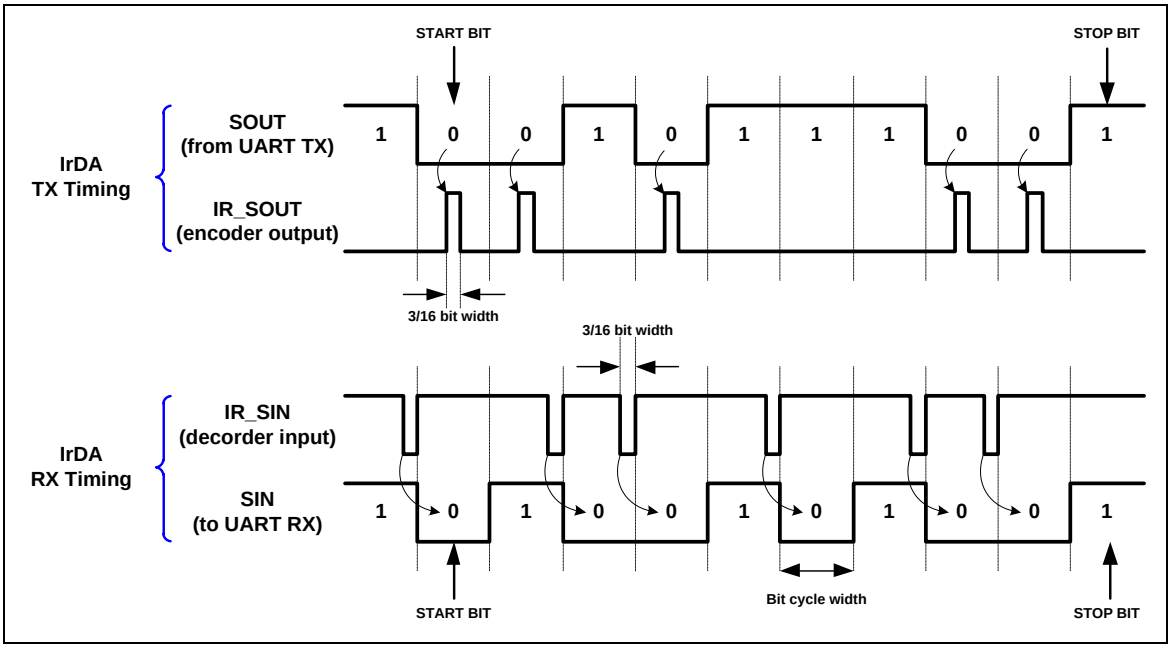


图 5-65 IrDA TX/RX 时序图

5.13.5.8 *LIN (本地互连网络) 模式*

UART0~UART2支持LIN功能，通过设置FUN_SEL (UA_FUN_SEL[1:0])为 ’01’可以将UART设定为LIN模式。在主机模式，UART0~UART2支持LIN break和分隔符产生以及break和分隔符侦测，在LIN从机模式下，支持报头侦测和自动重新同步。

LIN 帧结构

根据LIN协议，所有的传输信息被打包为帧。一个帧由一个报头（主机任务提供）和一个紧跟其后的应答（从机任务提供）组成。报头（主机任务提供）由一个break域和一个同步域再跟一个帧识别码(frame ID)组成。帧ID仅作为定义帧的用途。从机任务负责回应相关的帧ID。响应由一个数据域和一个校验域组成。下图是LIN帧的结构。

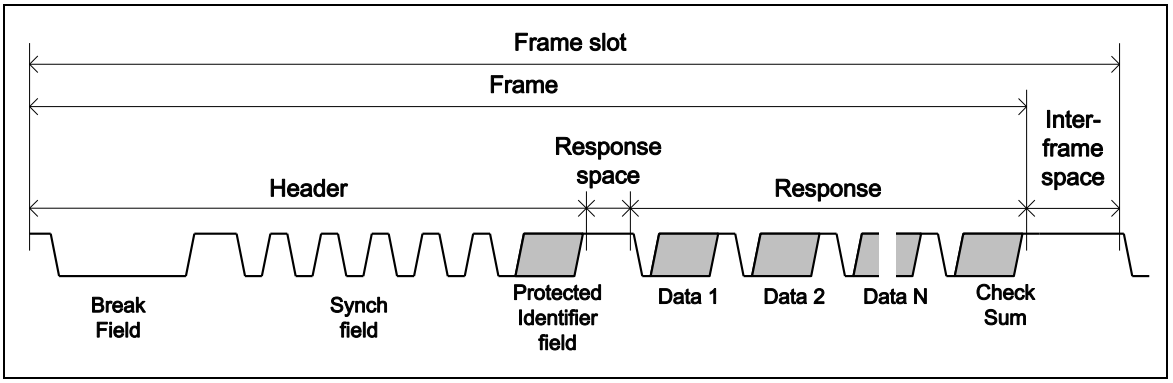


图 5-66 LIN 的帧结构

LIN 字节结构

在LIN模式，根据LIN的标准，每个字节由值为0（显性）的START位开始，接着是8位数据位，没有

奇偶校验位，LSB优先，由一个值为1（隐性）的STOP位结束。字节的结构如下：

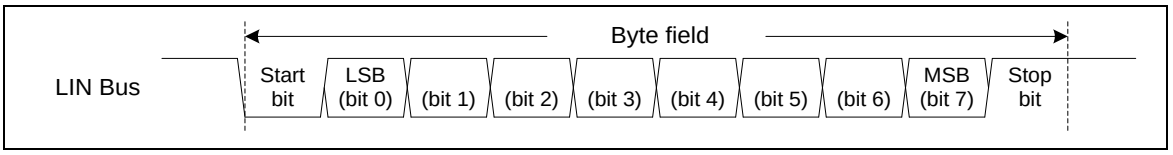


图 5-67 LIN 字节结构

LIN 主机模式

UART0~UART2控制器支持LIN主机模式，使能并初始化LIN主机模式需要如下步骤：

1. 设置UA_BAUD 寄存器设定波特率.

2. 设置WLS (UA_LCR[1:0])位为‘11’配置数据长度为8位，清除PBE (UA_LCR[3])位禁止奇偶校验，通过清NSB (UA_LCR[2])位配置1位stop位。

3. 设置FUN_SEL (UA_FUN_SEL[1:0]) 位为“01”选择LIN功能模式

一个完整的报头由一个break域和同步域再跟一个帧标识符(帧ID)组成。UART0/UART1控制器可以选择三种报头发送模式，通过设置LIN_HEAD_SEL (UA_LIN_CTL[23:22])位可以选择“break 域” 或“break 域和同步域” 或 “break 域, 同步域和 帧 ID 域”选择发送的报头。如果选择的报头是“break域”，软件必须依次填同步数据(0x55)和帧ID数据到UA_THR 寄存器，按顺序来发送一个完整的报头到总线。如果选择的报头是“break 域和同步域”，软件必须填帧ID数据到UA_THR 寄存器，按顺列来发送一个完整的报头到总线。如果选择的报头是“break 域, 同步域和 帧 ID 域”，硬件会自动控制报头发送顺序，但是软件必须填帧ID数据到LIN_PID (UA_LIN_CTL [31:24])位。当选择的报头模式是“break 域, 同步域和 帧 ID 域”时，帧ID校验位可以由软件或硬件来计算，这取决于LIN_IDPEN(UA_LIN_CTL[9])位是否置位。

LIN_HEAD_SEL	Break域	同步域	ID域
0	由H/W产生	由S/W产生	由SW处理
1	由H/W产生	由H/W产生	由SW处理
2	由H/W产生	由H/W产生	由H/W产生(但是软件需要先填ID 到LIN_PID (UA_LIN_CTL[31:24])

表 5-19 LIN 在主模式下的报头选择

当UART 工作于LIN数据传输模式时，LIN总线传输状态可以由硬件或软件监控。通过设置BIT_ERR_EN (UA_LIN_CTL [12])位为 “1”使能硬件监控。如果在LIN发送状态输入管脚(UART_RX)状态不同于输出管脚(UART_TX) 状态，硬件会产生一个中断到CPU。软件也能通过读回UA_RBR寄存器数据监视LIN总线传输状态。下面是一个编程次序示例。

在主机模式下，没有软件错误监控的步骤:

1. 填受保护的ID到LIN_PID (UA_LIN_CTL[31:24])
2. 通过设置LIN_HEAD_SEL (UA_LIN_CTL [23:22])位=10，选择硬件传输的报头域，包括“break域 + 同步域 + 受保护ID域”
3. 通过设置LIN_SHD (UA_LIN_CTL[8])=1选择硬件传输报头域.
4. 等待LIN_SHD (UA_LIN_CTL[8])被硬件清零

5. 等待TE_FLAG (UA_FSR[28])被硬件置位

注意1: break域 的缺省值是12 个显性位 (break 域) 和1 个隐性位 (break/sync 分隔符). 软件可以通过设定LIN_BKFL (UA_LIN_CTL [19:16])和LIN_BS_LEN (UA_LIN_CTL[21:20])来改变break域的长度和break/同步分隔符的长度.

注意2: break/同步分隔符缺省长度是1比特时间，字节间隔缺省也是1个比特时间。软件可以通过设定LIN_BS_LEN (UA_LIN_CTL[21:20])和DLY(UA_TOR[7:0])来改变它们的间隔.

注意3: 如果报头包含“break 域，sync 域和帧ID 域”，在触发硬件发送报头之前(设定LIN_SHD (UA_LIN_CTL[8])，软件必须填帧 ID 到LIN_PID (UA_LIN_CTL[31:24])。根据LIN_IDPEN (UA_LIN_CTL[9])的设定，帧 ID 校验可以由硬件或者软件产生。如果校验由软件产生(LIN_IDPEN (UA_LIN_CTL[9])为0，软件必须填8 比特数据 (包括 2 比特校验)到帧ID域；如果校验比特由硬件产生(LIN_IDPEN (UA_LIN_CTL[9])= 1)，软件填 ID0~ID5就好，硬件负责计算P0和 P1

在主机模式下，有软件错误监控的步骤：

1. 设置LIN_HEAD_SEL (UA_LIN_CTL [23:22])= 00，选择硬件传输的报头域，其内容只包括“break 域”
2. 设置LIN_BKDET_EN (UA_LIN_CTL[10])位使能 break 域侦测功能
3. 设置LIN_SHD (UA_LIN_CTL[8])位请求硬件传输 break + break/同步 分隔符
4. 等待，直到LIN_BKDET_F (UA_LIN_SR[8]) 标志被硬件置为 “1”.
5. 写0x55到UA_THR 寄存器发送 sync 域
6. 等待，直到RDA_IF (UA_ISR[0])标志被硬件置为 “1”，然后读回UA_RBR 寄存器的值
7. 写protected identifier 到UA_THR 寄存器，发送帧ID 域
8. 等待，直到RDA_IF (UA_ISR[0])标志被硬件置为 “1”， 然后读回UA_RBR 寄存器的值

LIN break和分隔符侦测

当软件通过设定LIN_BKDET_EN (UA_LIN_CTL[10])使能break 域侦测功能时， break域侦测功能被激活. break 域侦测电路和UART0/UART1接收电路是完全独立的

当break 域侦测功能使能时,电路侦看UART_RX引脚的起始信号。如果UART LIN控制器侦测到显性连续大于11个比特并跟随1个隐性比特 (分隔符), break域结束时LIN_BKDET_F (UA_LIN_SR[8])标志将被置位，如果LIN_IEN (UA_IER[8]) =1, LIN_INT (UA_ISR[15])中断将发生，break 侦测和break标志如下图所示

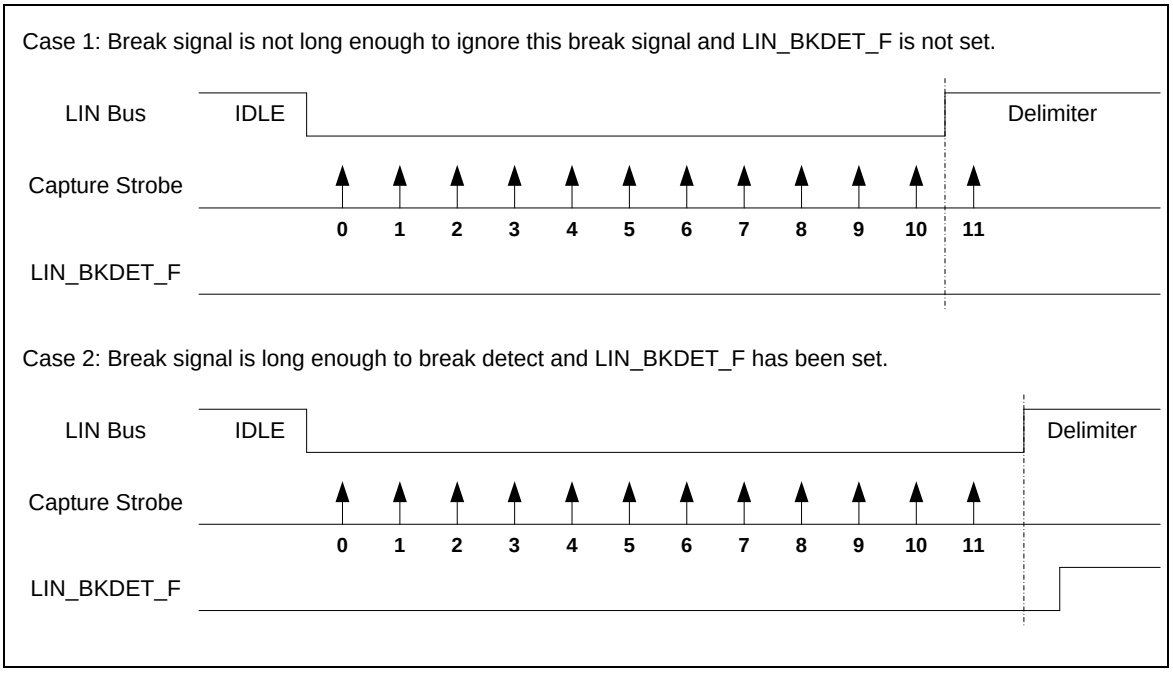


图 5-68 LIN 模式下 Break 检测

LIN break和分隔符的检测

LIN 主机可以发送响应(主机是响应的发起者) 也可以接收响应(主机是响应的接收者). 当主机是响应的发起者时, 主机通过UA_THR 寄存器发送响应；主机是响应的接收者时，主机将从其它从机接收响应.

LIN的帧ID奇偶校验格式

LIN模式下LIN的帧ID值如下，帧ID奇偶校验可以通过软件或硬件产生，产生方式取决于IDPEN (UART_LINCTL[9])位的设置

如果奇偶校验时通过硬件产生，用户需要填写ID0~ID5 (UART_LINCTL [29:24])，硬件将会计算P0 (UART_LINCTL[30])和P1 (UART_LINCTL[31])，否则用户必须填写帧ID和奇偶校验位

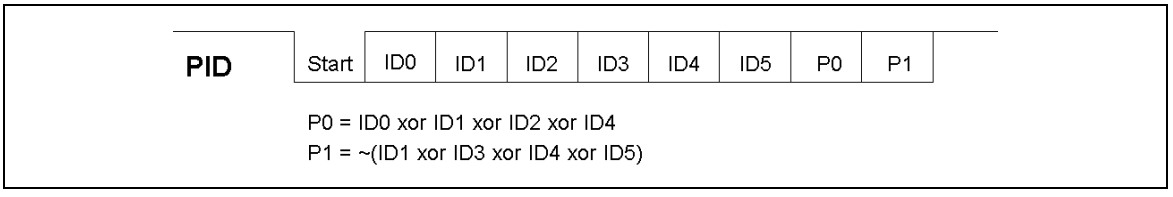


图 5-69 LIN 帧 ID 和奇偶校验格式

LIN 从模式

UART0/UART1控制器支持LIN从模式。使能和初始化LIN从模式的必要步骤如下：

1. 设置UA_BAUD 寄存器设定波特率
2. 设定WLS (UA_LCR[1:0])= 11设定数据长度为8 比特，清PBE (UA_LCR[3])位禁止奇偶校验，清NSB (UA_LCR[2])位设定停止位为1比特
3. 设定FUN_SEL (UA_FUN_SEL[1:0])位为“01”，选择 LIN 功能
4. 设定LINS_EN (UA_LIN_CTL[0])位为1，使能LIN 从机模式

LIN报头接收

根据LIN 协议, 从节点必须等待从主节点接收一个有效的报头，然后应用程序可以采取以下动作 (根据主报头 帧ID 的值)

- 接收响应.
- 发送响应.
- 忽略相应，等待下一个报头.

LIN 从模式下，通过设定LINS_HDET_EN (UA_LIN_CTL[10])位使能从机报头检测功能来侦测完整的报头(接收 “break 域”, “sync 域” 和 “frame ID 域”)。接收到一个 LIN 报头后， LINS_HDET_F (UA_LIN_SR[0]) 标志将被置位。如果(UA_IER[8]) =1, 中断将发生。 通过设定LIN_IDPEN (UA_LIN_CTL[9])位使能帧ID校验检查功能。如果收到的帧ID校验位不正确时 (break 和 sync 域正确), LIN_IDPERR_F (UA_LIN_SR[2])标志将被置1。如果LIN_IEN(UA_IER[8]) =1, 中断将发生。LINS_HDET_F (UA_LIN_SR[0])会被置1。通过设定LIN_MUTE_EN (UA_LIN_CTL[4])=1，用户也可以将 LIN 设为mute 模式。该模式只允许检测报头(break + sync + frame ID) ，不允许接收任何其它字符。为了避免比特率容差, 控制器支持自动重同步功能，避免时钟误差错误，通过设定LINS_ARS_EN (UA_LIN_CTL[2])位使能该特性。

LIN 发送响应

LIN 从机模式可以发送响应和接收响应。当从机节点是响应的发送者时，通过将数据填到UA_THR寄存器发送响应；如果从机节点是响应的接收者时，从接节点从LIN总线接收数据(读UA_RSR寄存器).

LIN报头超时错误

LIN 从机控制器包含一个报头超时计数器。如果整个报头没有在57比特的最大时间限制内收到，报头错误标志LINS_HERR_F (UA_LIN_SR [1])将被置位。超时计数器在每个break侦测沿使能，在下列条件将停止：

- LIN 帧 ID 域已经收到了
- 报头错误标志被置起
- 写 1 到LINS_SYNC_F (UA_LIN_SR[3])比特来重新侦测一个新的报头

Mute模式和 LIN 从mute模式退出的条件

Mute模式下, LIN 从机节点直到满足一定的条件时才能接收其它数据。此模式只允许侦测报头，禁止接收任何其它数据. 通过设定LIN_MUTE_EN (UA_LIN_CTL[4])位使能Mute 模式，设置LIN_HEAD_SEL (UA_LIN_CTL[23:22])位来退出Mute模式。.

注意: 校验字节发送之后，推荐设定LIN从机节点到Mute模式

LIN 从机控制器从Mute模式退出描述如下：如果LIN_HEAD_SEL (UA_LIN_CTL[23:22])设为“break 域”， 当 LIN 从机控制器检测到一个有效的LIN break + 分隔符时, 控制器将使能接收器(退出Mute 模式) 随后的数据(sync 数据, 帧ID , 响应数据) 将被收到 RX-FIFO中。

如果LIN_HEAD_SEL (UA_LIN_CTL[23:22])设为 “break 域 和 sync 域”, 当 LIN 从机控制器检测到一个有效的LIN break + 分隔符后面跟一个有效的同步域并且没有帧错误时，控制器将使能接收器(退出Mute 模式) 随后的数据(帧 ID, 响应数据) 将被收到 RX-FIFO中。

如果LIN_HEAD_SEL (UA_LIN_CTL[23:22])设为“break 域, sync 域和 ID 域”, 当 LIN 从机控制器检测到一个有效的LIN break + 分隔符和有效的同步域并且没有帧错误后面跟一个有效的帧ID域并且没有帧错误，并且收到的帧ID和LIN_PID (UA_LIN_CTL[31:24])中的值匹配时，控制器将使能接收器 (退出Mute 模式) 随后的数据(响应数据) 将被收到 RX-FIFO中。

非自动重新同步从机模式

用户可以关闭自动重新同步功能来固定通信波特率。当工作在非自动重新同步模式时，软件需要一些初始化过程，初始化过程如下所示：

1. 设定UA_BAUD 寄存器设定波特率
2. 设定UA_FUN_SEL (UA_FUN_SEL[1:0])为 “01”选择 LIN 功能
3. 设定LINS_ARS_EN (UA_LIN_CTL[2])= 0关闭自动重新同步功能
4. 设定LINS_EN (UA_LIN_CTL[0])为1使能 LIN 从机模式.

自动重新同步从机模式

自动重新同步模式下，每次收到同步域时，控制器将调整比特率发生器。软件需要一些初始化过程，初始化过程如下所示

1. 设定UA_BAUD 寄存器设定波特率
2. 设定UA_FUN_SEL (UA_FUN_SEL[1:0])为“01”选择 LIN 功能模式
3. 设定LINS_ARS_EN (UA_LIN_CTL[2]) = 1使能自动重新同步功能
4. 设定LINS_EN (UA_LIN_CTL[0])为1，使能LIN从机模式

当自动重新同步功能使能后，每个LIN break 域后面，用LIN的工作时钟持续采样5个下降沿的时间，测量的结果储存在内部一个13-bit 寄存器中，在第5个下降沿结束，UA_BAUD 寄存器的值将被自动更新。如果在5个下降沿之前，测量计数器(13-bit) 溢出，报头错误标志LIN_HERR_F (UA_LIN_SR [1])将被置位.

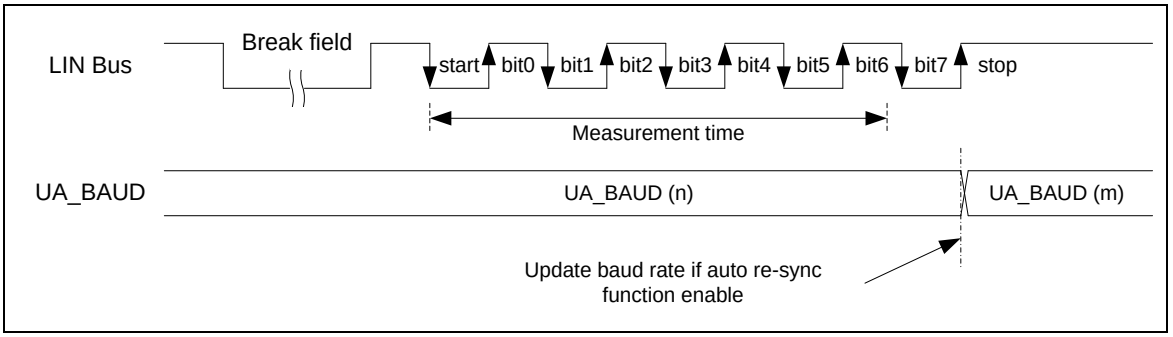


图 5-70 LIN 同步域的测量

自动重新同步模式下，软件必须通过设定UA_BAUD 寄存器设定希望的波特率，硬件将保存到内部的TEMP_REG 寄存器, 每次LIN break 域之后, 5个下降沿被采样，测量结果保存到内部 13-bit 寄存器 (BAUD_LIN) ，这个结果将自动更新到UA_BAUD 寄存器.

为了保证传输波特率,每个新的break域收到之前，波特率发生器必须重新加载初始值。初始值在初始化的时候由应用程序设定（TEMP_REG）。用户可以设定LINS_DUM_EN (UA_LIN_CTL [3]) 位来使能自动加载初始波特率功能。如果LINS_DUM_EN (UA_LIN_CTL [3]) = 1，当前帧结束，收到下一个字符之前，硬件将自动加载初始值到UA_BAUD 寄存器，UA_BAUD 被更新之后, LINS_DUM_EN (UA_LIN_CTL [3])位将被自动清0。LIN波特率的更新方法如下图所示

注意1: 收到检验字符之前，推荐设定LINS_DUM_EN比特为1.

注2: 侦测到报头错误时，用户需要对LINS_SYNC_F (UA_LIN_SR[3]) 位写1来重新检测报头. 当写1到该位时，硬件将重载初始波特率(TEMP_REG) 并重新搜索新的报头.

注 3: 自动重新同步模式下，波特率必须设定为mode2 (DIV_X_EN (UA_BAUD [29]) 和 DIV_X_ONE (UA_BAUD[28]) 必须都为1)

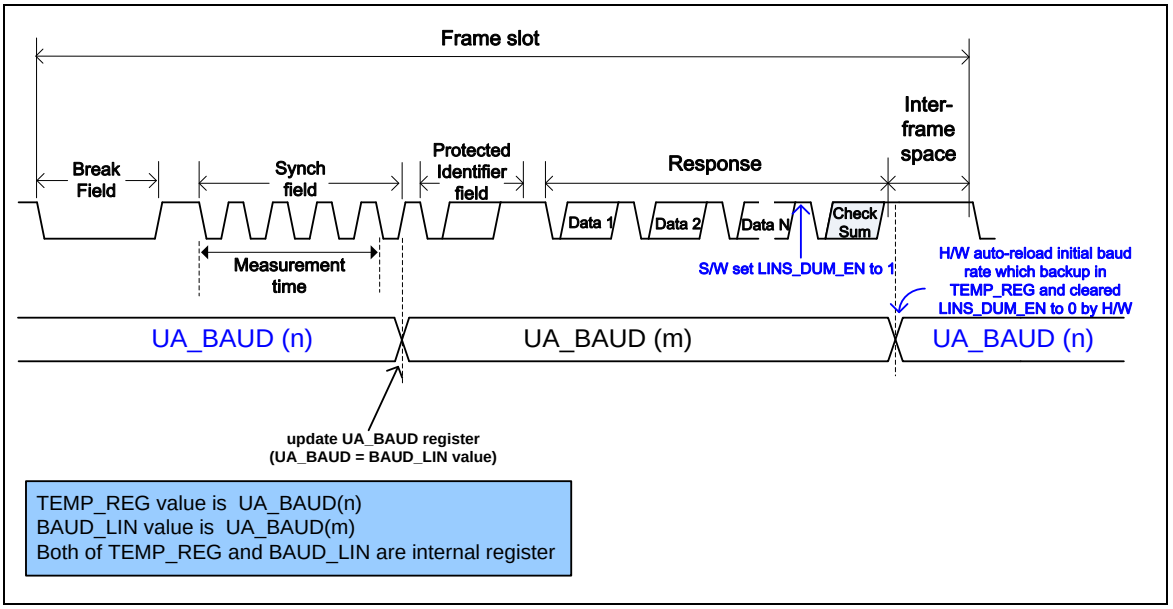


图 5-71 当 LINS_DUM_EN (UA_LIN_CTL[3]) = 1 时自动重新同步模式下 UA_BAUD 更新次序

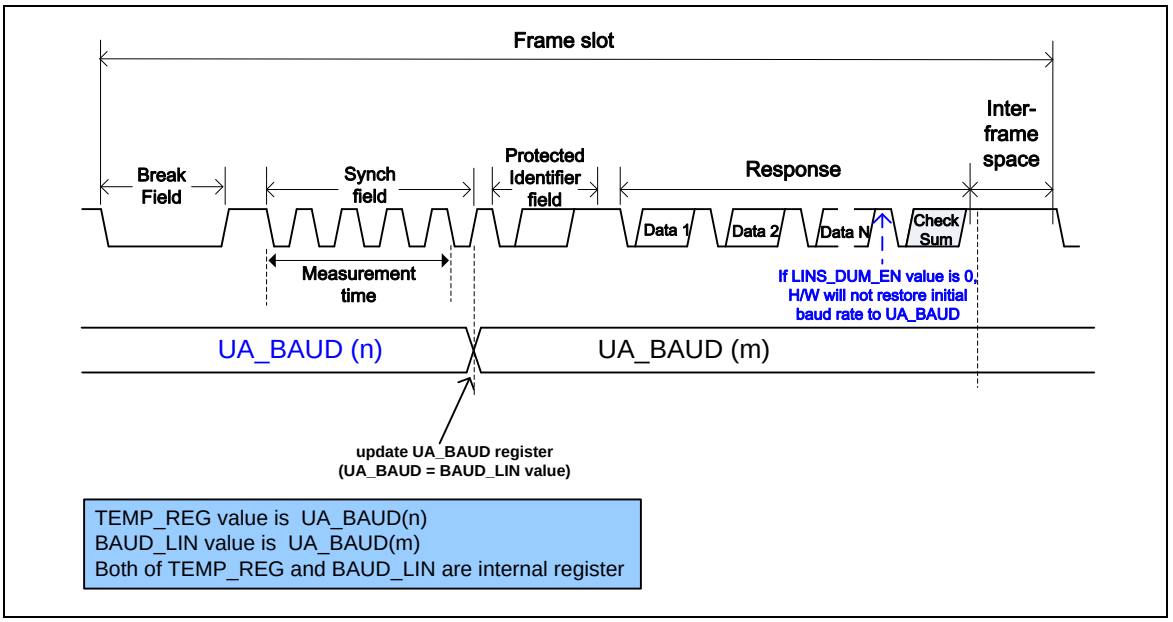


图 5-72 当 LINS_DUM_EN (UA_LIN_CTL[3])= 0 时自动重新同步模式下 UA_BAUD 更新次序

同步域误差错误

自动重新同步模式下，控制器将检测同步域的误差错误。误差错误检测比较当前波特率和接收到的同步域的波特率。两个检测被同步执行.

检查1:根据同步域的第一个下降沿和最后一个下降沿的测量值

- 如果误差大于14.84%, 报头错误标志LINS_HERR_F (UA_LIN_SR[1])将被置位
- 如果误差小于14.06%, 报头错误标志LINS_HERR_F (UA_LIN_SR[1])不会被置位.
- 如果误差在14.84% 和 14.06%之间, 报头错误标志LINS_HERR_F (UA_LIN_SR[1])可能被置位也可能没有被置位 (取决于数据失相)

检查2: 根据同步域的每一个下降沿的测量值.

- 如果误差大于18.75%,报头错误标志LINS_HERR_F (UA_LIN_SR[1])将被置位
- 如果误差小于15.62%,报头错误标志LINS_HERR_F (UA_LIN_SR[1])不会被置位
- 如果误差在18.75% 和 15.62%之间, 报头错误标志LINS_HERR_F (UA_LIN_SR[1])可能被置位也可能没有被置位(取决于数据失相)

注意: 误差检测基于当前波特率时钟。因而，为了保证误差检测的正确性，通过设定LINS_DUM_EN (UA_LIN_CTL[3])位，新的break域收到之前，波特率必须重新加载为初始值(每个checksum收到之前，推荐设定LINS_DUM_EN (UA_LIN_CTL[3])位为1)

LIN 报头错误侦测

LIN 从机模式下, 当用户通过设定LINS_HDET_EN (UA_LIN_CTL[1])使能报头检测功能时, 硬件将处理报头检测流程。如果报头有错误，LIN 报头错误标志LIN_HERR_F (UA_LIN_SR[1])将被置位，如果LIN_IEN (UA_IER[8]) =1，中断将发生。报头错误被检测到时，用户必须写1到LINS_SYNC_F (UA_LIN_SR[3])位来复位检测电路以重新检测新的报头

如果下列一个条件发生，LIN 报头错误标志LIN_HERR_F (UA_LIN_SR[1])将被置位

- Break分隔符太短 (小于 0.5 比特时间).
- 同步域或者帧ID域帧错误
- 同步域数据不是0x55 (非自动重新同步模式)
- 同步域误差错误(自动重新同步模式).
- 同步域测量超时 (自动重新同步模式)
- LIN报头接收超时

5.13.5.9 *RS-485功能模式*

UART控制器另一个可选择的功能是RS-485功能（用户必须设置UA_FUN_SEL [1:0]为 “11”来使能RS-485功能），方向控制则由异步串口的RTS脚来控制。RS-485收发器的驱动控制是通过RTS控制信号来驱动的。RS-485模式下的RX和TX大多数特性与UART相同。

RS-485模式，控制器可以配置成 RS-485 可寻址的从机模式，RS-485 主机发送可通过设置奇偶检验位 (9^{th} bit) 为 1标识地址特性。对于数据特性，奇偶检验位设置为 0。设置寄存器 UA_LCR 控制第9位（PBE(UA_LCR[3]), EPE(UA_LCR[4])和SPE(UA_LCR[5])被置位时，第9位发送 0；PBE 和 SPE 置位，EPE清零时，第 9 位发送1）。

该控制器支持三种操作模式：RS-485 普通多点操作模式 (NMM)，RS-485 自动地址识别模式 (AAD) 和 RS-485 自动方向控制模式 (AUD)，可通过UA_ALT_CSR寄存器的设置选择其中一种工作模式，通过设置DLY (UA_TOR [15:8])可以设置上一个停止位与下一个开始位之间的延迟时间。

RS-485 普通多点操作模式 (NMM)

RS-485 普通多点操作模式(RS485_NMM(UA_ALT_CSR[8]) = 1)，首先，软件决定在检测到地址字节之前的数据是否存储到 RX-FIFO。如果软件想忽略在检测到地址之前的任何数据，流程是设置RX_DIS (UA_FCR [8])，然后使能RS485_NMM (UA_ALT_CSR [8])，接收器将会忽略数据，直到检测到地址字节 (bit9 =1) 并且地址字节数据存储到RX-FIFO。如果软件想接收在检测到地址字节之前的任何数据，流程是禁用RX_DIS (UA_FCR [8])，然后使能RS485_NMM (UA_ALT_CSR [8])，接收器将接收任何数据。

如果检测到地址字节 (bit9 =1)，会产生一个中断到 CPU，软件可以通过设置RX_DIS (UA_FCR [8])，决定是否使能或禁用接收器接收数据。如果使能接收器，就会接收所有字节数据并存储到 RX-FIFO。如果设置RX_DIS (UA_FCR [8])位禁用接收器，会忽略所有接收到的字节数据，直到检测到下一个地址字节。当检测到地址字节后，控制器将清除RX_DIS (UA_FCR [8])位且地址字节数据将存储到RX-FIFO。

RS-485自动地址识别工作模式(AAD)

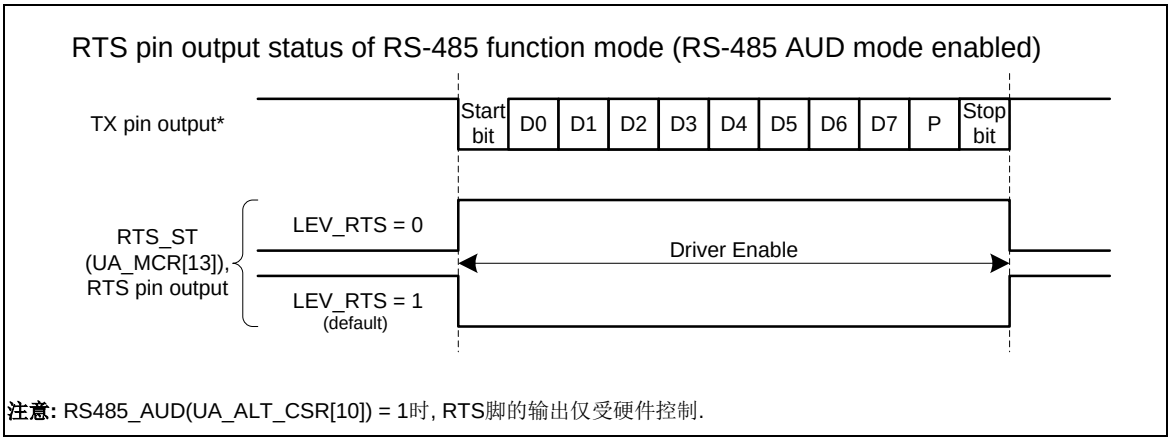
RS-485 自动地址识别模式(RS485_AAD(UA_ALT_CSR[9]) = 1)，接收器在检测到地址字节 (bit9=1) 并且地址字节数据与ADDR_MATCH (UA_ALT_CSR[31:24])的值相匹配之前将忽略所有数据。地址字节数据将存储在 RX-FIFO。所有接收字节数据将被接收并存储于 RX-FIFO 直到地址字节或数据字节与(UA_ALT_CSR[31:24])的值不匹配。

RS-485自动方向模式 (AUD)

RS-485 控制器的另一个功能是 RS-485 自动方向控制功能(RS485_AUD(UA_ALT_CSR[10) = 1)。RS-485 通过 RTS 驱动控制异步串口的控制信号，使能RS-485 驱动器。RTS 连接到RS-485 驱动器，设置RTS线为高（逻辑1）使能RS-485 驱动器。设置 RTS 为低（逻辑0），使驱动器进入 tri-state 状态。用户通过设置寄存器 UA_MCR 中的 LEV_RTS 位改变 RTS 驱动电平

以下框图展示了AUD模式下RS-485 RTS驱动电平。RTS脚在TX数据发送阶段自动驱动收发器。

设置 LEV_RTS(UA_MCR[9]) 位可以控制 RTS 脚的输出电平。用户可以通过读 RTS_ST(UA_MCR[13])位来知道RTS脚上实际的输出逻辑电平。

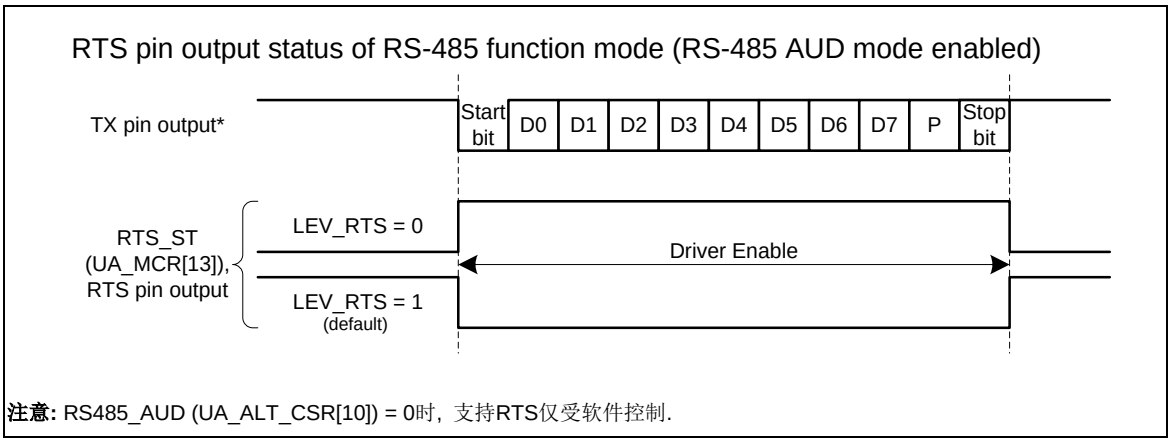


注意: RS485_AUD(UA_ALT_CSR[10]) = 1时, RTS脚的输出仅受硬件控制.

图 5-73 自动方向模式下的 RS-485 RTS 驱动电平

下图展示了通过软件控制(RS485_AUD(UA_ALT_CSR[10])=0)RS-485 RTS脚的驱动电平。RTS驱动电平通过RTS(UA_MCR[1])位来控制。

设置LEV_RTS(UA_MCR[9])位可以控制RTS脚的输出与RTS(UA_MCR[1])控制位是否反向。用户可以读RTS_ST(UA_MCR[13])位来知道RTS脚上实际的逻辑电平。

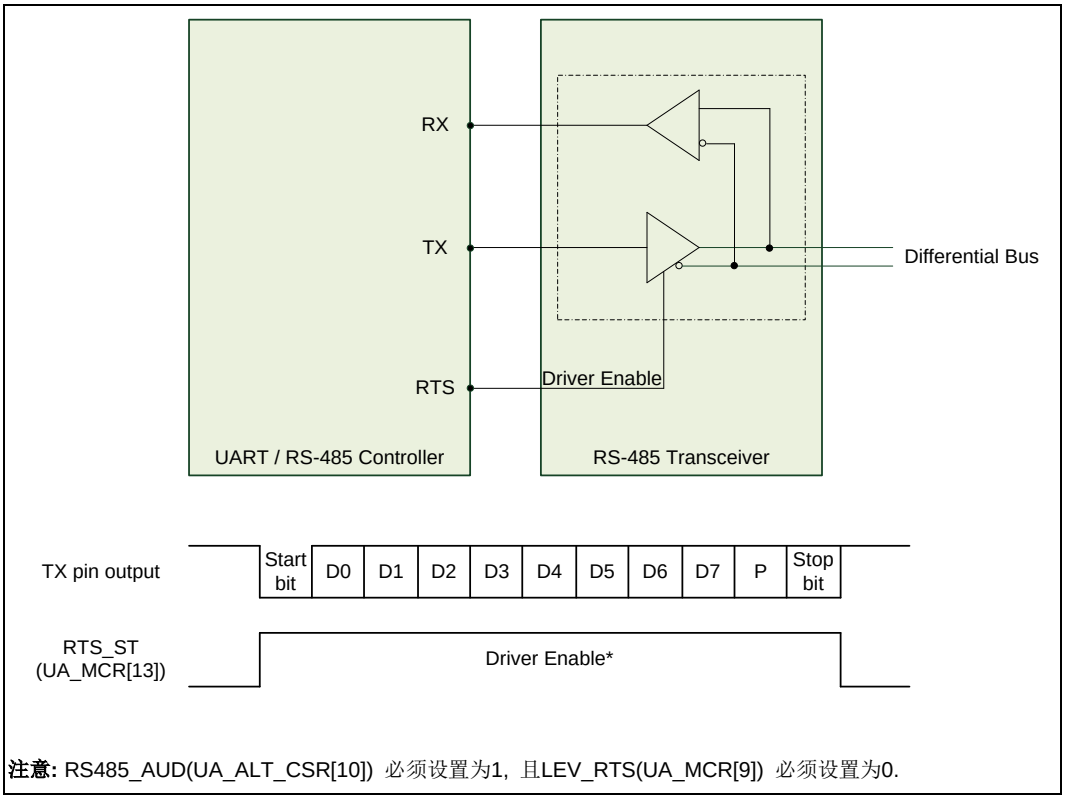


注意: RS485_AUD (UA_ALT_CSR[10]) = 0时，支持RTS仅受软件控制.

图 5-74 RS-485 RTS 软件控制下的驱动电平

编程流程示例：

1. 设置 UA_FUN_SEL 中的 FUN_SEL 选择 RS-485 功能
2. 设置RX_DIS (UA_FCR[8])位使能或禁用 RS-485 接收器
3. 设置RS485_NMM (UA_ALT_CSR[8])或 RS485_AAD (UA_ALT_CSR[9])模式
4. 如果选择RS485_AAD (UA_ALT_CSR[9])模式 ， ADDR_MATCH (UA_ALT_CSR[31:24]) 需设置成自动地址匹配值
5. 设置RS485_AUD (UA_ALT_CSR[10])来决定是否为自动方向控制



注意: RS485_AUD(UA_ALT_CSR[10]) 必须设置为1, 且LEV_RTS(UA_MCR[9]) 必须设置为0.

图 5-75 RS-485 帧结构

5.13.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移地址	R/W	描述	复位值
UART 基地址: **UART0_BA = 0x4005_0000** **UART1_BA = 0x4015_0000** **UART2_BA = 0x4015_4000**				
UA_RBR **x=0,1,2**	UARTx_BA+0x00	R	UART 接收缓冲寄存器	未定义
UA_THR **x=0,1,2**	UARTx_BA+0x00	W	UART 发送保持寄存器	未定义
UA_IER **x=0,1,2**	UARTx_BA+0x04	R/W	UART 中断使能寄存器	0x0000_0000
UA_FCR **x=0,1,2**	UARTx_BA+0x08	R/W	UART FIFO 控制寄存器	0x0000_0101
UA_LCR **x=0,1,2**	UARTx_BA+0x0C	R/W	UART线控寄存器	0x0000_0000
UA_MCR **x=0,1**	UARTx_BA+0x10	R/W	UART Modem 控制寄存器	0x0000_0200
UA_MSR **x=0,1**	UARTx_BA+0x14	R/W	UART Modem 状态寄存器	0x0000_0110
UA_FSR **x=0,1,2**	UARTx_BA+0x18	R/W	UART FIFO状态寄存器	0x1040_4000
UA_ISR **x=0,1,2**	UARTx_BA+0x1C	R/W	UART Interrupt 状态寄存器	0x0000_0002
UA_TOR **x=0,1,2**	UARTx_BA+0x20	R/W	UART 定时溢出寄存器	0x0000_0000
UA_BAUD **x=0,1,2**	UARTx_BA+0x24	R/W	UART波特率分频寄存器	0x0F00_0000
UA_IRCR **x=0,1,2**	UARTx_BA+0x28	R/W	UART IrDA控制寄存器	0x0000_0040
UA_ALT_CSR **x=0,1,2**	UARTx_BA+0x2C	R/W	UART选择控制/状态寄存器	0x0000_000C
UA_FUN_SEL **x=0,1,2**	UARTx_BA+0x30	R/W	UART功能选择寄存器	0x0000_0000
UA_LIN_CTL **x=0,1,2**	UARTx_BA+0x34	R/W	UART LIN 控制寄存器	0x000C_0000

UA_LIN_SR **x=0,1,2**	UARTx_BA+0x38	R/W	UART LIN 状态寄存器	0x0000_0000

5.13.7 寄存器描述

UART 接收缓冲区寄存器 (UA_RBR)

寄存器	偏移地址	R/W	描述	复位值
UA_RBR **x=0,1,2**	UARTx_BA+0x00	R	UART接收缓冲区寄存器	未定义

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

位	描述	
[31:8]	保留	保留.
[7:0]	**RBR**	接收缓冲区寄存器**(只读)** 读该寄存器, UART 将返回从RX脚接收到的8位数据(LSB 优先).

UART 发送保持寄存器 (UA_THR)

寄存器	偏移地址	R/W	描述	复位值
UA_THR x=0,1,2	UARTx_BA+0x00	W	UART发送保持寄存器	未定义

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

位	描述	
[31:8]	保留	保留.
[7:0]	**THR**	**发送保持寄存器** 写数据到该寄存器，数据将会保存到发送FIFO. UART控制器将会通过TX脚把存放在FIFO里的最前面的数据发送出去

UART 中断使能寄存器 (UA_IER)

寄存器	偏移地址	R/W	描述	复位值
UA_IER x=0,1,2	UARTx_BA+0x04	R/W	UART 中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
DMA_RX_EN	DMA_TX_EN	AUTO_CTS_EN	AUTO_RTS_EN	TIME_OUT_EN	保留		LIN_IEN
7	6	5	4	3	2	1	0
保留	WAKE_EN	BUF_ERR_IEN	TOUT_IEN	MODEM_IEN	RLS_IEN	THRE_IEN	RDA_IEN

位	描述	
[31:16]	保留	保留.
[15]	DMA_RX_EN	**RX DMA 使能位(UART2 通道无效)** 该位可以使能或禁止RX DMA功能 0 = RX DMA 禁止. 1 = RX DMA 使能.
[14]	DMA_TX_EN	**TX DMA Enable Bit (UART2 通道无效)** 该位可使能或禁止TX DMA功能 0 = TX DMA 禁止. 1 = TX DMA 使能.
[13]	AUTO_CTS_EN	**CTS自动流控使能位 (UART2 通道无效)** 0 = CTS自动流控禁止. 1 = CTS自动流控使能. 当 CTS 自动流控使能后，当nCTS输入有效， UART 会发送数据到外部设备。
[12]	AUTO_RTS_EN	**RTS 自动流控使能位 (UART2 通道无效)** 0 = RTS 自动流控禁止 1 = RTS 自动流控使能 当 RTS 自动流控使能后，如果RX FIFO中字节的数量等于RTS_TRI_LEV (UA_FCR[19:16])，UART会自动禁止RTS信号
[11]	TIME_OUT_EN	超时计数器使能位 0 = 超时计数器禁止 1 = 超时计数器使能.
[10:9]	保留	保留.

[8]	**LIN_IEN**	**LIN 总线中断使能位** 0 = **LIN 总线中断禁止** 1 = **LIN 总线中断使能** **注:** 该位用于LIN 功能模式
[7]	保留	保留.
[6]	**WAKE_EN**	**UART 唤醒功能使能(位UART2 通道无效)** 0 = UART 唤醒功能禁止. 1 =UART 唤醒功能使能，当芯片进入掉电模式，一个外部CTS信号改变会使芯片从掉电模式中唤醒
[5]	**BUF_ERR_IEN**	**缓存错误中断使能** 0 = INT_BUF_ERR 禁止 1 = INT_BUF_ERR 使能
[4]	**TOUT_IEN**	**RX超时中断使能位** 0 = TOUT_INT 中断关闭. 1 = TOUT_INT 中断使能.
[3]	**MODEM_IEN**	**Modem 状态中断使能位 (UART2 通道无效)** 0 = MODEM_INT 状态中断关闭 1 = MODEM_INT状态中断使能..
[2]	**RLS_IEN**	**Receive 线状态中断使能位** 0 = RLS_INT 关闭. 1 = RLS_INT 使能.
[1]	**THRE_IEN**	**发送保持寄存器空中断使能位** 0 = THRE_INT 关闭 1 = THRE_INT 使能.
[0]	**RDA_IEN**	**接收数据可用中断使能位** 0 = RDA_INT关闭 1 = RDA_INT使能.

UART FIFO 控制寄存器(UA_FCR)

寄存器	偏移地址	R/W	描述	复位值
UA_FCR x=0,1,2	UARTx_BA+0x08	R/W	UART FIFO 控制寄存器	0x0000_0101

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

位	描述	
[31:20]	保留	保留.
[19:16]	**RTS_TRI_LEV**	**RTS 自动流控触发阈值(UART2 通道无效)** 0000 = RTS 触发阈值为1 byte. 0001 = RTS触发阈值为4 bytes. 0010 = RTS触发阈值为8 bytes. 0011 = RTS触发阈值为14 bytes. 0100 = RTS触发阈值为30/14 bytes (高速/常速). 0101 = RTS触发阈值为46/14 bytes (高速/常速).. 0110 = RTS触发阈值为62/14 bytes (高速/常速). Other = 保留. 注：该区域用于自动 RTS 流控制
[15:9]	保留	保留.
[8]	**RX_DIS**	**接收器禁止设置位** 是否禁用接收器（置 1 禁用接收器） 0 =接收器使能 1 =接收器禁止 注：该位用于 RS-485 普通多点模式。需要在UA_ALT_CSR [RS-485_NMM]之前设置
[7:4]	**RFITL**	**RX FIFO 中断 (INT_RDA) 触发级别** 当FIFO 接收字节数等于 RFITL 后，RDA_IF 将被置位（如果UA_IER [RDA_IEN] 使能，将产生中断）。 0000 = RX FIFO 触发中断阈值为 1 byte. 0001 = RX FIFO 触发中断阈值为4 bytes. 0010 = RX FIFO 触发中断阈值为8 bytes. 0011 = RX FIFO 触发中断阈值为14 bytes.

		0100 = RX FIFO 触发中断阈值为30/14 bytes （高速/常速） 0101 = RX FIFO 触发中断阈值为46/14 bytes （高速/常速） 0110 = RX FIFO 触发中断阈值为62/14 bytes （高速/常速） 其它 = 保留.
[3]	**保留**	**保留**.
[2]	**TFR**	**TX 软件复位** 当TFR置位， 发送 FIFO 内的所有数据和 TX 内部状态机将被清除。 0 =不影响. 1 =该位写 1 将复位 TX 内部状态机和指针 注：该位至少 3个 UART 时钟周期后会自动清零
[1]	**RFR**	**RX 软件复位** 当RFR被置位， 接收 FIFO 内的所有数据和 RX 内部状态机将被清除。 0 = 不影响. 1 =该位写 1 将复位 RX 内部状态机和指针. 注：该位至少 3个 UART 时钟周期后会自动清零。
[0]	**保留**	**保留.**

UART 线控制寄存器 (UA_LCR)

寄存器	偏移地址	R/W	描述	复位值
UA_LCR x=0,1,2	UARTx_BA+0x0C	R/W	UART 线控寄存器	0x0000_0000

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

位	描述	
[31:7]	保留	保留.
[6]	**BCB**	**Break 控制位** 当该位被置逻辑 1，串行数据输出 (TX) 将强制到 Spacing 状态 (logic 0)。该位仅作用于 TX ，对传输逻辑不起作用。
[5]	**SPE**	**Stick 奇偶校验使能** 1 =如果PBE (UA_LCR[3]) 和 EBE (UA_LCR[4])为逻辑 1，奇偶校验位发送和检验值为逻辑 0。如果PBE (UA_LCR[3]) 是1 ，EBE (UA_LCR[4])是 0，则奇偶校验位发送和检验值为 1。 0 = 禁用 Stick 奇偶校验
[4]	**EPE**	**Even 奇偶校验使能** 1 = 逻辑 1的偶数数目在每个字中被发送和检验 0 = 逻辑 1的奇数数目在每个字节中被发送和检验 该位只在PBE (UA_LCR[3])置位时有效。
[3]	**PBE**	**奇偶校验位使能** 1 = 每一个发送字符中都产生奇偶校验位，对每一个传进来的数据进行奇偶校验位检测 0 = 无奇偶校验位
[2]	**NSB**	**“停止位”个数** 1= 当发送数据，选择 5-位 字长度时，产生 1.5 “ STOP bit” 当选择 6-, 7- 和 8-位字长度时，产生 2 “STOP bit” 0= 当发生数据时，产生 1 “ STOP bit”
[1:0]	**WLS**	**字长度选择** 00 =字长是 5-bit. 01 =字长是6-bit. 10 =字长是7-bit.

		11 =字长是8-bit.

UART MODEM 控制寄存器 (UA_MCR) (UART2 通道无效)

寄存器	偏移地址	R/W	描述	复位值
UA_MCR **x=0,1**	UARTx_BA+0x10	R/W	UART Modem 控制寄存器	0x0000_0200

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

位	描述	
[31:14]	保留	保留.
[13]	**RTS_ST**	**RTS 脚状态(只读) (UART2 通道无效)** 该位的值对应于RTS脚的输出电平 0 = RTS脚为低电平逻辑状态 1 = RTS脚为高电平逻辑状态.
[12:10]	保留	保留.
[9]	**LEV_RTS**	**RTS 脚的有效电平(UART2 通道无效)** 该位定义了RTS输出脚的有效电平 0 = RTS 脚输出为高电平有效. 1 = RTS 脚输出为低电平有效. **注意1:** UART功能模式请参考图5-62 和图5-63 **注意2:** RS-485 功能模式请参考图图5-73 和 图5-74
[8:2]	保留	保留.
[1]	**RTS**	**RTS (请求发送) 信号控制(UART2 通道无效)** 该位直接控制内部RTS脚信号是否有效，然后使用LEV_RTS位的配置驱动RTS脚输出 0 = RTS 信号有效. 1 = RTS 信号无效. **注意1:** UART 功能模式下，当RTS自动流控被使能后，RTS信号控制位无效 **注意2:** RS-485模式下，当RS-485自动方向模式（AUD）被使能后，RTS信号控制位无效
[0]	保留	保留.

UART Modem 状态寄存器 (UA_MSR) (UART2 通道无效)

寄存器	偏移地址	R/W	描述	复位值
UA_MSR x=0,1	UARTx_BA+0x14	R/W	UART Modem 状态寄存器	0x0000_0110

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

位	描述	
[31:9]	保留	保留.
[8]	**LEV_CTS**	**CTS 脚有效电平** 该位定义了CTS输入脚的有效电平 0 = CTS输入脚高电平有效 1 = CTS输入脚低电平有效 **注意:** 请参考 图5-61
[7:5]	保留	保留.
[4]	**CTS_ST**	**CTS 脚状态(只读) (UART2 通道无效)** 该位对应于CTS脚输入逻辑状态 0 = CTS脚输入状态为低电平 1 = CTS脚输入状态为高电平 **注意:**当UART控制器外围时钟被使能，且CTS功能被使能，该位才有效。
[3:1]	保留	保留.
[0]	**DCTSF**	**检测到 CTS 引脚电平改变标志 (只读) (UART2 通道无效)** 如果CTS输入脚上有电平变化，该位将被置位，如果MODEM_IEN (UA_IER [3])位被置位，将会产生Modem中断。 0 = CTS 输入脚没有电平变化. 1 = CTS输入脚有电平变化. **注意:** 该位只读，写 “1” 清零.

UART FIFO状态寄存器 (UA_FSR)

寄存器	偏移地址	R/W	描述	复位值
UA_FSR x=0,1,2	UARTx_BA+0x18	R/W	UART FIFO 状态寄存器	0x1040_4000

31	30	29	28	27	26	25	24
保留			TE_FLAG	保留			TX_OVER_IF
23	22	21	20	19	18	17	16
TX_FULL	TX_EMPTY	TX_POINTER					
15	14	13	12	11	10	9	8
RX_FULL	RX_EMPTY	RX_POINTER					
7	6	5	4	3	2	1	0
保留	BIF	FEF	PEF	RS485_ADD_DETF	保留		RX_OVER_IF

位	描述	
[31:29]	保留	保留.
[28]	**TE_FLAG**	**发送空标志(只读)** 当TX FIFO (UA_THR)为空并，且最后一个字节的STOP位也已经被发送完毕，那么**TE_FLA**G被硬件置1. 0 = TX FIFO 不为空. 1 = TX FIFO 为空. 注意：当TX FIFO不为空或最后一个字节没有被全部发送完毕，此位被自动清零。
[27:25]	保留	保留.
[24]	**TX_OVER_IF**	**TX溢出错误中断标志 (只读)** 如果TX FIFO (UA_THR)满，如果再向UA_THR写入数据，将会导致此位被置1. 0 = TX FIFO 未溢出 1 = TX FIFO 已溢出. 注意：此位只读，但可以写1清零。
[23]	**TX_FULL**	**发送FIFO满(只读)** 此位用于指示TX FIFO是否已满 0 = TX FIFO 未满 1 = TX FIFO 已满. 当TX FIFO Buffer中的字节数量等于64/16/16(UART0/UART1/UART2),此位将被置1.否则被硬件清零
[22]	**TX_EMPTY**	**发送FIFO 空(只读)** 此位指示TX FIFO是否为空 0 = TX FIFO不为空. 1 = TX FIFO为空. 注意：当TX FIFO中的最后一个字节被发送到发送移位寄存器，硬件将把此位置1.当对THR

		(TX FIFO不为空).此位被清零
[21:16]	**TX_POINTER**	**TX FIFO** 指针(只读) 此位指示TX FIFO缓冲区指针位置。当CPU写一个字节到UA_THR,寄存器，TX_POINTER将累加1.当TX FIFO发送一个字节到发送移位寄存器中，TX_POINTER指针将减1. TX_POINTER显示的最大值是63/15/15 (UART0/UART1/UART2)。当TX FIFO缓冲区所填充数据数量达到64/16/16，TX_FULL将被置1，TX_POINTER被清零。如果TX FIFO中发送一个字节到发送移位寄存器，TX_FULL位将被清零，TX_POINTER显示 63/15/15 (UART0/UART1/UART2)
[15]	**RX_FULL**	**Receiver FIFO** 满(只读) 该位指示RX FIFO 是否已满 0 = RX FIFO 未满 1 = RX FIFO 已满 注意：当RX FIFO缓冲区中的数据数量等于64/16/16(UART0/UART1/UART2)后，此位将被置1，否则被硬件清零。
[14]	**RX_EMPTY**	**Receiver FIFO空**(只读) 此位指示RX FIFO是否为空 0 = RX FIFO不为空. 1 = RX FIFO为空. 注意：当RX FIFO中最后一个字节被CPU读取后，硬件将对此位置1，UART接收到新数据后此位将被清零。
[13:8]	**RX_POINTER**	**RX FIF**O指针(只读) 此位指示RX FIFO缓冲区指针。当UART从外部设备接收到一个字节，RX_POINTER将累加1.当RX FIFO的数据被CPU读取一个字节，RX_POINTER将递减1. RX_POINTER显示的最大值是63/15/15 (UART0/UART1/UART2)。当RX FIFO的数据达到64/16/16，RX_FULL位将被置1，RX_POINTER显示0，RX FIFO 当中的数据被CPU读取一个后，RX_FULL 将被清零，RX_POINTER显示63/15/15 (UART0/UART1/UART2)
[7]	保留	保留.
[6]	**BIF**	**Break**中断标志位(只读) 每当接收到数据输入 (RX) 维持在 “spacing state” (logic 0) 的时间长于一个全字的传输时间（即“start bit” + data bits + parity + stop bits 的总时间），该位置 1。当 CPU 向该位写 1，该位重置。 0 =没有Break中断产生. 1 =有Break中断产生. 注意：此位只读。但是可以写1清零
[5]	**FEF**	帧错误标志(只读) 每当接收到的字符没有合法的 “停止位”（即停止位跟着最后的数据位后或奇偶校验位检测为0）该位置 1。当 CPU 向该位写 1，该位重置 0 =没有真错误产生 1 =有真错误产生. 注：该位只读，但是可以写 1 清零。
[4]	**PEF**	奇偶校验错误标志(只读) 每当接收到的字符没有合法的奇偶校验位，该位置 1。当 CPU 向该位写 1，该位重置。 0 =.没有奇偶校验错误产生 1 =有奇偶校验错误产生 注：该位只读，但是可以写 1 清零。

[3]	**RS485_ADD_DETF**	RS-485 地址数据检测标志(只读) 0 =接收到的数据没有地址位标记 (bit 9 ='1'). 1 =接收到的数据有地址位标记(bit 9 ='1'). 注意1：此位用于RS-485功能模式，且RS485_ADD_EN (UA_ALT_CSR[15])位被置1使能地址检测模式 注意2：该位只读，但是可以写 1 清零。
[2:1]	保留	保留.
[0]	**RX_OVER_IF**	RX 溢出错误标志(只读) 当RX FIFO溢出时被置1 如果接收到的数据数量大于 RX_FIFO (UA_RBR)64/16/16 字节（UART0/UART1/UART2），该位置位 0 = RX FIFO未溢出. 1 = RX FIFO溢出 注意：该位只读，但是可以写 1 清零

UART中断状态控制寄存器(UA_ISR)

寄存器	偏移地址	R/W	描述	复位值
UA_ISR x=0,1,2	UARTx_BA+0x1C	R/W	UART中断状态寄存器	0x0000_0002

31	30	29	28	27	26	25	24
保留		HW_BUF_ERR_INT	HW_TOUT_INT	HW_MODEM_INT	HW_RLS_INT	保留	
23	22	21	20	19	18	17	16
保留		HW_BUF_ERR_IF	HW_TOUT_IF	HW_MODEM_IF	HW_RLS_IF	保留	
15	14	13	12	11	10	9	8
LIN_INT	保留	BUF_ERR_INT	TOUT_INT	MODEM_INT	RLS_INT	THRE_INT	RDA_INT
7	6	5	4	3	2	1	0
LIN_IF	保留	BUF_ERR_IF	TOUT_IF	MODEM_IF	RLS_IF	THRE_IF	RDA_IF

位	描述	
[31:30]	保留	保留.
[29]	**HW_BUF_ERR_INT**	**DMA 模式下，Buffer 错误中断标志(只读)** 如果BUF_ERR_IEN (UA_IER[5])和HW_BUF_ERR_IF (UA_ISR[5])都被置为1，该位置被置1。 0 = DMA模式下，没有buffer 错误中断产生。. 1 = DMA模式下，buffer 错误中断产生。.
[28]	**HW_TOUT_INT**	**DMA模式，超时溢出中断标志(只读)** 如果TOUT_IEN (UA_IER[4])和HW_TOUT_IF(UA_ISR[20])都被置为1，该位置1。 0 = DMA 模式下，没有超时溢出中断产生。 1 = DMA 模式下，有超时溢出中断产生。
[27]	**HW_MODEM_INT**	**DMA 模式，MODEM 状态中断标志(只读) (UART2 通道无效)** 如果MODEM_IEN(UA_IER[3])和HW_MODEM_IF(UA_ ISR[3])都被置为1，该位置 1。 0 = DMA 模式下，没有Modem 中断产生。 1 = DMA 模式下，有Modem 中断产生。
[26]	**HW_RLS_INT**	**DMA 模式，接收线状态中断标志 ((只读)** 如果RLS_IEN (UA_IER[2])和HW_RLS_IF(UA_ISR[18])都被置为1，该位置 1。 0 = DMA模式下，没有RLS 中断产生 1 = DMA模式下，有RLS 中断产生.
[25:22]	保留	保留.
[21]	**HW_BUF_ERR_IF**	**DMA模式下，Buffer 错误中断标志(只读)** 当TX或RX FIFO溢出（TX_OVER_IF (UA__FSR[24])或RX_OVER_IF (UA_FSR[0])被置位），该位置位。 当BUF_ERR_IF (UA_ISR[5])被置位，传输也许不正确。如果

		BUF_ERR_IEN (UA_IER [5])被使能，buffer 错误中断将产生。 0 = 没有buffer 错误中断标志产生 1 =有buffer 错误中断标志产生 注：当TX_OVER_IF (UA_FSR[24]]) 和 RX_OVER_IF (UA_FSR[0])都被清除时，该位被清除。
[20]	**HW_TOUT_IF**	**DMA模式下，定时溢出中断标志(只读)** 当RX FIFO非空而且RX FIFO无活动发生，定时溢出计数器等于TOIC (UA_TOR[7:0])时，该位置位。如果TOUT_IEN (UA_IER [4])使能，定时溢出中断将产生。 0 =没有定时溢出中断标志产生 1 =有定时溢出中断标志产生 注：该位只读，用户可以读 UA_RBR（RX在活动时）清除该位。
[19]	**HW_MODEM_IF**	**DMA模式下，MODEM中断标志(只读) (UART2 通道无效)** 当 CTS 管脚出现状态改变 (DCTSF (US_MSR[0] =1))，该位置位。如果MODEM_IEN (UA_IER [3])使能，Modem中断将产生 0 =没有Moden中断标志产生. 1 =有Moden中断标志产生. 注：该位只读，当DCTSF(US_MSR[0])位被写 1 清除时，该位清零。
[18]	**HW_RLS_IF**	**DMA模式下，接收线状态标志(只读)** 当RX 接收数据有 parity error, framing error 或 break error（至少BIF (UA_FSR[6]), FEF (UA_FSR[5])和PEF (UA_FSR[4]) 3 位中有1位被置）该位置位。如果RLS_IEN (UA_IER [2])被使能，RLS 中断将产生。 0 =没有RLS中断标志产生 1 =有RLS中断标志产生 **注意1:**在 RS-485 功能模式下，该域包括"接收检测任一接收到的地址字节字符 (bit9 = '1')位" **注意2:** UART 功能模式下，此位只读，当BIF(UA_FSR[6]) ，FEF(UA_FSR[5]) 和PEF(UA_FSR[4])都被清零后，此位清零 **注意3:** RS-485 功能模式下，此位只读，当 BIF(UA_FSR[6]) ，FEF(UA_FSR[5]),PEF(UA_FSR[4])和RS485_ADD_DETF (UA_FSR[3]) 所有位被清零，此位清零
[17:16]	保留	保留.
[15]	**LIN_INT**	**LIN总线中断标志(只读)** 如果LIN_IEN (UA_IER[8])和LIN _IF(UA_ISR[7])都被置1，该位置 1 0 =没有LIN总线中断产生 1 =有LIN总线中断产生.
[14]	**保留**	保留.
[13]	**BUF_ERR_INT**	**Buffer错误中断标志(只读)** 如果BUF_ERR_IEN(UA_IER[5] and BUF_ERR_IF(UA_ISR[5])都被置1，该位置 1 0 =没有buffer错误中断产生. 1 =有buffer错误中断产生
[12]	**TOUT_INT**	**定时溢出中断标志(只读)** 如果TOUT_IEN(UA_IER[4])和TOUT_IF(UA_ISR[4])都被置1，该位置 1 0 =没有定时溢出中断产生 1 =有定时溢出中断产生.

[11]	**MODEM_INT**	**MODEM状态中断标志(只读) (UART2通道无效)** 如果MODEM_IEN(UA_IER[3]和MODEM_IF(UA_ISR[4])都被置1，该位置 1 0 =没有Modem 中断产生. 1 =有Modem 中断产生..
[10]	**RLS_INT**	**接收线状态中断标志(只读)** 如果RLS_IEN (UA_IER[2])和RLS_IF(UA_ISR[2])都被置1，该位置 1 0 =没有RLS中断产生 1 =有RLS中断产生.
[9]	**THRE_INT**	**发送保持寄存器空中断标志(只读)** 如果THRE_IEN (UA_IER[1])和THRE_IF(UA_SR[1])都被置1，该位置 1 0 =没有THRE中断产生 1 =有THRE中断产生
[8]	**RDA_INT**	**接收可用数据中断标志(只读)** 如果RDA_IEN (UA_IER[0])和RDA_IF (UA_ISR[0])都被置1，该位置 1 0 =没有RDA中断产生. 1 =有RDA中断产生.
[7]	**LIN_ IF**	**LIN总线标志(只读)** 当LIN从机头部侦测到(LINS_HDET_F (UA_LIN_SR[0] =1))，LIN break 侦测到LIN_BKDET_F(UA_LIN_SR[9]=1)，位错误侦测到(BIT_ERR_F(UA_LIN_SR[9]=1)，LIN从机ID校验错误LINS_IDPERR_F(UA_LIN_SR[2] = 1)或LIN从机头部错误侦测到LINS_HERR_F (UA_LIN_SR[1])，该位置1，如果LIN_ IEN (UA_IER [8])被使能，LIN中断产生 0 =LINS_HDET_F, LIN_BKDET_F, BIT_ERR_F, LINS_IDPERR_F和LINS_HERR_F 没有一个标志产生 1 =LINS_HDET_F, LIN_BKDET_F, BIT_ERR_F, LINS_IDPERR_F and LINS_HERR_F至少有一个标志产生 注：该位只读。当LINS_HDET_F(UA_LIN_SR[0]), LIN_BKDET_F(UA_LIN_SR[9]), BIT_ERR_F(UA_LIN_SR[9]), LINS_IDPENR_F (UA_LIN_SR[2]) 和LINS_HERR_F(UA_LIN_SR[1])都被清0，该位清0。
[6]	保留	保留.
[5]	**BUF_ERR_IF**	**Buffer 错误中断标志 (只读)** 当TX FIFO或RX FIFO溢出 (TX_OVER_IF (UA_FSR[24]) 或 RX_OVER_IF (UA_FSR[0])被复位)该位置1. 当BUF_ERR_IF (UA_ISR[5])被置位，传输出错.如果BUF_ERR_IEN (UA_IER [8])被使能, buffer 错误中断将产生 0 = 没有buffer 错误中断标志产生 1 = 有buffer 错误中断标志产生 **注意:** 此位只读。当TX_OVER_IF(UA_FSR[24])和RX_OVER_IF(UA_FSR[0]) 都被清零，该位被清零.
[4]	**TOUT_IF**	**定时溢出中断标志 (只读)** 当 RX FIFO 非空而且 RX FIFO无活动发生，定时溢出计数器等于TOIC 时，该位置位。如果TOUT_IEN (UA_IER [4])使能，定时溢出中断将产生。 0 =没有定时溢出中断标志产生. 1 =有定时溢出中断标志产生 注：该位只读，用户可以读 UA_RBR（RX处于活动状态）清除该位
[3]	**MODEM_IF**	**MODEM 中断标志(只读) (UART2 通道无效)**

		当CTS管脚有状态改变 (DCTSF (UA_MSR[0]) = 1)，该位置位。如果MODEM_IEN (UA_IER [3])被使能，Modem 中断将产生。 0 =没有Modem 中断标志产生. 1 =有Modem 中断标志产生. 注：该位只读，当向DCTSF(UA_MSR[0])写1清除后，该位清除
[2]	**RLS_IF**	**接收线中断标志(只读)** 当 RX 接收数据有 parity error, framing error 或 break error（至少BIF(UA_FSR[6]), FEF(UA_FSR[5])和PEF(UA_FSR[4]) 3位中有1位被置）该位置位。如果RLS_IEN (UA_IER [2])使能，RLS 中断将产 0 =没有RLS中断标志产生 1 =有RLS中断标志产生. **注意1:** 在 RS-485功能模式，该域包括"接收检测任一接收到的地址字节字符 (bit9 = '1')位"，同时，UA_FSR[RS485_ADD_DETF]位也被置1 **注意2:** 该位只读，当BIF(UA_FSR[6]), FEF(UA_FSR[5]) and PEF(UA_FSR[4])三位都被清除后，该位重置为 0 **注意3:** 在RS-485 功能模式下,该位只读，且当BIF(UA_FSR[6]) ，FEF(UA_FSR[5]), PEF(UA_FSR[4]) 和RS485_ADD_DETF (UA_FSR[3]) 所有位被清零，该位清零
[1]	**THRE_IF**	**发送保持寄存器空中断标志(只读)** 当TX FIFO的最后数据传输到发送移位寄存器时，该位置位。如果THRE_IEN (UA_IER[1])被使能，THRE 中断将产生。 0 =没有THRE中断标志产生 1 =有THRE中断标志产生. **注意:** 该位只读，当写数据到 THR（TX FIFO 非空）时，该位将被清除
[0]	**RDA_IF**	**接收可用数据中断标志(只读)** 当 RX FIFO的数据数量等于 RFITL时，RDA_IF(UA_ISR[0])将被置位。如果RDA_IEN (UA_IER [0])被使能，RDA 中断将产生。 0 =没有RDA中断标志产生. 1 =有RDA中断标志产生. 注：该位只读，当 RX FIFO 的未读数据低于阈值(RFITL(UA_FCR[7:4]).时，该位将被清除

UART定时溢出寄存器(UA_TOR)

寄存器	偏移地址	R/W	描述	复位值
UA_TOR x=0,1,2	UARTx_BA+0x20	R/W	UART定时溢出寄存器	0x0000_0000

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

位	描述	
[31:16]	保留	保留.
[15:8]	**DLY**	**TX延迟时间值** 该域用于编程上一次停止位和下一次开始位之间的传输延迟时间
[7:0]	**TOIC**	**定时溢出中断比较器** 当RX FIFO接收到一个新的数据字，定时溢出计数器重置并开始计数 (计数时钟 = 波特率)。一旦超时计数器的内容等于超时中断比较器TOIC (UA_TOR[7:0])，如果此时TOUT_IEN (UA_IER [4])为使能，则接收超时中断(INT_TOUT)产生。新来的数据字或RX FIFO 为空清除TOUT_INT(UA_IER[9])。为了避免接收超时中断一接收一个字符就立即产生，TOIC (UA_TOR[7:0])的值必须设置在 40 到 255 之间。例如，如果TOIC (UA_TOR[7:0])为 40，当 UART 传输设置为1 停止位和无奇偶校验位时，超时中断将在4个字符没有收到之后产生。

UART波特率分频寄存器(UA_BAUD)

寄存器	偏移地址	R/W	描述	复位值
UA_BAUD x=0,1,2	UARTx_BA+0x24	R/W	UART波特率分频寄存器	0x0F00_0000

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

位	描述	
[31:30]	保留	保留.
[29]	**DIV_X_EN**	分频 X 使能位 BRD = 波特率分频值, 波特率的公式为 波特率= Clock / [M * (BRD + 2)]; M的默认值是16. 0 = 分频值X禁用 (M = 16). 1 = 分频值X使能 (M = X+1, 但是DIVIDER_X [27:24] 必须>= 8). 参考表 5-12查看详细信息 注意**:** IrDA模式下，该位须禁止.
[28]	**DIV_X_ONE**	分频X 等于1 0 = 分频值M = X (M = X+1, 但是DIVIDER_X[27:24] 必须>= 8). 1 = 分频值M = 1 (M = 1, 但是BRD [15:0] 必须>= 3). 参考表 5-12 查看详细信息
[27:24]	**DIVIDER_X**	分频X M = X+1.
[23:16]	保留	保留.
[15:0]	**BRD**	波特率分频器 该域表示波特率分频

UART IrDA 控制寄存器(IRCR)

寄存器	偏移地址	R/W	描述	复位值
UA_IRCR x=0,1,2	UARTx_BA+0x28	R/W	UART IrDA控制寄存器	0x0000_0040

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

位	描述	
[31:7]	保留	保留.
[6]	**INV_RX**	**IrDA**反向接收输入信号控制 0 =不反向接收输入信号. 1 =反向接收输入信号.
[5]	**INV_TX**	**IrDA**反向发送信号控制 0 =信号不反向发送. 1 =信号反向发送
[4:2]	保留	保留.
[1]	**TX_SELECT**	**TX_SELECT** 0 = IrDA发送禁止接收使能. 1 = IrDA发送使能接收禁止
[0]	保留	保留.

注意: 在IrDA 模式下, the UA_BAUD (UA_BAUD [29]) 寄存器必须禁止 (波特率公式必须是 Clock / 16 * (BRD)

UART复用控制/状态寄存器(UA_ALT_CSR)

寄存器	偏移地址	R/W	描述	复位值
UA_ALT_CSR x=0,1,2	UARTx_BA+0x2C	R/W	UART复用控制/状态寄存器	0x0000_000C

31	30	29	28	27	26	25	24
ADDR_MATCH							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
RS485_ADD_EN	保留				RS485_AUD	RS485_AAD	RS485_NMM
7	6	5	4	3	2	1	0
LIN_TX_EN	LIN_RX_EN	保留		LIN_BKFL			

位	描述	
[31:24]	**ADDR_MATCH**	地址匹配值寄存器 该位包括 RS-485 地址匹配值. 注：该域用于 RS-485 自动地址检测模式
[23:16]	保留	保留.
[15]	**RS485_ADD_EN**	**RS-485** 地址检测使能位 该位用于使能 RS-485 地址检测使能模式. 0 =地址检测模式禁止. 1 =地址检测模式使能. 注：该位用于 RS-485 任意操作模式.
[14:11]	保留	保留.
[10]	**RS485_AUD**	**RS-485** 自动方向模式 (AUD) 0 =RS-485自动方向操作模式AUD)禁止. 1 =RS-485自动方向操作模式AUD)使能 注：在RS-485_AAD 或 RS-485_NMM 操作模式有效
[9]	**RS485_AAD**	**RS-485** 自动地址检测操作模式 (**AAD)** 0 = RS-485自动地址方向操作模式 (AAD)禁止. 1 =RS-485自动地址方向操作模式 (AAD)使能. 注：在 RS-485_NMM 操作模式下无效
[8]	**RS485_NMM**	**RS-485** 普通多点操作模式 (NMM) 0 = RS-485 普通多点操作模式 (NMM)禁止. 1 = RS-485 普通多点操作模式 (NMM)使能 注：在 RS-485_AAD 操作模式下无效.

[7]	**LIN_TX_EN**	**LIN TX Brea**k 模式使能 0 = LIN TX Break 模式禁止 1 = LIN TX Break 模式使能. 注：当 TX break 域传输操作完成后，该位将被自动清除.
[6]	**LIN_RX_EN**	LIN RX 使能 0 =LIN RX 模式禁止. 1 = LIN RX 模式使能
[5:4]	保留	保留.
[3:0]	**LIN_BKFL**	**UART LIN Break** 域长度 该域表示一个 4-位 LIN TX break 域计数 注意1**:** 该break 域长度为 UA_LIN_BKFL + 1 注意2**:** 根据LIN 规范, 复位值是0xC (break 域长度= 13).

UART 功能选择寄存器(UA_FUN_SEL)

寄存器	偏移地址	R/W	描述	复位值
UA_FUN_SEL x=0,1,2	UARTx_BA+0x30	R/W	UART 功能选择寄存器	0x0000_0000

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

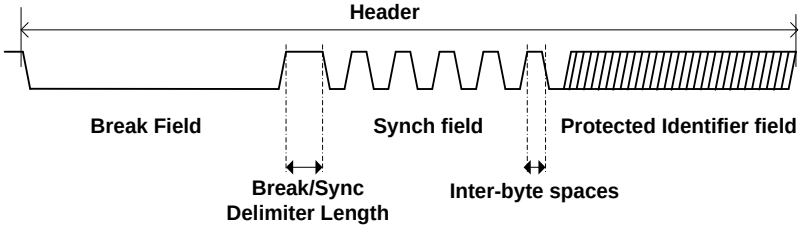
位	描述	
[31:2]	保留	保留.
[1:0]	**FUN_SEL**	功能选择使能位 00 = UART功能使能. 01 = LIN 功能使能 10 = IrDA功能使能 11 = RS-485功能使能.

UART LIN控制寄存器(UA_LIN_CTL)

寄存器	偏移地址	R/W	描述	复位值
UA_LIN_CTL x=0,1,2	UARTx_BA+0x34	R/W	UART LIN控制寄存器	0x000C_0000

31	30	29	28	27	26	25	24
LIN_PID							
23	22	21	20	19	18	17	16
LIN_HEAD_SEL		LIN_BS_LEN		LIN_BKFL			
15	14	13	12	11	10	9	8
保留			BIT_ERR_EN	LIN_RX_DIS	LIN_BKDET_EN	LIN_IDPEN	LIN_SHD
7	6	5	4	3	2	1	0
保留			LIN_MUTE_EN	LINS_DUM_EN	LINS_ARS_EN	LINS_HDET_EN	LINS_EN

位	描述	
[31:24]	LIN_PID	**LIN PID**寄存器 当在LIN功能模式下，该域包含LIN帧ID值。帧ID的校验可以由软件或硬件产生，这取决于LIN_IDPEN (UA_LIN_CTL[9]) 的设置。 如果校验由硬件产生， 用户填ID0~ID5, (LIN_PID[24:29]硬件会计算P0(LIN_PID[30]) 和P1(LIN_PID[31])，否则用户必须填帧ID和校验 PID | Start | ID0 | ID1 | ID2 | ID3 | ID4 | ID5 | P0 | P1 P0 = ID0 xor ID1 xor ID2 xor ID4 P1 = ~(ID1 xor ID3 xor ID4 xor ID5) **注1:** 用户可以填任何 8-bit 值到该域，位 24 表示 ID0 (LSB 优先) **注2:** 该域可以用在 LIN 主机模式或从机模式
[23:22]	LIN_HEAD_SEL	LIN帧头部选择 00 = LIN帧头部包括 “break 域” 01 = LIN 帧头部包括 “break 域” 和 “同步域””. 10 = LIN 帧头部 包括 “break域”, “同步域” 域“帧ID域”. 11 = 保留 注意：该位用作 LIN 主机模式发送帧头部域LIN_SHD (UA_LIN_CTL [8]) = 1)，或用作从机指示从mute模式退出条件(LIN_MUTE_EN (UA_LIN_CTL[4] = 1)。
[21:20]	LIN_BS_LEN	LIN Break/同步分隔符长度 00 = LIN break/同步分隔符长度为1 bit时间. 10 = LIN break/同步分隔符长度为2 bit时间. 10 = LIN break/同步分隔符长度为3 bit时间. 11 = LIN break/同步分隔符长度为4 bit时间.

		 **注:** 该位用于LIN 主机发送帧头部域
[19:16]	**LIN_BKFL**	LIN Break 域长度 该域指示一个4-bit LIN TX break域数量 注意1: 这个寄存器是LIN_BKFL的影子寄存器，用户可以通过设置LIN_BKFL (UA_ALT_CSR[3:0])或LIN_BKFL (UA_LIN_CTL[19:16]对break域的长度进行读/写。 注意2: 该 break 域长度为 LIN_BKFL + 1 注意3: 根据LIN 规范，**复位值**是 12 (break域长度= 13).
[15:13]	保留	保留.
[12]	**BIT_ERR_EN**	位错误侦测使能 0 =位错误侦测功能禁止. 1 =位错误侦测使能. 注意: 在 LIN功能模式下，当位错误发生时，BIT_ERR_F (UA_LIN_SR[9])标志会被置位。如果LIN_IEN (UA_IER[8]) = 1，会产生一个中断
[11]	**LIN_RX_DIS**	**LIN**接收器禁止位 如果接收器使能 (LIN_RX_DIS (UA_LIN_CTL[11]) = 0)，所有接收到的数据字节会被接受并存储在RX-FIFO，如果接收器禁止(LIN_RX_DIS (UA_LIN_CTL[11] = 1)，接收到的所有数据会被忽略 0 = LIN 接收器禁止. 1 = LIN 接收器使能. 注意: 该位仅工作在 LIN功能模式下有效 (FUN_SEL (UA_FUN_SEL[1:0]) = 01)
[10]	**LIN_BKDET_EN**	**LIN Break** 域检测使能位 当检测到连续显性位长度到过11位和一个分隔符，break域结束时LIN_BKDET_F (UA_LIN_SR[8])标志被置位。如果LIN_IEN (UA_IER [8])=1，将会产生中断。 0 = LIN break 域检测禁止 1 = LIN break 域检测使能.
[9]	**LIN_IDPEN**	**LIN ID** 奇偶校验使能位 0 = LIN 帧ID奇偶校验禁止. 1 = LIN 帧ID奇偶校验使能. 注意1: 该位可以让LIN主机用来发送帧头域(LIN_SHD (UA_LIN_CTL[8])) = 1且LIN_HEAD_SEL (UA_LIN_CTL[23:22]) = 10) 或用来使能 LIN 从机接收到的帧 ID奇偶校验检查。 注意2:该位仅在操作标题发送器是在LIN_HEAD_SEL (UA_LIN_CTL[23:22]) = 10时有用。
[8]	**LIN_SHD**	LIN TX 发送帧头使能位 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 LIN_HEAD_SEL (UA_LIN_CTL[23:22]). LIN TX 帧头可以是“break 域” 或 “break 和同步域” 或 “break, 同步和帧 ID 域”，这取决于LIN_HEAD_SEL (UA_LIN_CTL[23:22])位的设置 0 = LIN TX 帧头发送禁止. 1 = LIN TX 帧头发送使能.

		注意1: 这些 寄存器是LIN_SHD (UA_ALT_CSR [7])的影子寄存器;用户可以通过LIN_SHD (UA_ALT_CSR [7])或LIN_SHD (UA_LIN_CTL [8])进行读写设置 注意2: 当发送器帧头域 (可能是 “break” 或 “break + 同步” 或 “break + 同步 + 帧 ID”通过LIN_HEAD_SEL (UA_LIN_CTL[23:22])域选择) 传输操作完成, 该位会自动清0.
[7:5]	保留	保留.

[4]	**LIN_MUTE_EN**	**LIN Mute 模式使能位** **0 = LIN Mute 模式禁止** **1 = LIN Mute 模式使能.** **注意: 退出Mute模式条件，该域的控制和相互作用在 (LIN从机模式) 中解释**
[3]	**LINS_DUM_EN**	LIN从机除频器更新方式使能 0 = UA_BAUD 被软件更新(如果同时没有自动重同步发生) 1 = UA_BAUD在下次接收到字符的时候更新。用户必须在校验和接收之前设置该位. 注意1: 该位仅在 LIN 从模式有效 (LINS_EN (UA_LIN_CTL[0]) = 1) 注意2: 该位用于LIN 从机自动重同步模式. (对于非自动重同步模式，该位应该保持清除) 注意3: 该域的控制和相互作用在0节中解释(带自动重同步的从机模式)
[2]	**LINS_ARS_EN**	LIN 从机自动重同步模式使能位 0 = LIN从机自动重同步模式禁止 1 = LIN从机自动重同步模式使能. 注意1: 该位仅在 LIN 从机模式(LINS_EN (UA_LIN_CTL[0]) = 1) 有效. 注意2: 当工作在自动重同步模式，波特率的设置必须是mode2 (BAUD_M1 (UA_BAUD [29]) 和BAUD_M0 (UA_BAUD [28])必须是 1). 注意3: 该域的控制和相互作用在 6.18.5.8.4.节中解释(带自动重同步的从机模式)
[1]	**LINS_HDET_EN**	LIN从机报头侦测使能位 0 = LIN 从机报头检测禁止. 1 = LIN 从机报头检测使能. 注意1: 该位仅在 LIN 从机模式有效 (LINS_EN (UA_LIN_CTL[0]) = 1) 注意2: 在 LIN功能模式， 当侦测标题域(break + sync + frame ID), LINS_HDET_F (UA_LIN_SR [0]) 标志会被置位，如果LIN_IEN (UA_IER[8]) = 1，会产生一个中断
[0]	**LINS_EN**	LIN从模式使能位 0 = LIN 从模式禁止 1 = LIN 从模式使能

UART LIN状态寄存器(UA_LIN_SR)

寄存器	偏移地址	R/W	描述	复位值
UA_LIN_SR x=0,1,2	UARTx_BA+0x38	R/W	UART LIN状态寄存器	0x0000_0000

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

位	描述	
[31:10]	保留	保留.
[9]	**BIT_ERR_F**	位错误侦测状态标志 (只读). TX传输状态，硬件会监视总线状态，如果输入管脚 (SIN) 状态不等于输出管脚 (SOUT) 状态, BIT_ERR_F (UA_LIN_SR[9])位会被置位。 当发生位错误，如果LIN_IEN (UA_IER[8]) = 1，会产生一个中断 注意1: 该位只读，向该位写1清0 **注意2:** 该位仅当位错误侦测功能使能时有效 (BIT_ERR_EN (UA_LIN_CTL [12]) = 1)
[8]	**LIN_BKDET_F**	**LIN Break** 检测标志(只读) 当一个break 侦测到，该位由硬件置位，通过软件写1清0 0 = LIN break 没有被检测到 1 = LIN break 被检测到. **注意1:** 该位只读，向该位写1清0. **注意2:**该位仅当 LIN break侦测功能使能时有效 (LIN_BKDET_EN (UA_LIN_CTL[10]) =1)
[7:4]	保留	保留.
[3]	**LINS_SYNC_F**	**LIN** 从机同步域. 该位表示在自动重同步模式， LIN同步域正在被分析。 当接收器标题侦测到一些错误，用户必须通过写1到该位复位内部电路来重新搜索新的帧报头 0 =当前字符不在 LIN同步状态. 1 =当前字符在 LIN 同步状态. 注意1：该位仅在 LIN 从机模式有效 (LINS_EN = 1) 注意2: 该位只读，向该位写1清0 注意**3:** 当向该位写1，硬件会重载初始波特率并重新搜索新的帧报头
[2]	**LINS_IDPERR_F**	**LIN** 从机 ID 奇偶错误标志 (只读)

		当接收帧ID奇偶校验错误，该位由硬件置1 0 = 无效. 1 = 收到的帧ID奇偶校验错误. **注意1:** 该位只读，向该位写1清0.. **注意2:**该位仅在 LIN 从机模式有效 (LINS_EN (UA_LIN_CTL [0])= 1) 并使能LIN帧ID奇偶校验功能 (LIN_IDPEN (UA_LIN_CTL [9]))
[1]	**LINS_HERR_F**	**LIN**从机报头错误标志 (只读). 在LIN从机模式，当侦测到一个LIN报头错误时，该位由硬件置1，向该位写1清0。报头错误包括“break 间隔符太短（小于0.5位的时间）”， “在同步域或在识别域中帧错误，”“在非自动重同步模式同步域数据不是0x55”， “在自动重同步模式同步域偏离错误”， “同步域测量超时带自动重同步模式”和“LIN 标题接收超时’ 0 = LIN未检测到报头错误 1 = LIN检测到报头错误. 注意1: T该位只读，向该位写1清0 注意**2:** 该位仅在 LIN 从机模式有效(LINS_EN (UA_LIN_CTL [0]) = 1) 并且使能LIN 从机报头侦测功能 (LINS_HDET_EN (UA_LIN_CTL [1]))
[0]	**LINS_HDET_F**	**LIN 从机报头侦测标志 (只读)** 在LIN从机模式，当侦测到一个LIN报头，该位由硬件置位，通过软件写1清0 0 = LIN 报头没有被侦测到. 1 = LIN 报头被侦测到(break + 同步+ 帧ID) 注意1: 该位只读，向该位写1清0 注意2:该位仅在 LIN 从机模式有效(LINS_EN (UA_LIN_CTL [0]) = 1) 并且使能LIN 从机报头侦测功能 (LINS_HDET_EN (UA_LIN_CTL [1])) 注意**3:** 当使能ID 奇偶校验 (LIN_IDPEN (UA_LIN_CTL [9]))，如果硬件侦测到完整的报头(“break + sync + frame ID”)， LINS_HEDT_F 会被置位，无论帧ID是否正确

5.14 智能卡主机接口(SC)

5.14.1 概述

智能卡接口控制器(SC controlller)是基于ISO/IEC 7816-3标准并完全兼容PC/SC规格。它也提供卡插入/移除的状态，也支持串口模式的全双工异步通信。

5.14.2 特性

- 支持3个ISO-7816端口(SC0,SC1和SC2)
  - 兼容 ISO-7816-3 T=0,T=1
  - 兼容 EMV2000
  - 接收和发送各4字节缓存
  - 可编程的发送时钟频率
  - 可编程的接收器缓存触发水平
  - 可编程的保护时间选择(11 ETU ~ 267 ETU)
  - 一个24-位和两个8-位计数器用于请求应答(Answer to Request (ATR))和等待时间处理
  - 支持自动反向约定功能
  - 支持传送器和接收器错误重试和错误数目限制功能
  - 支持硬件激活系列，硬件暖复位系列和硬件释放系列处理
  - 支持当检测到卡移除时，硬件自动释放系列
- 支持3个串口(UART3，UART4，UART5)
  - 全双工异步通信
  - 可编程数据位长度，5-，6-，7-，8位字符
  - 独立的接收和发送各4字节缓存
  - 每个通道都支持可编程的波特率发生器
  - 支持可编程的接收器缓存触发水平
  - 可编程的发送数据延时时间(最后一个停止位从TX-FIFO离开到释放的时间)
  - 可编程的偶，奇或者无校验位的生成和检测
  - 可编程的停止位，1或者2停止位生成

5.14.3 框图

SC的时钟控制和框图如下所示。SC的控制器是和两时钟域(PCLK时钟和SC_CLK外围时钟)完全异步的设计，**注意**PCLK的时钟频率必须比高于外围时钟频率。

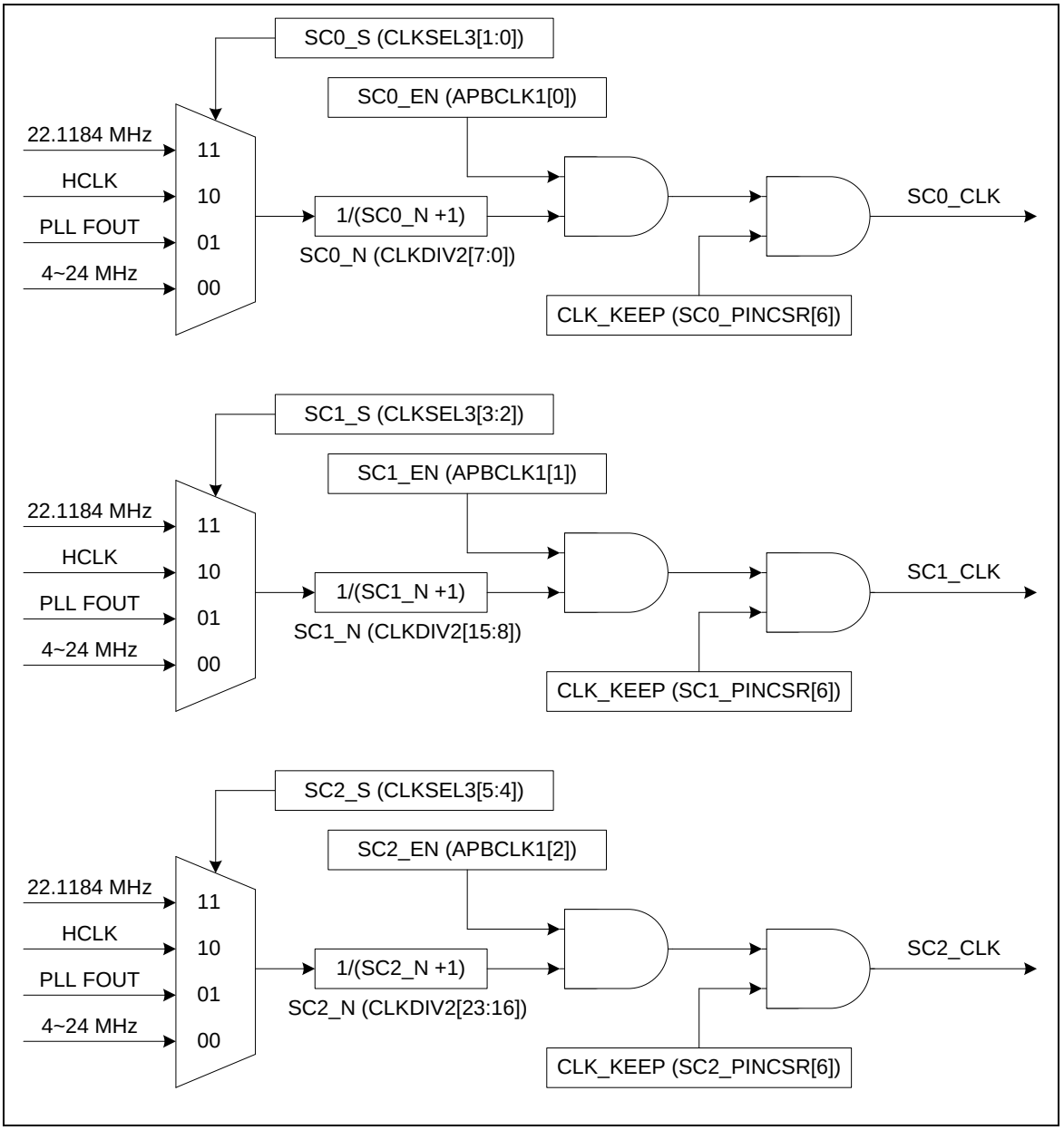


图 5-76 SC 时钟控制框图 (时钟控制器中的8-位预分频计数)

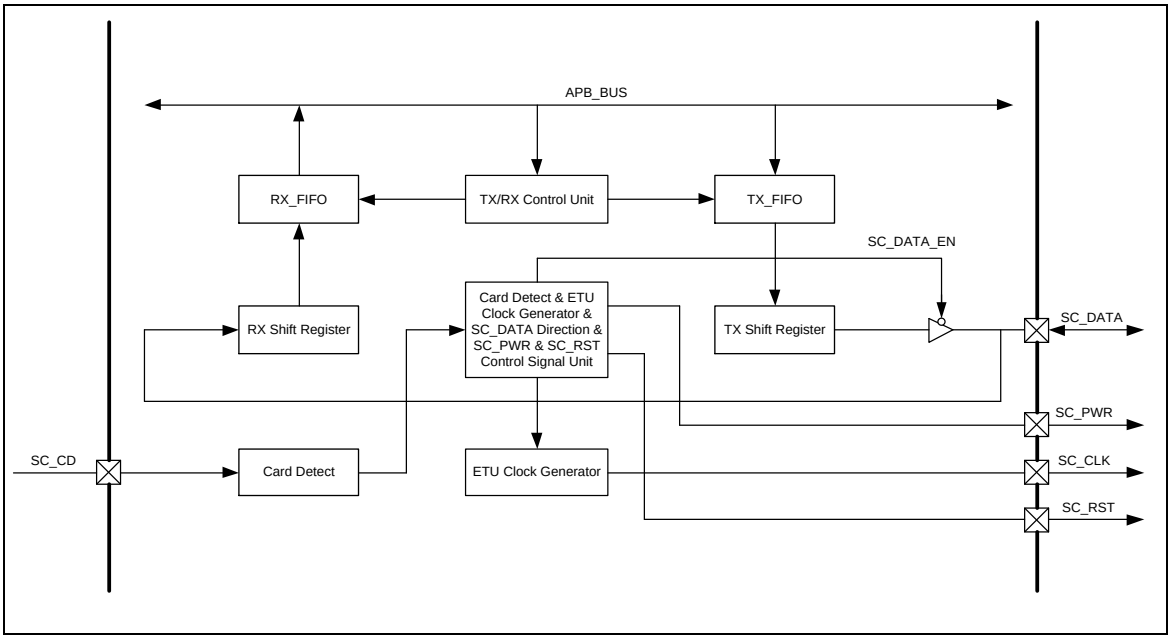


图 5-77 SC 控制器框图

5.14.4 基本配置

SC的功能引脚是有GPA_MFP，GPB_MFP，GPC_MFP的多功能引脚寄存器来配置的。

SC主控制器引脚的描述如下所示：

引脚	类型	描述
SC_DATA	双向的	SC 主控制器数据
SC_CD	输入	SC 主控制器卡检测
SC_PWR	输出	SC 主控制器卡电源开关
SC_CLK	输出	SC 主控制器时钟
SC_RST	输出	SC 主控制器复位

表 5-20 SC 主控制器引脚

串口引脚描述如下所示：

引脚	类型	描述
SC_DATA	输入	串口接收数据
SC_CLK	输出	串口发送数据

表 5-21 串口引脚

5.14.5 功能描述

基本上，智能卡接口扮演的是一个半双工异步通信端口的角色，它的数据格式如下图所示的10个连续位组成。

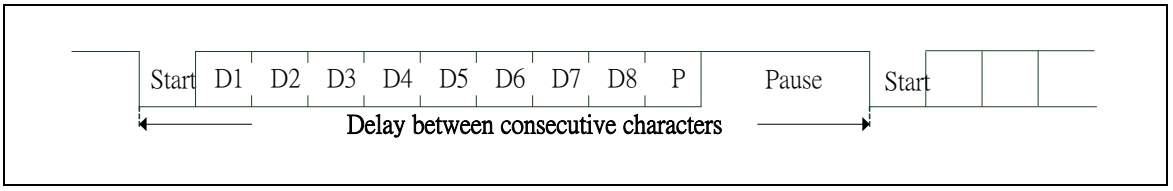


图 5-78 SC 数据字符

5.14.5.1 *激活，暖复位，释放系列.*

智能卡接口控制器支持硬件激活，暖复位，释放系列。激活，暖复位和释放序列如下所示：

激活

- 设置SC_RST为低
- 设置SC_PWR在高电平，SC_DAT在高电平(接收模式)
- 使能 SC_CLK 时钟
- 释放SC_RST到高电平

激活序列可以有两种方法控制。其过程如下所示：

软件控制时序：

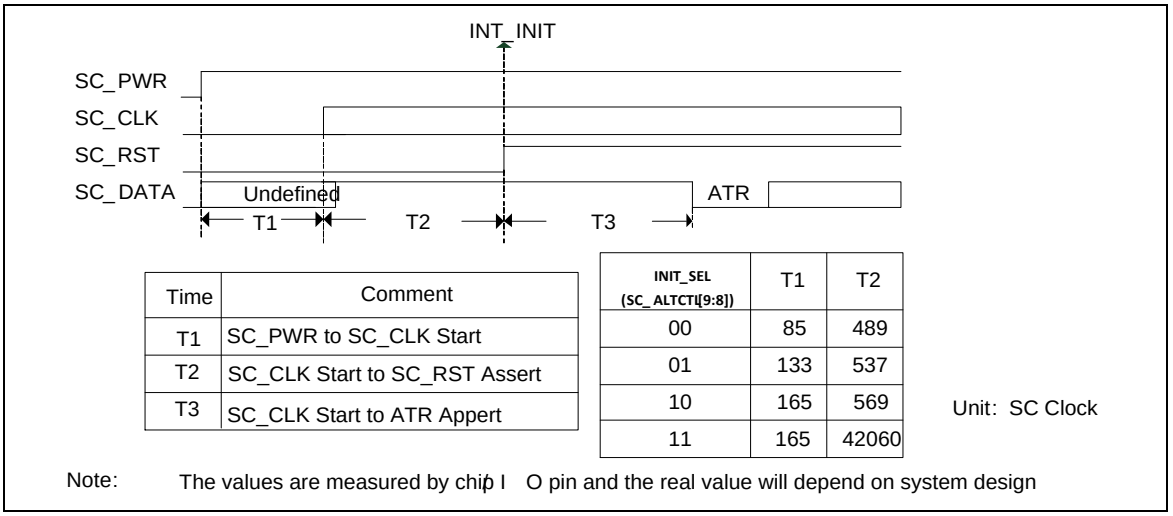
软件可以通过设置SC_PINCSR和SC_TMRx(x=0,1,2)来处理激活序列。SC_PINCSR可以设置SC_PWR,SC_CLK,SC_RST和SC_DATA引脚状态。设置SC_TMRx(x=0,1,2)可以控制激活序列的时序。这种编程过程给用户为激活序列提供了灵活的时序设置。

硬件时序控制：

软件设置 (寄存器SC_ALTCTL的第3位) ACT_EN为1，接口将通过硬件执行激活序列。SC_PWR上电到SC_CLK 开始时间(T1)和SC_CLK开始到SC_RST宣告的时间(T2)可以通过设置寄存器 (寄存器SC_ALTCTL的8,9位) INIT_SET来选择。这种编程过程给用户为激活序列提供了一种简单的设置。

如下是由硬件产生的激活控制序列：

- 通过设置 (寄存器SC_ALTCTL的8,9位) INIT_SEL来设置激活时序
- 通过设置 (寄存器SC_CTL的13,14位) TMR_SEL，可为'01'，'10'或者'11'，来选择TMR0。
- 设置操作模式 (寄存器SC_TMR0的24到27位) MODE为'0011'，通过设置(寄存器SC_TMR0的0到23位)的CNT来提供请求应答的值。
- 当硬件释放SC_RST到高时，硬件将产生一个中断INT_INIT到CPU，同时(寄存器SC_IER的第8位)INIT_IE置为1。
- 如果TMR0减小计数值到0(从SC_RST释放开始)，而且在此之前卡没有回应ATR，硬件将产生TMR0_IS(寄存器SC_ISR的第3位)中断到CPU。



Time	Comment
T1	SC_PWR to SC_CLK Start
T2	SC_CLK Start to SC_RST Assert
T3	SC_CLK Start to ATR Appert

INIT_SEL (SC_ALTCTL[9:8])	T1	T2
00	85	489
01	133	537
10	165	569
11	165	42060

Unit: SC Clock

Note: The values are measured by chip I O pin and the real value will depend on system design

图 5-79 SC 激活序列

暖复位

暖复位序列如下：

- 设置SC_RST为低而且设置SC_DAT为高
- 设置SC_RST为高

暖复位序列可以有两种方法来控制，其过程如下所示。

软件时序控制：

软件通过设置SC_PINCSR和SC_TMRx(x=0,1,2)来处理暖复位序列。可以通过编程SC_PINCSR来设置SC_RST和SC_DATA的引脚状态。也可以通过设置SC_TMRx(x=0,1,2)来控制暖复位序列的时序。这种编程过程给用户为暖复位序列提供了灵活的时序设置。

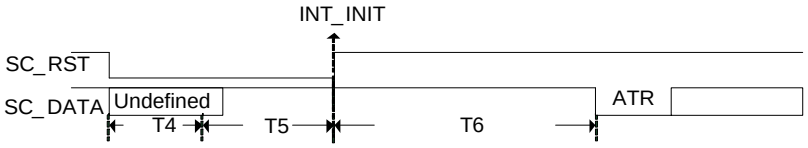
硬件时序控制：

软件通过设置 (寄存器SC_ALTCTL的第4位) WARST_EN为1，接口将通过硬件执行暖复位序列。可以通过编程 (寄存器SC_ALTCTL的8,9位) INIT_SET来选择SC_RST到SC_DATA接收模式的时间T4和SC_DATA接收模式到SC_RST宣告的时间T5。这种编程过程给用户为暖复位序列提供了一种简单的设置。

如下是由硬件产生的暖复位控制序列：

- 通过设置 (寄存器SC_ALTCTL的8,9位) INIT_SEL来设定暖复位的时序。
- 通过设置 (寄存器SC_CTL的13,14位) TMR_SEL，可以设为’01’，’10’或者’11’，来选择TMR0
- 设置操作模式 (寄存器SC_TMR0的 24到27位) MODE为’0011’，和通过设置(寄存器SC_TMR0[23:0])的CNT值来提供请求应答的值。
- 通过设置(寄存器SC_ALTCTL的第5位)TM0_SEN和(寄存器SC_ALTCTL的第4位)WARST来启动计数。
- 当硬件释放SC_RST为高时，硬件将同时产生INIT_IS(寄存器SC_ISR的第8位)中断给CPU(寄存器SC_IER的第8位INIT_IS需设为1)。
- 如果TMR0的计数值减小到0(从SC_RST开始)，而且在那之前卡没有回应ATR，硬件将

产生 (寄存器SC_ISR的第3位) TMR0_IS中断给CPU。



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

INIT_SEL (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

图 5-80 SC 热复位序列

释放

释放序列如下：

- 设置SC_RST为低
- 停止SC_CLK
- 设置SC_DATA状态为低
- 释放SC_PWR.

释放序列可以有两种方法来控制，其过程如下所示。

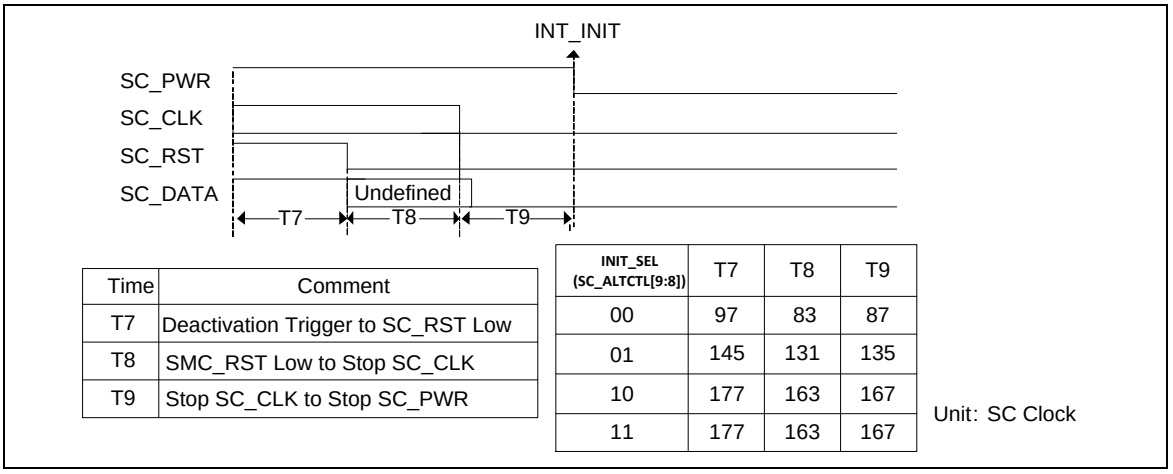
软件时序控制：

软件通过设置SC_PINCSR和SC_TMR0来处理释放序列。可以通过编程SC_PINCSR来设置SC_PWR,SC_CLK,SC_RST和SC_DATA的引脚状态。也可以通过设置SC_TMR0来控制释放序列的时序。这种编程过程给用户为释放序列提供了灵活的时序设置。

硬件时序控制：

软件通过设置 (寄存器SC_ALTCTL的第2位) DACT_EN为1，接口将通过硬件执行释放序列。可以通过编程 (寄存器SC_ALTCTL的8,9位) INIT_SET来选择释放触发到SC_RST变低的时间T7，SC_RST变低到SC_CLK停止的时间T8，和SC_CLK停止到SC_PWR停止的时间T9。这种程序过程给用户为释放序列提供了一种简单的设置。

当设置了卡移除检测(置位寄存器SC_ALTCTL的11位ADAC_CDEN)时，SC控制器也支持自动释放序列。



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

INIT_SEL (SC_ALTCTL[9:8])	T7	T8	T9
00	97	83	87
01	145	131	135
10	177	163	167
11	177	163	167

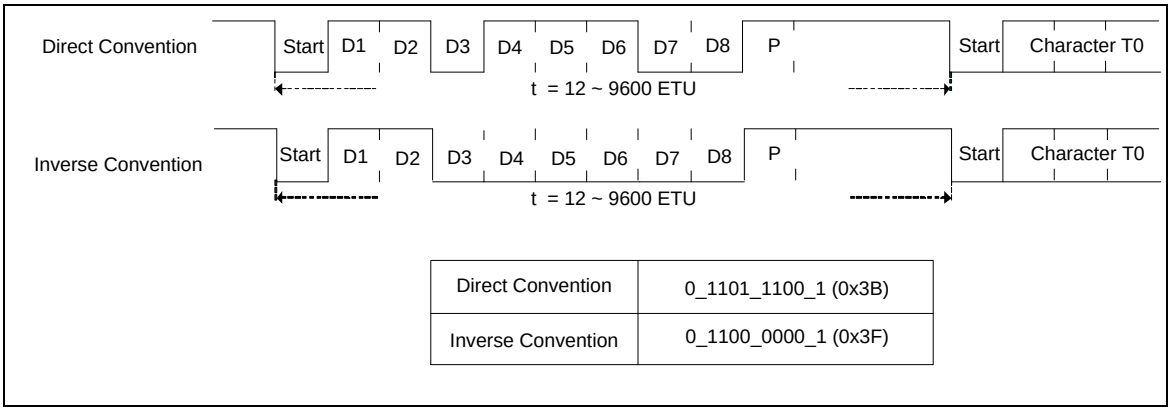
Unit: SC Clock

图 5-81 SC 释放序列

5.14.5.2 *初始化字符 TS*

依据7816-3，初始化字符TS有两种可能的模式(如下图所示)。如果TS模式是1100_0000，则是反向约定。当反向约定进行解码后，则传送的字节等于0x3F。如果TS模式是1101_1100，则是直接约定。当按直接约定解码后，则传送的字节等于0x3B。软件可以设置 (寄存器SC_CTL的第3位) AUTO_CON_EN，由硬件来决定操作的约定。软件也可以设置 (寄存器SC_CTL的4,5位) CON_SEL为'00'或者'11'，在收到ATR的TS后去改变操作的约定。

如果软件通过设置 (寄存器SC_CTL的第3位) AUTO_CON_EN使能自动约定功能，则设置步骤必须在ATR状态之前完成，而且第一个数据必须是0x3B 或0x3F。在硬件收到第一个数据并放到缓存中之后，硬件将决定约定模式并自动改变 (寄存器SC_CTL的4,5位) CON_SEL。如果第一个数据既不是0x3B也不是0x3F，则硬件将产生一个INT_ACON_ERR中断给CPU(如果寄存器SC_IER的ACON_ERR_IE位设置为1)。



Direct Convention	0_1101_1100_1 (0x3B)
Inverse Convention	0_1100_0000_1 (0x3F)

图 5-82 初始化字符TS

5.14.5.3 *错误信号和字符重复*

依据ISO7816-3 T=0模式描述，如下所示，如果接收器接收到一个错误的校验位，接收器将会拉低SC_DATA 1.5个位周期去通知发送器校验错误。然后发送器将重传该字符。SC接口控制器支持接收器硬件错误检测功能和发送器硬件重传功能。软件可以通过设置 (寄存器SC_CTL的23位) TX_ERETRY_EN来使能重传功能。软件也可以在 (寄存器SC_CTL的20到22位) TX_ERETRY中定义重传的次数限制。重传次数可以达TX_ERETRY+1,当重传次数等于TX_ERETRY+1时，硬件将会置TX_OVER_REERR标志，而且如果 (寄存器SC_IER的第2位) TERR_IE使能，SC接口控制器将会产生一个传输错误中断给CPU。软件也可以在(寄存器SC_CTL的16到18位)RX_ERETRY定义接收重传次数的限制。接收重传次数可达RX_ERETRY+1，如果接收错误的次数等于RX_ERETRY+1,接收器将接收该错误的数据到缓存并且硬件将会置位RX_OVER_REERR，如果(寄存器SC_IER的第2位)TERR_IE使能，SC控制将产生一个传输错误中断给CPU。

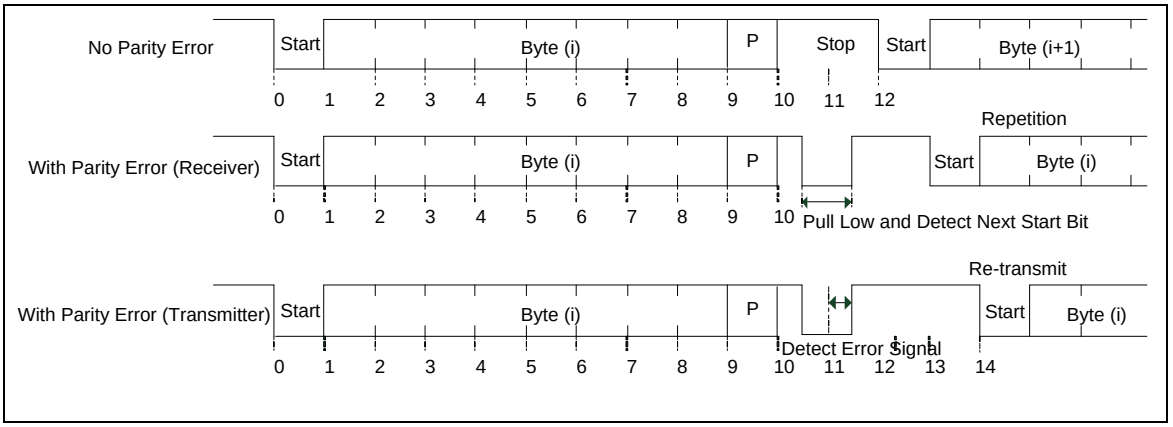


图 5-83 SC 错误信号

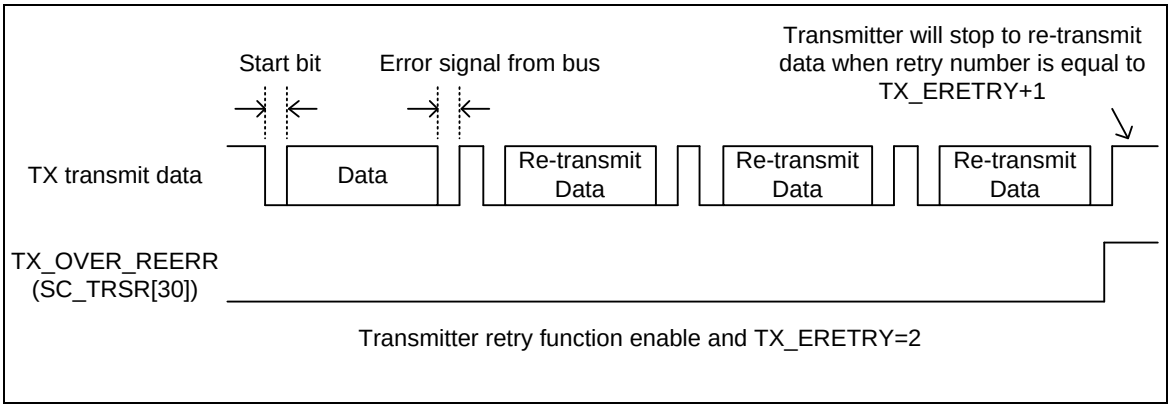


图 5-84 SC 重发送次数和重试过载标志

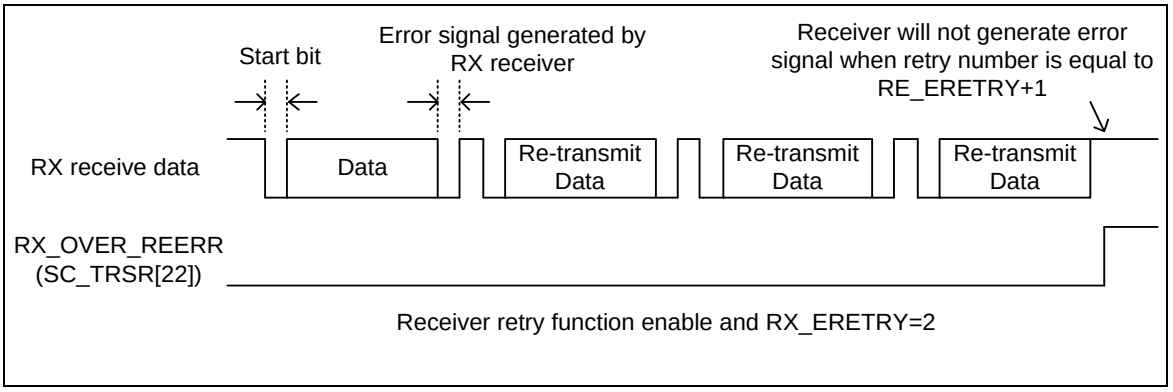


图 5-85 SC 重接收次数和重试过载标志

5.14.5.4 *内部超时计数*

智能卡接口包括一个24位超时计数器和两个8位超时计数器。这些计数器帮助处理不同的实时间隔。每个计数器可以被设置成一旦触发使能位被写或者检测到一个START位就开始进行计数。

如下是编程流程：

通过设置(寄存器SC_CTL的13,14位)TMR_SEL来使能计数。通过设置(寄存器SC_TMRx的24到27位)MODE选择操作模式和该(寄存器的0到23位)CNT来提供一个计数值。通过设置(寄存器SC_ALTCTL的第5位TMR0_SEN，第6位TMR1_SEN或者第7位TMR2_SEN)来开始进行计数。

SC_TMR0，SC_TMR1和SC_TMR2计数器的操作模式如下表：

注意：只有SC_TMR0支持模式0011.

TMRx _SEL (X=0 ~2)	**操作描述**	
0000	当(寄存器SC_ALTCTL的5到7位)TMRx_SEN使能时，向下计数器开始计数，当计数器超时时停止计数。超时值是(寄存器SC_TMR0的0到23位, SC_TMR1的0到7位,SC_TMR2的0到7位)CNT+1的值。	
	开始	当(寄存器SC_ALTCTL的第5到7位)TMRx_SEN使能时开始计数。
	停止	当向下计数器的值等于0时，硬件将置位(寄存器SC_ISR的第3到5位)TMRx_IS并且自动清除(寄存器SC_ALTCTL的第5到7位)TMRx_SEN。
0001	当第一个开始位(接收或者发送)检测到时向下计时器开始计数，当计数器超时时停止计数。超时值是(寄存器SC_TMR0的0到23位, SC_TMR1的0到7位,SC_TMR2的0到7位)CNT+1的值。	
	开始	在(寄存器SC_ALTCTL的第5到7位)TMRx_SEN设置为1后，当第一个开始位(接收或者传送)检测到时计时器开始计数
	停止	当向下计数器的值等于0时，硬件将置位(寄存器SC_ISR的第3到5位)TMRx_IS并且自动清除(寄存器SC_ALTCTL的第5到7位)TMRx_SEN。
0010	当第一个开始位(接收)检测到时，向下计数器开始计数，当计数器超时时停止计数。超时值是(寄存器SC_TMR0的0到23位, SC_TMR1的0到7位,SC_TMR2的0到7位)CNT+1的值。	
	开始	在(寄存器SC_ALTCTL的第5到7位)TMRx_SEN设置为1后，当第一个开始位(接收)检测到时计时器开始计数
	停止	当向下计数器的值等于0时，硬件将置位(寄存器SC_ISR的第3到5位)TMRx_IS并且自动清除(寄存

		器SC_ALTCTL的第5到7位)TMRx_SEN。
0011	向下计数器只用于硬件激活，暖复位序列来测量ATR时序。时序开始于SC_RST释放，结束于ATR应答收到或超时。如果在ATR应答收到之前，计数器减到到0，则硬件将产生一个中断给CPU。超时值为(寄存器SC_TMR0的0到23位) CNT+1。	
	开始	在(寄存器SC_ALTCTL的第5位)TMR0_SEN设为1后，当SC_RST释放时，计数器开始计数。可用于硬件激活和暖复位模式。
	停止	在ATR应答收到之前，当向下计数器的值等于0时，硬件将自动置位TMR0_IS并清(寄存器SC_ALTCTL的第5位)TMR0_SEN。 当收到ATR且向下计数器计数不等于0时，硬件将自动清除(寄存器SC_ALTCTL的第5位)TMR0_SEN。
0100	和模式0000一样，但是当向下计数器等于0时，硬件将置位(寄存器SC_ISR的3到5位)TMRx_IS而且计数器将重载(寄存器SC_TMR0的0到23位，寄存器SC_TMR1的0到7位，寄存器SC_TMR2的0到7位) CNT的值，并重新计数直到软件清除 (寄存器SC_ALTCTL的5到7位) TMRx_SEN。 当（寄存器SC_ALTCTL的13到15位）TMRx_ATV=1，软件能在任何时候改变 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的值。当向下计数器计数到等于0时，计数器将重载（寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位）CNT的新值并重新计数。 超时值为 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT+1。	
0101	和模式0001一样，但是当向下计数器计数值等于0时，硬件将置位 (寄存器SC_ISR的3到5位) TMRx_IS而且计数器将重载 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的值。当检测到下一个START位时，计数器将重新计数直到软件清除 (寄存器SC_ALTCTL的5到7位) TMRx_SEN。 当寄存器SC_ALTCTL的13到15位) TMRx_ATV(=1时，软件能在任何时候改变 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的值。当向下计数器等于0时，计数器将重载 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的新值并重新计数。 超时值为 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT+1。	
0110	和模式0010一样，但是当向下计数器等于0时，硬件将置位 (寄存器SC_ISR的3到5位) TMRx_IS而且计数器将重载 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的值。当检测到下一个START位时，计数器将重新计数直到软件清除 (寄存器SC_ALTCTL的5到7位) TMRx_SEN。 当 (寄存器SC_ALTCTL的13到15位) TMRx_ATV=1时，软件能在任何时候改变 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的值。当向下计数器等于0时，计数器将重载 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的新值并重新计数。 超时值为 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT+1。	
0111	当第一个START位(接收和发送)检测到时，向下计数器开始计数。当软件清除 (寄存器SC_ALTCTL的5到7位) TMRx_SEN位时，计数器停止。如果下一个START位被检测到，计数器将重载 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的新值并重新计数 如果在下一个START位被检测到之前，计数器减到0，硬件将产生一个中断给CPU。 超时值为 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT+1。	
	开始	在(寄存器SC_ALTCTL的5到7位)TMR0_SEN设为1后，当第一个START位检测到时，计数器开始计数。
	停止	在(寄存器SC_ALTCTL的5到7位)TMR0_SEN设为0后，计数器停止计数。
1000	当(寄存器SC_ALTCTL的5到7位) TMRx_SEN使能时，向上计数器开始计数。当 (寄存器SC_ALTCTL的5到7位) TMRx_SEN禁止时，停止计数。计数值将被保存在 (寄存器SC_TDRA的0到23位，SC_TDRB的0到7位，SC_TDRB的8到15位) TDRx+1。在该模式下，硬件不能产生任何中断给CPU，实际计数值将是 (寄存器SC_TDRA的0到23位，SC_TDRB的0到7位，SC_TDRB的8到15位) TDRx+1。	
	开始	在 (寄存器SC_ALTCTL的5到7位) TMRx_SEN设为1后，开始计数，开始计数的值为0(硬件将忽略 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的值).。
	停止	在 (寄存器SC_ALTCTL的5到7位) TMRx_SEN设为0后，停止计数，计数值保存到 (寄存器SC_TDRA的0到23位，SC_TDRB的0到7位，SC_TDRB的8到15位) TDRx。
1111	当软件置位 (寄存器SC_ALTCTL的5到7位) TMRx_SEN或者任何START位检测到时，向下计数器开始计数，当	

	软件清 (寄存器SC_ALTCTL的5到7位) TMRx_SEN位时计数器停止计数。如果下一START位检测到，计数器将重载 (寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT的新值并重新计数。 如果在下一个START位被检测到之前，计数器减到0，硬件将产生一个中断给CPU。 超时值为(寄存器SC_TMR0的0到23位，SC_TMR1的0到7位，SC_TMR2的0到7位) CNT+1。	
	开始	当(寄存器SC_ALTCTL的5到7位)TMRx_SEN置为1时，计数器开始计数。
	停止	在(寄存器SC_ALTCTL的5到7位)TMR0_SEN设为0后，计数器停止计数。

表 5-22 Timer2/Timer1/Timer0 操作模式

5.14.5.5 *块保护时间和扩展保护时间*

块保护时间是在不同传输方向之间的两个连续字符的第一位边缘之间的最短位长度。该域表示用于块保护时间的计数器。依据ISO7816-3，在T=0的模式下，软件必须填充该域为15(实际的块保护时间=16.5)，在T=1的模式下，软件必须填充该域位21(实际的块保护时间位22.5)。

在发送方向，智能卡先发送数据到智能卡主控制器。在一段时间超过(寄存器SC_CTL的8到12位)BGT后，智能卡主控制器开始发送数据。

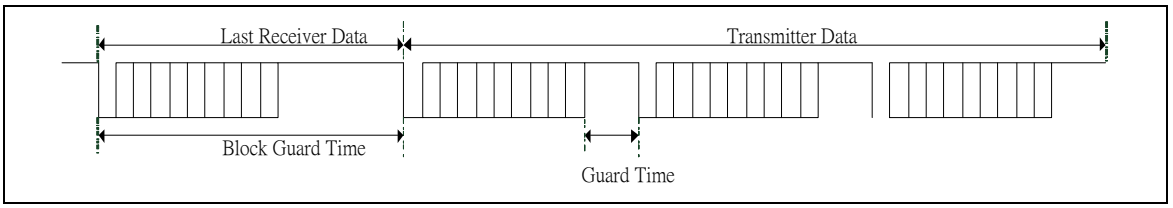


图 5-86 发送方向块保护时间操作

在接收方向，智能卡主控制器先发送数据给智能卡。如果智能卡发送数据给智能卡主控制器的时间少于(寄存器SC_CTL的8到12位)BGT，当(寄存器SC_ALTCTL的12位)RX_BGT_EN使能时，块保护时间中断(寄存器SC_ISR的第6位)BGT_IS发生。

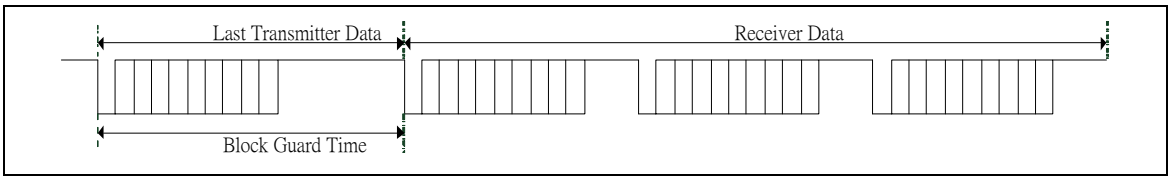


图 5-87 接收方向块保护时间操作

扩展保护时间是两个ETU加(寄存器SC_EGTR的0到位)EGT，其格式如下所示：

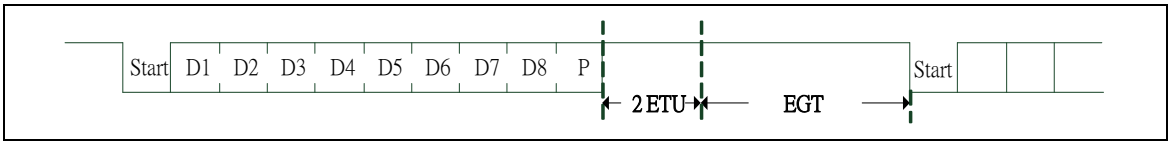


图 5-88 扩展保护时间操作

5.14.5.6 *串口模式*

当(寄存器SC_UACTL的第0位)UA_MODE_EN被置位，智能卡控制器也可以作为一个基本的串口功能来使用。下面是一个串口模式的编程例子。

编程例子：

1. 软件可以通过设定(寄存器SC_UACTL的第0位)UA_MODE_EN位来进入串口模式。
2. 通过设定(寄存器SC_ALTCTL的第1位)RX_RST和(寄存器SC_ALTCTL的第0位)TX_RST位进行软件复位，以确保所有的机器状态回到空闲状态。
3. 填充0到(寄存器SC_CTL的4到5位)CON_SEL和(寄存器SC_CTL的第3位)AUTO_CON_EN域。(当工作在串口模式时，这些域必须为0)
4. 通过设置(寄存器SC_ETUCR的0到11位)ETU_RDIV域来选择串口波特率。例如，如果智能卡模块的时钟为12MHZ，而且目标波特率位115200，则ETU_RDIV应设填为(12000000/115200-1)
5. 选择数据格式，包括数据长度(通过设置寄存器SC_UACTL的4到5位DATA_LEN)，校验位格式(通过设定寄存器SC_UACTL的第7位OPE和寄存器SC_UA_CTL的第6位PBDIS)和停止位长度(通过设定寄存器SC_CTL的15位SLEN或者寄存器SC_EGTR的0到7位EGT)。
6. 通过设定(寄存器SC_CTL的6到7位)RX_FTRI_LEV域来选择接受缓存触发水平，并通过设定(寄存器SC_RFTMR的0到8位)RFTMR域来选择接收缓存超时间隔。

写(寄存器SC_THR的0到7位)SC_THR或者读(寄存器SC_RBR的0到7位)SC_RBR可以执行串口功能。

5.14.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
SC 基地址: **SC0_BA = 0x4019_0000** **SC1_BA = 0x4019_4000** **SC2_BA = 0x4019_8000**				
SC_RBR **x=0,1,2**	SCx_BA+0x00	R	SC接收缓存寄存器(只读)	未定义
SC_THR **x=0,1,2**	SCx_BA+0x00	W	SC 发送保持寄存器	未定义
SC_CTL **x=0,1,2**	SCx_BA+0x04	R/W	SC 控制寄存器	0x0000_0000
SC_ALTCTL **x=0,1,2**	SCx_BA+0x08	R/W	SC 交替控制寄存器	0x0000_0000
SC_EGTR **x=0,1,2**	SCx_BA+0x0C	R/W	SC 扩展保护时间寄存器	0x0000_0000
SC_RFTMR **x=0,1,2**	SCx_BA+0x10	R/W	SC 接收缓存超时寄存器	0x0000_0000
SC_ETUCR **x=0,1,2**	SCx_BA+0x14	R/W	SC ETU 控制寄存器	0x0000_0173
SC_IER **x=0,1,2**	SCx_BA+0x18	R/W	SC 中断使能控制寄存器	0x0000_0000
SC_ISR **x=0,1,2**	SCx_BA+0x1C	R/W	SC 中断状态寄存器	0x0000_0002
SC_TRSR **x=0,1,2**	SCx_BA+0x20	R/W	SC 状态寄存器	0x0000_0202
SC_PINCSR **x=0,1,2**	SCx_BA+0x24	R/W	SC 管脚控制状态寄存器	0x0000_00x0
SC_TMR0 **x=0,1,2**	SCx_BA+0x28	R/W	SC 内部定时器控制寄存器0	0x0000_0000
SC_TMR1 **x=0,1,2**	SCx_BA+0x2C	R/W	SC 内部定时器控制寄存器1	0x0000_0000
SC_TMR2 **x=0,1,2**	SCx_BA+0x30	R/W	SC 内部定时器控制寄存器2	0x0000_0000
SC_UACTL **x=0,1,2**	SCx_BA + 0x34	R/W	SC 串口模式控制寄存器	0x0000_0000
SC_TDRA **x=0,1,2**	SCx_BA+0x38	R	SC 定时器当前数据寄存器A	0x0000_07FF
SC_TDRB **x=0,1,2**	SCx_BA+0x3C	R	SC 定时器当前数据寄存器B	0x0000_7F7F

注意: SCx_REG中的x代表 SC是通道。

5.14.7 寄存器描述

SC 接收缓存寄存器(SC_RBR)

寄存器	偏移地址	R/W	描述	复位值
SC_RBR	SCx_BA+0x00	R	SC 接收缓存寄存器	未定义

31	30	29	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
RBR							

位	描述	
[31:8]	Reserved	保留.
[7:0]	RBR	**接收缓存寄存器** 通过读该寄存器，SC将返回一个8-位的接收到的数据

SC 发送保持寄存器 (SC_THR)

寄存器	偏移地址	R/W	描述	复位值
SC_THR	SCx_BA+0x00	W	SC 发送保持寄存器	未定义

31	30	29	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
THR							

位	描述	
[31:8]	Reserved	保留
[7:0]	THR	**发送保持寄存器** 通过写该寄存器，SC将发送出一个8-位数据. **注意:** 如果(寄存器SC_CTL第0位) SC_CEN 没有使能，该寄存器将不能被编程.

SC 控制寄存器(SC_CTL)

寄存器	偏移地址	R/W	描述	复位值
SC_CTL	SCx_BA+0x04	R/W	SC 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	SYNC	Reserved				CD_DEB_SEL	
23	22	21	20	19	18	17	16
TX_ERETRY_EN	TX_ERETRY			RX_ERETRY_EN	RX_ERETRY		
15	14	13	12	11	10	9	8
SLEN	TMR_SEL		BGT				
7	6	5	4	3	2	1	0
RX_FTRI_LEV		CON_SEL		AUTO_CON_EN	DIS_TX	DIS_RX	SC_CEN

位	描述	
[31]	**Reserved**	保留
[30]	**SYNC**	**SYNC 标志指示器** 用于同步，软件在写入一个新值给RX_ERETRY和TX_ERETRY之前必须检查该位 0 = 同步已经完成，用户可以写入新数据给RX_ERETRY和TX_ERETRY 1 = 最后值在同步中. **注意:** 该位只读.
[30:26]	**Reserved**	保留
[25:24]	**CD_DEB_SEL**	**卡检测去抖选择** 该域指示卡检测去抖选择 00 = 每384(128*3)个SC 外部时钟去抖采样卡插入一次，每128个SC外部时钟去抖采样卡移除一次 01 = 每192(64*3)个SC 外部时钟去抖采样卡插入一次，每64个SC外部时钟去抖采样卡移除一次 10 = 每96(32*3)个SC 外部时钟去抖采样卡插入一次，每32个SC外部时钟去抖采样卡移除一次 11 = 每48(16*3)个SC 外部时钟去抖采样卡插入一次，每16个SC外部时钟去抖采样卡移除一次
[23]	**TX_ERETRY_EN**	**TX 错误重传使能** 当校验错误发生时，该位使能发送器重传功能 0 = 禁用TX 错误重传功能. 1 = 使能TX 错误重传功能
[22:20]	**TX_ERETRY**	**TX 错误重传次数** 该域表明当校验错误发送后，发送器允许尝试的最多重传次数。 注意1：实际的重传次数是TX_ERETRY+1，所以8次是最多的重传次数。

		注意2：当TX_ERETRY_EN使能时，该域不能被修改。修改该域的流程是先禁用TX_ERETRY_EN，然后填充新的重传值。
[19]	**RX_ERETRY_EN**	**RX 错误重接收使能位** 当校验位错误发送时，该位使能接收器重接收功能 0 = 禁止RX错误重接收功能 1 = 使能RX错误重接收功能 注意：在使能该位之前，软件必须填充RX_ERETRY
[18:16]	**RX_ERETRY**	**RX 错误重接收计数次数** 该域表示当校验错误发生时，接收器允许重接收的最多次数。. 注意1：实际的重接收次数是RX_ERETRY+1，所以8次是最多的重接收次数。 注意2：当RX_ERETRY_EN使能时，该域不能被修改。修改该域的流程是先禁用RX_ERETRY_EN，然后填充新的重接收值。
[15]	**SLEN**	**停止位长度** 该域表示停止位的长度t. 0 = 停止位的长度是2 ETU. 1 = 停止位的长度是1 ETU. **注意:** 默认停止位的长度是2。SMC和UART采用SLEN来编程停止位的长度。
[14:13]	**TMR_SEL**	**定时器选择** 00 = 禁用所有内部定时器功能 01 = 使能内部24位定时器。软件能够通过设置寄存器SC_TMR0的0到23位配置定时器，SC_TMR1和SC_TMR2在该模式下降被忽略。 10 = 使能内部24位定时器和8位内部定时器。软件可以通过设置寄存器SC_TMR0的0到23位来配置24位定时器和通过设置寄存器SC_TMR1的0到7位来配置8位定时器。SC_TMR2该模式下将被忽略。 11 = 使能内部24位定时器和两个8位定时器。软件可以通过设置寄存器SC_TRM0的0到23位，寄存器SC_TMR1的0到7位和寄存器SC_TMR2的0到7位来配置定时器。
[12:8]	**BGT**	**块保护时间 (BGT)** 块保护时间是在不同传输方向之间的两个连续字符的第一位边缘之间的最短位长度。该域表示用于块保护时间的计数器。依据ISO7816-3，在T=0的模式下，软件必须填充该域为15(实际的块保护时间=16.5)，在T=1的模式下，软件必须填充该域位21(实际的块保护时间位22.5) **注意:** 实际块保护时间是BGT + 1.
[7:6]	**RX_FTRI_LEV**	**RX 缓存触发水平** 当接收缓存的字节数目等于RX_FTRI_LEV时，RDA_IF将被置位(如果使能了寄存器IER的RDA_IEN位，将产生中断)。 00 = INTR_RDA 的触发水平 1 字节. 01 = INTR_RDA 的触发水平 2 字节. 10 = INTR_RDA 的触发水平 3 字节. 11 = 保留.
[5:4]	**CON_SEL**	**约定选择** 00 = 直接约定. 01 = 保留. 10 = 保留. 11 = 反向约定. **注意:**如果寄存器SC_CTL的第3位 AUTO_CON_EN使能， 该域将被忽略。
[3]	**AUTO_CON_EN**	**自动约定使能位**

		0 = 禁止自动约定. 1 = 使能自动预定。当硬件在ATR状态收到TS，且TS是直接约定，寄存器SC_CTL的4到5位CON_SEL将被自动设位0；否则如果TS是反向约定，CON_SEL将被设位11。 如果软件使能自动约定功能，设置步骤必须在ATR状态之前完成，而且第一个数据必须是0x3B或者0x3F。在硬件收到第一个数据并保存到缓存后，硬件将决定何种约定并自动改变寄存器SC_CTL的CONSEL的值。如果第一个数据既不是0x3B也不是0x3F，则硬件将产生INT_ACON_ERR中断到CPU(如果寄存器SC_IER的第10位ACON_ERR_IE =1)。
[2]	**DIS_TX**	**禁止TX 转换位** 0 = 发送器使能. 1 = 发送器禁用.
[1]	**DIS_RX**	**禁止RX 转换位** 0 = 接收器使能. 1 = 接收器禁止. **注意:** 如寄存器SC_CTL的第3位 AUTO_CON_EN使能，这些域必须忽略。
[0]	**SC_CEN**	**SC 引擎使能位** 设置该位为1使能SC操作。如果该位被清除，SC将强制所有转换到IDLE状态。

SC 交替控制寄存器(SC_ALTCTL)

寄存器	偏移地址	R/W	描述	复位值
SC_ALTCTL	SCx_BA+0x08	R/W	SC交替控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							OUTSEL
15	14	13	12	11	10	9	8
TMR2_ATV	TMR1_ATV	TMR0_ATV	RX_BGT_EN	Reserved		INIT_SEL	
7	6	5	4	3	2	1	0
TMR2_SEN	TMR1_SEN	TMR0_SEN	WARST_EN	ACT_EN	DACT_EN	RX_RST	TX_RST

位	描述	
[31:17]	**Reserved**	保留
[16]	**OUTSEL**	**智能卡数据引脚输出模式选择** 使用该位来选择智能卡数据引脚(SC_DATA)输出模式) 0 = 准双向模式. 1 = 开漏模式.
[15]	**TMR2_ATV**	**内部定时器2 激活状态(只读)**. 该位指示定时器2的定时器计数器状态 0 = 定时器2未激活. 1 = 定时器2激活.
[14]	**TMR1_ATV**	**内部定时器1 激活状态(只读)**. 该位指示定时器1的定时器计数器状态 0 = 定时器1未激活. 1 = 定时器1激活.
[13]	**TMR0_ATV**	**内部定时器0 激活状态(只读)**. 该位指示定时器0的定时器计数器状态 0 = 定时器0未激活. 1 = 定时器0激活.
[12]	**RX_BGT_EN**	**接收器块保护时间功能使能位** 0 = 禁止接收器块保护时间功能 1 = 使能接收器块保护时间功能.
[11:10]	**Reserved**	保留.
[9:8]	**INIT_SEL**	**初始时序选择** 该域表示硬件初始状态的时序(激活或暖复位或是释放). 单位: SC 时钟

		激活: 查阅SC激活序列的图 5-79. 暖复位: 查阅暖复位序列的图 5-80. 释放: 查阅释放序列的图 5-81.
[7]	**TMR2_SEN**	**内部定时器2开始使能位** 该位使能定时器2开始计数。软件能够填充0来停止计数器，设置1重载和计数。 0 = 停止计数 1 = 开始计数. **注意1:** 当寄存器SC_CTL的13到14位TMR_SEL=11时，该域用于内部8位定时器。当寄存器SC_CTL的13到14位TMR_SEL= 00或01或10时，不要填充TMR2_SEN。 **注意2:** 如果操作模式不是自动重载模式(寄存器SC_TMR2的第26位=0)，则该位将被硬件自动清除。 **注意3:** 该域会被寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST清除。所以不要同时设置该位寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST。 **注意4:** 如果寄存器 SC_CTL的第0位SC_CEN未使能，不能对该位进行编程。
[6]	**TMR1_SEN**	**内部定时器1开始使能位** 该位使能定时器1开始计数。软件能够填充0来停止计数器，设置1重载和计数。 0 = 停止计数 1 = 开始计数. **注意1:** 当寄存器SC_CTL的13到14位TMR_SEL=10或11时，该域用于内部8位定时器。当寄存器SC_CTL的13到14位TMR_SEL= 00或01时，不要填充TMR2_SEN。 **注意2:** 如果操作模式不是自动重载模式(寄存器SC_TMR1的第26位=0)，则该位将被硬件自动清除。 **注意3:** 该域会被寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST清除。所以不要同时设置该位寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST。 **注意4:** 如果寄存器 SC_CTL的第0位SC_CEN未使能，不能对该位进行编程。
[5]	**TMR0_SEN**	**内部定时器0开始使能位** 该位使能定时器2开始计数。软件能够填充0来停止计数器，设置1重载和计数。 0 = 停止计数 1 = 开始计数. **注意1:** 当寄存器SC_CTL的13到14位TMR_SEL=01时，该域用于内部24位定时器。 **注意2:** 如果操作模式不是自动重载模式(寄存器SC_TMR0的第26位=0)，则该位将被硬件自动清除。 **注意3:** 该域会被寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST清除。所以不要同时设置该位寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST。 **注意4:** 如果寄存器 SC_CTL的第0位SC_CEN未使能，不能对该位进行编程。
[4]	**WARST_EN**	**暖复位序列发生器使能位** 该位使能SC控制器通过暖复位序列初始化卡 0 = 不起作用. 1 = 使能暖复位序列发生器. **注意1:** 当暖复位序列完成后，该位将被自动清除，寄存器SC_ISR的第8位INIT_IS将被设置为1。 **注意2:**该域将被寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST清除，所以不要同时填充该位(寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST) **注意3:** 如果寄存器SC_CTL的第0位SC_CEN未使能，该域将不能被编程。
[3]	**ACT_EN**	**激活序列发生器使能位** 该位使能SC控制器通过激活序列初始化卡

		0 = 不起作用 1 = 使能激活序列发生器 **注意1:** 当激活序列完成后，该位将被自动清除，寄存器SC_ISR的第8位INIT_IS将被设置为1。 **注意2:**该域将被寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST清除，所以不要同时填充该位(寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST) **注意3:** 如果寄存器SC_CTL的第0位SC_CEN未使能，该域将不能被编程。
[2]	**DACT_EN**	**释放序列发生器使能位** 该位使能SC控制器通过释放序列初始化卡 0 = 不起作用 1 = 使能释放序列发生器 **注意1:** 当释放序列完成后，该位将被自动清除，寄存器SC_ISR的第8位INIT_IS将被设置为1。 **注意2:**该域将被寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST清除，所以不要同时填充该位(寄存器SC_ALTCTL的第0位TX_RST和第1位RX_RST) **注意3:** 如果寄存器SC_CTL的第0位SC_CEN未使能，该域将不能被编程。
[1]	**RX_RST**	**RX 软件复位** 当RX_RST被置位，接收缓存中的所有数据和RX内部状态机将被清除。 0 = 不起作用 1 = 复位RX内部状态机和指针. **注意:** 在复位完成后，该位将被自动清除
[0]	**TX_RST**	**TX 软件复位** 当TX_RST被置位，发送缓存中的所有数据和TX内部状态机将被清除。 0 = 不起作用 1 = 复位TX内部状态机和指针. **注意:** 在复位完成后，该位将被自动清除

SC 扩展保护时间寄存器 (SC_EGTR)

寄存器	偏移地址	R/W	描述	复位值
SC_EGTR	SCx_BA+0x0C	R/W	SC 扩展保护时间寄存器	0x0000_0000

31	30	29	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							

位	描述	
[31:8]	**Reserved**	保留
[7:0]	**EGT**	**扩展保护时间** 该域表示扩展保护时间的值 **注意:** 该计数器是基于ETU，实际的扩展保护时间是EGT

SC接收器缓存超时寄存器 (SC_RFTMR)

寄存器	偏移地址	R/W	描述	复位值
SC_RFTMR	SCx_BA+0x10	R/W	SC接收器缓存超时寄存器	0x0000_0000

31	30	29	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							

位	描述	
[31:9]	Reserved	保留.
[8:0]	RFTM	**SC 接收器缓存超时(基于ETU)** 无论什么时候RX缓存接收到一个新的数据字，超时计数器复位并开始计数。一旦计数器减少到1，而且没有接收到新的数据或者CPU读SC_RBR寄存器没有可读数据，则将产生一个接收器超时中断INT_RTMR(如果寄存器SC_IER的第9位RTMR_IE=1) **注意1:** 计数器单元是基于ETU，超时间隔的值是RFTM+0.5。 **注意2:**填充0到该域来禁止该功能。

SC ETU控制寄存器(SC_ETUCR)

寄存器	偏移地址	R/W	描述	复位值
SC_ETUCR	SCx_BA+0x14	R/W	SC ETU控制寄存器	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
COMPEN_EN	Reserved			ETU_RDIV			
7	6	5	4	3	2	1	0
ETU_RDIV							

位	描述	
[31:16]	Reserved	保留
[15]	COMPEN_EN	**补偿模式使能位** 该位使能时钟补偿功能。当该位使能，硬件将在n时钟周期和n-1时钟周期之间交替，其中n为写入到ETU_RDIV寄存器的值。 0 = 禁用补偿功能 1 = 使能补偿功能
[14:12]	Reserved	保留
[11:0]	ETU_RDIV	**ETU 速率除法器** 该域表示时钟速率除法器. 实际的ETU为ETU_RDIV + 1. **注意:**软件可以配置该域，但是该域必须大于0x004.

SC 中断控制寄存器(SC_IER)

寄存器	偏移地址	R/W	描述	复位值
SC_IER	SCx_BA+0x18	R/W	SC 中断使能控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					ACON_ERR_IE	RTMR_IE	INIT_IE
7	6	5	4	3	2	1	0
CD_IE	BGT_IE	TMR2_IE	TMR1_IE	TMR0_IE	TERR_IE	TXBE_IE	RDA_IE

位	描述	
[31:11]	**Reserved**	保留
[10]	**ACON_ERR_IE**	**自动约定错误中断使能位** 该域用于自动约定错误中断使能. 0 = 禁止自动约定错误中断. 1 = 使能自动约定错误中断.
[9]	**RTMR_IE**	**接收器缓存超时中断使能位** 该域用于接收器缓存超时中断使能 0 = 禁止接收缓存超时中断. 1 = 使能接收缓存超时中断.
[8]	**INIT_IE**	**初始化结束中断使能位** 该域用于激活(寄存器SC_ALTCTL的第3位ACT_EN = 1)，释放(寄存器SC_ALTCTL的第二位DACT_EN = 1)和暖复位(寄存器SC_ALTCTL的第4位WARST_EN)序列中断使能 0 = 禁止初始化结束中断. 1 = 使能初始化结束中断.
[7]	**CD_IE**	**卡检测中断使能位** 该域用于卡检测中断使能，卡检测状态的寄存器是SC_SR的12位CD_INS_F。 0 = 禁止卡检测中断. 1 = 使能卡检测中断.
[6]	**BGT_IE**	**块保护时间中断使能位** 该域用于块保护时间中断使能 0 = 禁止块保护时间中断 1 = 使能块保护时间中断.
[5]	**TMR2_IE**	**定时器2中断使能位**

		该域用于TRM2中断使能 0 = 禁止定时器2中断. 1 = 使能定时器2中断
[4]	**TMR1_IE**	**定时器1中断使能位** 该域用于TRM1中断使能 0 = 禁止定时器1中断. 1 = 使能定时器1中断
[3]	**TMR0_IE**	**定时器0中断使能位** 该域用于TRM0中断使能 0 = 禁止定时器0中断. 1 = 使能定时器0中断
[2]	**TERR_IE**	**传输错误中断使能位** 该域用于传输错误中断使能。传输错误状态在SC_SR寄存器，包括接收器断开错误(寄存器SC_SR的第6位RX_EBR_F)，帧错误(寄存器SC_SR的第5位RX_EFR_F)，校验错误(寄存器SC_SR的第4位RX_EPA_F)，接收缓存溢出错误(寄存器SC_SR的第0位RX_OVER_F)，传输缓存溢出错误(寄存器SC_SR的第8位RX_OVER_F)，接收器重接收超过限制错误(寄存器SC_SR的第22位RX_OVER_REERR)和传送器重传超过限制错误(寄存器SC_SR的第30位TX_OVER_REERR)。 0 = 禁止传输错误中断. 1 = 使能传输错误中断
[1]	**TXBE_IE**	**传送缓存空中断使能位**. 该域用于传送缓存空中断使能 0 = 禁止传送缓存空中断. 1 = 使能传送缓存空中断
[0]	**RDA_IE**	**接收数据到达中断使能位** 该域用于接收数据达到触发水平寄存器SC_CTL的第6到7位RX_FTRI_LEV的中断使能 0 = 禁止接收数据达到触发水平中断. 1 = 使能接收数据达到触发水平中断.

SC 中断状态寄存器(SC_ISR)

寄存器	偏移地址	R/W	描述	复位值
SC_ISR	SCx_BA+0x1C	R/W	SC 中断状态寄存器	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					ACON_ERR_IS	RTMR_IS	INIT_IS
7	6	5	4	3	2	1	0
CD_IS	BGT_IS	TMR2_IS	TMR1_IS	TMR0_IS	TERR_IS	TBE_IS	RDA_IS

位	描述	
[31:11]	**Reserved**	保留.
[10]	**ACON_ERR_IS**	**自动约定错误中断状态标志 (只读)** 该域指示自动约定序列错误。如果在ATR状态下收到的TS既不是0x3B也不是0x3F，该位将被置位。 **注意:**该位只读，但是可以写1清除。
[9]	**RTMR_IS**	**接收器缓存超时中断状态标志 (只读)** 该域用于接收器缓存超时中断状态标志。 **注意:**，该域是接收器缓存超时状态的状态标志。如果软件要清除该位，软件必须通过读SC_RBR寄存器读取接收器缓存中剩下的数据
[8]	**INIT_IS**	**初始化结束中断状态标志(只读)** 该位用于激活(寄存器SC_ALTCTL的第3位ACT_EN)，释放(寄存器SC_ALTCTL的第2位DACT_EN)和暖复位(SC_ALTCTL的第4位WARST_EN)序列中断状态标志 **注意:**该位只读，但是可以写1清除。
[7]	**CD_IS**	**卡检测中断状态标志(只读)** 该域用于卡检测中断状态标志。卡检测的状态是寄存器SC_SR的第12位CD_INS_F和第11位CD_REM_F。 **注意:** 该域是寄存器SC_SR的第12位CD_INS_F和第11位CD_REM_F的状态标志。所以软件想要清除该位，必须对该位写1。
[6]	**BGT_IS**	**块保护时间中断状态标志 (只读)** 该域用于块保护时间中断状态标志 **注意1:** 当寄存器SC_ALTCTL的12位RX_BGT_EN使能时，该位才有效。 **注意2:** 该位只读，但是可以写1清除。
[5]	**TMR2_IS**	**定时器2中断状态标志 (只读)** 该域用于TMR2中断状态标志

		注意:该位只读，但是可以写1清除
[4]	**TMR1_IS**	**T定时器1中断状态标志 (只读)** 该域用于TMR1中断状态标志 **注意:**该位只读，但是可以写1清除
[3]	**TMR0_IS**	**定时器0中断状态标志 (只读)** 该域用于TMR0中断状态标志 **注意:**该位只读，但是可以写1清除
[2]	**TERR_IS**	**传输错误中断状态标志(只读)** 该域用于传输错误中断状态标志。传输错误状态在SC_SR寄存器，包括接收器断开错误(寄存器SC_SR的第6位RX_EBR_F)，帧错误(寄存器SC_SR的第5位RX_EFR_F)，校验错误(寄存器SC_SR的第4位RX_EPA_F)，接收缓存溢出错误(寄存器SC_SR的第0位RX_OVER_F)，传输缓存溢出错误(寄存器SC_SR的第8位RX_OVER_F)，接收器重接收超过限制错误(寄存器SC_SR的第22位RX_OVER_REERR)和传送器重传超过限制错误(寄存器SC_SR的第30位TX_OVER_REERR)。 **注意:**该域是寄存器SC_SR的第6位RX_EBR_F，寄存器SC_SR的第5位RX_EFR_F，寄存器SC_SR的第4位RX_EPA_F，寄存器SC_SR的第0位RX_OVER_F，寄存器SC_SR的第8位RX_OVER_F，寄存器SC_SR的第22位RX_OVER_REERR和 (寄存器SC_SR的第30位TX_OVER_REERR的状态标志。如果软件想清除该位，必须写1到该位。
[1]	**TBE_IS**	**传输缓存空中断状态标志(只读)** 该域用于传输缓存空中断状态标志 **注意:**该域是传输缓存空状态的状态标志。如果软件想清除该位，必须写数据到寄存器SC_THR的0到7位THR同时该位将自动被清除。
[0]	**RDA_IS**	**接收数据到达中断状态标志 (只读)** 该域用于接收数据到达触发水平(寄存器SC_CTL的第6到7位RX_FTRI_LEV)中断状态标志 **注意:** 该域是接收数据到达寄存器SC_CTL的第6到7位RX_FTRI_LEV状态标志。如果软件从SC_RBR读取数据而且接收缓存的数据字节数少于寄存器SC_CTL第6到7位RX_FTRI_LEV，该位将自动被清除。

SC 传输状态寄存器(SC_TRSR)

寄存器	偏移地址	R/W	描述	复位值
SC_TRSR	SCx_BA+0x20	R/W	SC 状态寄存器	0x0000_0202

31	30	29	28	27	26	25	24
TX_ATV	TX_OVER_REERR	TX_REERR	Reserved		TX_POINT_F		
23	22	21	20	19	18	17	16
RX_ATV	RX_OVER_REERR	RX_REERR	Reserved		RX_POINT_F		
15	14	13	12	11	10	9	8
Reserved					TX_FULL_F	TX_EMPTY_F	TX_OVER_F
7	6	5	4	3	2	1	0
Reserved	RX_EBR_F	RX_EFR_F	RX_EPA_F	Reserved	RX_FULL_F	RX_EMPTY_F	RX_OVER_F

位	描述	
[31]	TX_ATV	**传输激活状态标志(只读)** 0 = 当TX传输完成或者最后一个字节传输已经完成，则该位自动清除 1 = 当TX传输处于激活状态而且最后字节的STOP位已经发送被发送，硬件将置位该位。
[30]	TX_OVER_REERR	**发送器重传超过限制次数错误(只读)** 当发送器重传次数超过限制时，该位被置位 **注意:**该位只读，但是能够写1清除该位。
[29]	TX_REERR	**发送重传错误(只读)** 当发送器重传时，该位被置位。 **注意1:** 该位只读，但是能够写1清除该位。. **注意2:** 该位只是一个标志位，不能产生任何中断给CPU。
[28:26]	Reserved	保留
[25:24]	TX_POINT_F	**发送缓存指针状态标志 (只读)** 该域指示TX缓存指针状态标志。当CPU写数据到SC_THR，TX_POINT_F增加1。当TX缓存的一个字节被传输到发送器移位寄存器时，TX_POINT_F减少1。
[23]	RX_ATV	**接收器激活状态标志(只读)** 当RX传输处于激活时，该位被置位。 当RX传输完成时，该位被启动清除。
[22]	RX_OVER_REERR	**接收器重接收次数超过限制错误(只读)** 当RX传输错误重接收超过重接收次数限制时，该位有硬件置位。 **注意1:** 该位只读，但是能够写1清除该位。 **注意2:**如果CPU通过设置寄存器SC_CTL的19位RX_ERETRY_EN使能接收器重接收功能，寄存器SC_TRSR的第4位RX_EPA_F标志将被忽略(硬件将不会设置寄存器SC_TRSR的第4位RX_EPA_F)。

[21]	**RX_REERR**	**接收器重传错误 (只读)** 当RX有任何错误并且重传时，该位有硬件置位。 **注意1:** 该位只读，但是能够写1清除该位。 **注意2:** 该位是标志，不能产生任何中断给CPU。 **注意3:**.如果CPU通过设置寄存器SC_CTL的19位RX_ERETRY_EN使能接收器重试功能，寄存器SC_TRSR的第4位RX_EPA_F将被忽略(硬件将不能设置寄存器SC_TRSR的第4位RX_EPA_F)。
[20:18]	**Reserved**	保留
[17:16]	**RX_POINT_F**	**接收器缓存指针状态标志(只读)** 该域指示RX缓存指针状态标志。当SC从外部设备收到一个字节，寄存器SC_SR的第16到17位RX_POINT_F加1。当RX缓存被CPU读取了一个字节，寄存器SC_SR的第16到17位RX_POINT_F减1。
[15:11]	**Reserved**	保留
[10]	**TX_FULL_F**	**发送缓存满状态标志 (只读)** 该位指示TX缓存是否满了。当TX指针等于4时，该位被置位，否则会被硬件清除。
[9]	**TX_EMPTY_F**	**发送缓存空状态标志(只读)** 该位指示TX缓存是否为空。 当TX缓存的最后一个字节已经被传送到发送器移位寄存器后，硬件将置位为高。当写数据到寄存器SC_THR的第0到7位THR时(TX缓存不为空)，该位被清除。
[8]	**TX_OVER_F**	**TX溢出错误中断状态标志(只读)** 如果TX缓存为满，一个额外的写数据到寄存器SC_THR的0到7位THR将导致该位由硬件置为1。 **注意:** 该位只读，但是能够写1清除该位
[7]	**Reserved**	保留
[6]	**RX_EBR_F**	**接收器断开错误状态标志(只读)** 每当接收到的输入数据(RX)保持在“spacing state”(逻辑0)状态的时间长过一个完整的字节传输时间(也就是“start bit”+ data bits + parity + stop bits 的总时间)，该位被设置为逻辑1。 **注意1:** 该位只读，但是能够写1清除该位. **注意2:** 如果CPU通过设置寄存器SC_CTL的第19位RX_ERETRY_EN设置接收器重接收功能，硬件将不会设置该标志位。
[5]	**RX_EFR_F**	**接收器帧错误状态标志(只读)** 每当接收的字符没有一个有效的”“停止位”(也就是紧跟在最后一位数据位或校验位后的停止位被检测为逻辑0)，该位被置为逻辑1。 **注意1:** 该位只读，但是能够写1清除该位. **注意2:** 如果CPU通过设置寄存器SC_CTL的第19位RX_ERETRY_EN设置接收器重接收功能，硬件将不会设置该标志位。
[4]	**RX_EPA_F**	**接收器校验错误状态标志(只读)** 每当收到的字符没有有效的“校验位”，该位设置为逻辑1. **注意1:** 该位只读，但是能够写1清除该位. **注意2:** 如果CPU通过设置寄存器SC_CTL的第19位RX_ERETRY_EN设置接收器重接收功能，硬件将不会设置该标志位。
[3]	**Reserved**	保留
[2]	**RX_FULL_F**	**接收器缓存满状态标志(只读)** 该位指示RX缓存是否为满。

		如果RX指针等于4时该位将被置位，否则该位将被硬件清除。
[1]	**RX_EMPTY_F**	**接收缓存空状态标志(只读)** 该位指示RX缓存是否为空。 当RX缓存的最后一个字节被CPU读取后，硬件将置位为高。当SC收到任何新的数据，该位被清除。
[0]	**RX_OVER_F**	**RX溢出错误状态标志R (只读)** 当RX缓存溢出时，该位被置位。 如果收到的数据个数大于RX缓存的大小(4字节)，该位将被置位。 **注意:** 该位只读，但是能够写1清除该位。

SC 管脚控制状态寄存器 (SC_PINCSR)

寄存器	偏移地址	R/W	描述	复位值
SC_PINCSR	SCx_BA+0x24	R/W	SC 管脚控制状态寄存器	0x0000_00x0

31	30	29	28	27	26	25	24
Reserved	SYNC	Reserved					
23	22	21	20	19	18	17	16
Reserved							SC_DATA_I_ST
15	14	13	12	11	10	9	8
Reserved				POW_INV	CD_LEV	SC_DATA_O	SC_OEN_ST
7	6	5	4	3	2	1	0
ADAC_CD_EN	CLK_KEEP	Reserved	CD_PIN_ST	CD_INS_F	CD_REM_F	SC_RST	POW_EN

位	描述	
[31]	**Reserved**	保留
[30]	**SYNC**	**同步标志指示器** 由于同步，当写新值到寄存器SC_PINCSR时，软件必须检查该位 0 = 同步完成，用户可以写新值到寄存器SC_PINCSR 1 = 上一值正在同步中. **注意:** 该位只读.
[29:17]	**Reserved**	保留
[16]	**SC_DATA_I_ST**	**SC 数据输入管脚状态 (只读)** T该位是SC_DATA_I的管脚状态 0 = SC_DATA 管脚为低. 1 = SC_DATA 管脚为高.
[15:12]	**Reserved**	保留
[11]	**POW_INV**	**SC_POW 管脚反向** 该位用于SC_POW管脚反向 通过寄存器SC_PINCSR的第11位POW_INV和第0位POWEN，SC_POW管脚的设置有4种组合。POW_INV和POW_EN的组合为SC_POW_PIN提供高或低电压选择。 00 = SC_POW_ Pin为0. 01 = SC_POW _Pin为1. 10 = SC_POW _Pin为1. 11 = SC_POW_ Pin为0. **注意:**.在通过寄存器SC_CTL的第0位SC_CEN使能智能卡之前，软件必须选择好寄存器SC_PINCSR的11位POW_INV。
[10]	**CD_LEV**	**卡检测电平**

		0 = 当硬件检测到卡检测管脚从高到低，该位指示检测到一张卡 1 = 当硬件检测到卡检测管脚从低到高，该位指示检测到一张卡  **注意:** 软件必须在智能卡引擎使能之前，选择卡检测的电平。
[9]	**SC_DATA_O**	**SC 数据输出管脚** 该位是SC_DATA_O的管脚状态，但是用户可以通过设置该位驱动SC_DATA_O管脚为高或低。 0 = 驱动SC_DATA_O 管脚到低. 1 = 驱动SC_DATA_O 管脚到高. **注意:**.当SC处于激活，暖复位或者释放模式时，该位将被自动改变，所以SC在这些模式下，不要设置该位。
[8]	**SC_OEN_ST**	**SC 数据输出使能管脚状态(只读)** 该位是SC_DATA_OEN的管脚状态 0 = SC_DATA_OEN 管脚状态为低 1 = SC_DATA_OEN 管脚状态为高.
[7]	**ADAC_CD_EN**	**当卡移除时，自动释放** 0 = 禁止当硬件检测到卡移除时自动释放 1 = 使能当硬件检测到卡移除时自动释放 **注意:**当卡被移除时，硬件将停止所有处理然后进行释放序列(如果该位被置位)。当该过程完成后，硬件将产生一个初始化结束中断给CPU。
[6]	**CLK_KEEP**	**SC 时钟使能位** 0 = 禁用SC 时钟产生 1 = 使能SC 时钟总是保持自由运行 **注意:** 如果操作在激活，暖复位或者释放模式时，该位将会自动改变。所以当操作在这些模式时，不要填充该位。
[5]	**Reserved**	保留
[4]	**CD_PIN_ST**	**SC_CD卡检测状态管脚状态 (只读)** 该位为SC_CD的管脚状态 0 = SC_CD 管脚状态为低. 1 = SC_CD 管脚状态为高
[3]	**CD_INS_F**	**SC_CD管脚卡检测插入状态 (只读)** 每当卡插入时，该位被置位。 0 = 不起作用 1 = 卡插入. **注意1:** 该位只读，可以写1清除 **注意2:** 卡检测引擎将在寄存器SC_CTL的第0位SC_CEN设置后开启。
[2]	**CD_REM_F**	**SC_CD管脚卡检测移除状态 (只读)** 每当卡移除后，该位被置位 0 = 不起作用 1 = 卡移除

		注意1: 该位只读，可以写1清除 **注意2:** 卡检测引擎将在寄存器SC_CTL的第0位SC_CEN设置后开启。
[1]	**SC_RST**	**SC_RST 管脚信号** 该位是SC_RST的管脚状态，但是用户可以通过设置该位驱动SC_RST管脚为高或低 写该域来驱动SC_RST管脚 0 = 驱动SC_RST管脚为低. 1 = 驱动SC_RST管脚为高. 读该域来获取SC_RST管脚状态. 0 = SC_RST管脚状态为低. 1 = SC_RST管脚状态为高. **注意:**当操作在激活，暖复位或释放模式时，该位会自动改变。所以当操作在这些模式时，不要填充该位。
[0]	**POW_EN**	**SC_POW_EN 管脚信号** 软件可以通过设置寄存器SC_PINCSR的第0位POW_EN和第11位POW_INV来决定SC_PWR管脚为高电平或者低电平 写该域以驱动SC_PWR管脚 编程SC_PWR管脚电平可参考寄存器SC_PINCSR的第11位POW_INV的描述 读该域来获取SC_PWR管脚状态 0 = SC_PWR 管脚状态为低 1 = SC_PWR 管脚状态为高. **注意:**当操作在激活，暖复位或释放模式时，该位会自动改变。所以当操作在这些模式时，不要填充该位。

SC 定时器控制寄存器 0 (SC_TMR0)

寄存器	偏移地址	R/W	描述	复位值
SC_TMR0	SCx_BA+0x28	R/W	SC 内部定时器控制寄存器 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				MODE			
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							

位	描述	
[31:28]	**Reserved**	保留
[27:24]	**MODE**	**定时器 0 操作模式选择** 该域指示内部24位定时器的操作选择 参考5.14.5.4 定时器0编程
[23:0]	**CNT0**	**定时器 0 计数器值(基于ETU)** 该域指示内部定时器操作值

SC 定时器控制寄存器 1 (SC_TMR1)

寄存器	偏移地址	R/W	描述	复位值
SC_TMR1	SCx_BA+0x2C	R/W	SC 内部定时器控制寄存器 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				MODE			
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							

位	描述	
[31:28]	**Reserved**	保留
[27:24]	**MODE**	**定时器 1 操作模式选择** 该域指示内部8位定时器的操作选择 参考5.14.5.4 定时器1编程
[7:0]	**CNT1**	**定时器 1 计数器值(基于ETU)** 该域指示内部定时器操作值

SC 定时控制寄存器 2 (SC_TMR2)

寄存器	偏移地址	R/W	描述	复位值
SC_TMR2	SCx_BA+0x30	R/W	SC 内部定时器控制寄存器 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				MODE			
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							

位	描述	
[31:28]	**Reserved**	保留
[27:24]	**MODE**	**定时器 2 操作模式选择** 该域指示内部8位定时器的操作选择 参考5.14.5.4 定时器2编程
[7:0]	**CNT2**	**定时器 2 计数器值(基于ETU)** 该域指示内部定时器操作值

SC 串口模式控制寄存器 (SCx_UACTL)

寄存器	偏移地址	R/W	描述	复位值
SC_UACTL	SCx_BA + 0x34	R/W	SC 串口模式控制寄存器.	0x0000_0000

31	30	29	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	PBDIS	DATA_LEN		Reserved			UA_MODE_EN

位	描述	
[31:8]	Reserved	保留
[7]	OPE	**奇检验使能位** 0 = 奇数个逻辑1被发送，或者在接收模式下检查数据字和校验位 1 = 偶数个逻辑1被发送，或者在接受模式下检查数据字或校验位 **注意:** .该位仅在PBDIS位为0时起作用
[6]	PBDIS	**校验位禁止位** 0 = 校验位产生或者在串行数据的“最后一个数据位”和“停止位”之间被检查. 1 = 校验位不产生(发送数据)或者在传输过程中不被检查(接收数据) **注意:** 当工作在智能卡模式时，该域必须设置为0(默认设置带校验位)
[5:4]	DATA_LEN [1:0]	**数据长度** 00 = 字符数据长度为8位. 01 = 字符数据长度为7位. 10 = 字符数据长度为6位. 11 = 字符数据长度为5位. **注意:** 当工作在智能卡模式时，该DATA_LEN必须设置为00。
[3:1]	Reserved	保留
[0]	UA_MODE_EN	**串口模式使能位** 0 = 智能卡模式 1 = 串口模式 **注意1:** 当工作于串口模式，用户必须设置寄存器SC_CTL的第4到5位CON_SEL=00和第3位AUTO_CON_EN=0。 **注意 2:** 当工作在智能卡模式时，用户必须设置寄存器SC_UACTL的第0位UA_MODE_EN=0。 **注意3:**当串口功能被使能时，硬件将产生一个复位信号来复位内部缓存和内部状态机。

SC 定时器当前数据寄存器 A (SC_TDRA)

寄存器	偏移地址	R/W	描述	复位值
SC_TDRA	SCx_BA+0x38	R	SC 定时器当前数据寄存器 A	0x0000_07FF

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
TDR0							
15	14	13	12	11	10	9	8
TDR0							
7	6	5	4	3	2	1	0
TDR0							

位	描述	
[31:24]	Reserved	保留
[23:0]	TDR0	定时器0 当前数据值(只读) 该域指示定时器0当前的计数值

SC 定时器当前数据寄存器B (SC_TDRB)

寄存器	偏移地址	R/W	描述	复位值
SC_TDRB	SCx_BA+0x3C	R	SC 定时器当前数据寄存器 B	0x0000_7F7F

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TDR2							
7	6	5	4	3	2	1	0
TDR1							

位	描述	
[31:16]	Reserved	保留
[15:8]	TDR2	**定时器2 当前数据值(只读)** 该域指示定时器2当前的计数值
[7:0]	TDR1	**定时器1 当前数据值(只读)** 该域指示定时器1当前的计数值

5.15 PS/2 设备控制器 (PS2D)

5.15.1 概述

PS/2 设备控制器为PS/2 通讯提供基本时序控制. 所有在设备和主机之间的通讯都是通过PS2_CLK 和PS2_DATA 引脚控制. 不同于PS/2 键盘和鼠标设备控制器, 接收/传输代码需要固件进行代码转换成有意义的代码. PS/2 设备控制器在接收到“发送请求”后产生PS2_CLK信号，但是在通信过程中主机拥有最终的控制权。主机发送到设备的数据是在上升沿读取，设备向主机发送的数据在上升沿之后改变. 16 个字节的FIFO用来减少CPU 的介入.关于连续传输，软件可设定1 ~ 16 字节FIFO.

5.15.2 特性

- 主机通讯禁止和”请求发送”状态侦测
- 接收帧错误侦测
- 1 ~ 16字节的可编程传输缓存以减少CPU的介入
- 数据接收的双缓存
- 软件重写总线

5.15.3 系统方块图

PS/2 设备驱动器由APB 接口和用于PS2_DATA 和PS2_CLK 的时序控制逻辑部分组成

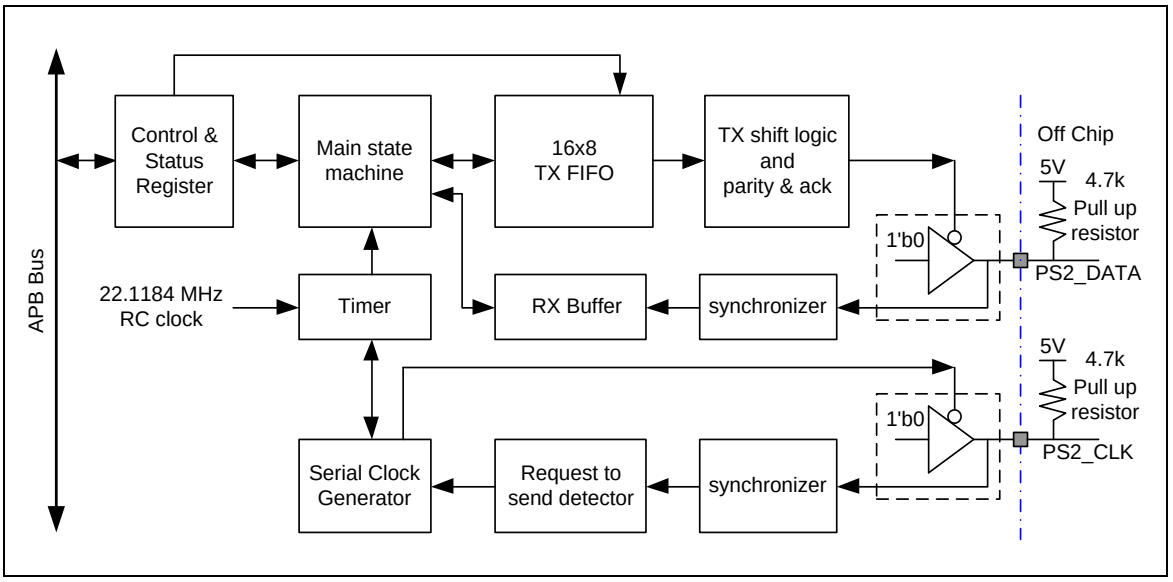


图 5-89 PS/2 设备框图

5.15.4 基本设定

PS/2设备控制器的基本设定如下：

- PS2_CLK and PS2_DATA 的引脚功能在 GPF_MFP[3:2]寄存器中设定
- PS/2 设备的总线时钟使能（PS2_EN）在 APBCLK[31]寄存器中设定
- PS/2 设备的复位（PS2_RST）在 IPRSTC2[23]寄存器中设定

5.15.5 功能描述

5.15.5.1 *通信*

PS/2 设备具有双向同步串行协议。当两条线都为高(open-collector) 时，总线为"空闲" 模式. 该状态为设备开始PS2_DATA传输的唯一状态. 主机在总线上有最终的控制权，并且任何时刻都可以通过下拉PS2_CLK线来禁止通讯.

PS2_CLK 信号由PS/2设备产生，如果主机需要发送PS2_DATA, 其必须首先下拉PS2_CLK线为低，禁止设备进行通讯，然后拉PS2_DATA为低并且释放PS2_CLK. 这是"请求发送"的状态, 通知设备开始发送PS2_CLK脉冲.

PS2_DATA	PS2_CLK	总线状态
1	1	空闲
1	0	通讯禁止
0	1	主机请求发送

所有数据每次传输1 字节，每个字节被包含在一个11或12位的帧中。这些位的定义如下：

- 1个开始位. 总是为0
- 8个数据位，低位在前
- 1 个校验位(奇校验)
- 1 个停止位. 该位一直为1
- 1 个应答位(只是主机和设备通讯时)

如果数据位有偶数个1，校验位将置1；如果数据位有奇数个1，校验位将清0. 数据位中的1的个数加上校验位，其和总是一个奇数，而这个可用于奇偶错误侦测，设备需要检查该位，如果该位错误，应该返回无效命令的响应。

主机可在任何时候通过拉低PS2_CLK 线至少100us 来禁止通讯。如果11th 时钟脉冲之前传输被禁止，设备必须退出当前传输，当主机释放CLK 时，设备将准备重新传输当前数据。为了有充足的时间让软件解码主机命令，可通过设定RXINT(PS2INTID[0])位来阻塞传输逻辑，软件必须清RXINT 位来重新开始传输. 如果有需要，软件可写CLRFIFO(PS2CON[8])为1来复位FIFO 指针

设备向主机传输

设备使用一个11位的帧窜行协议，位的定义如下：

- 1 个开始位：内容始终为0
- 8 个数据位. 最低位优先传输
- 1个校验位(奇校验)

- 1个停止位。该位内容总是为1

当PS2_CLK为高时，设备写1位数据到PS2_DATA总线，当PS2_CLK为低时，该数据被主机读走

相关说明见图6-71：

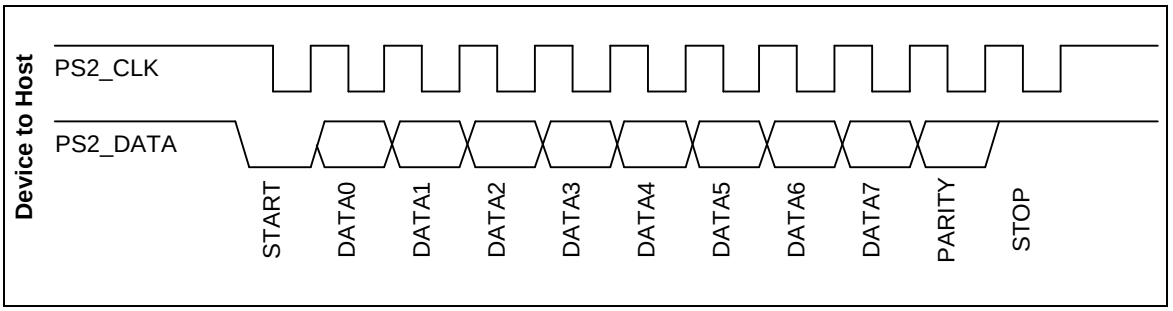


图 5-90 设备向主机传输的数据格式

主机向设备传输:

首先，PS/2 设备总是先产生一个PS2_CLK信号，如果主机想发送一个PS/2数据，它必须把PS2_CLK和 PS2_DATA线拉到“请求发送”的状态，要求如下：

- 拉低PS2_CLK至少100 us 以禁止通讯
- 拉PS2_DATA l 为低，然后释放PS2_CLK来“请求发送”

设备以不超过10 ms 的间隔不间断的监控状态，当设备检测到此状况，它将产生PS2_CLK信号，将8位数据、1位校验位、1位停止位接收过来，主机只能在PS2_CLK为低时才可改变PS2_DATA；当PS2_CLK为高时，设备读取PS2_DATA。

当设备接收到停止位后，设备将发出应答信号：把PS2_DATA线拉低并且产生最后一个CLK 脉冲。如果在第十一个CLK脉冲后主机未释放PS2_DATA线，设备将继续产生PS2_CLK脉冲直到PS2_DATA 线 释放

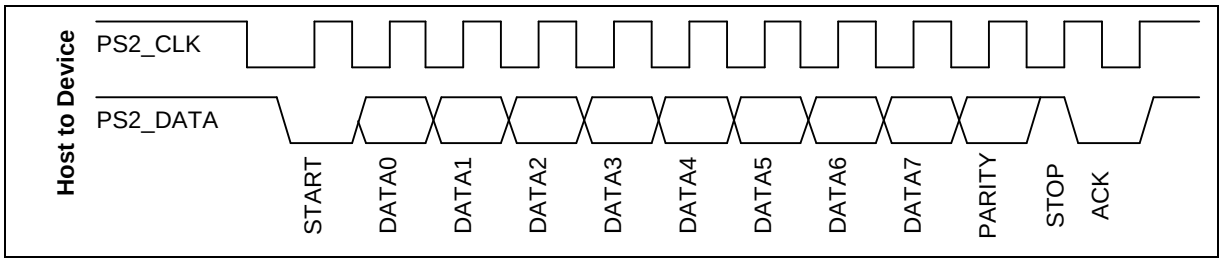


图 5-91 主机向设备传输的数据格式

主机和外设通讯的DATA 和CLK 详细时序框图如下：

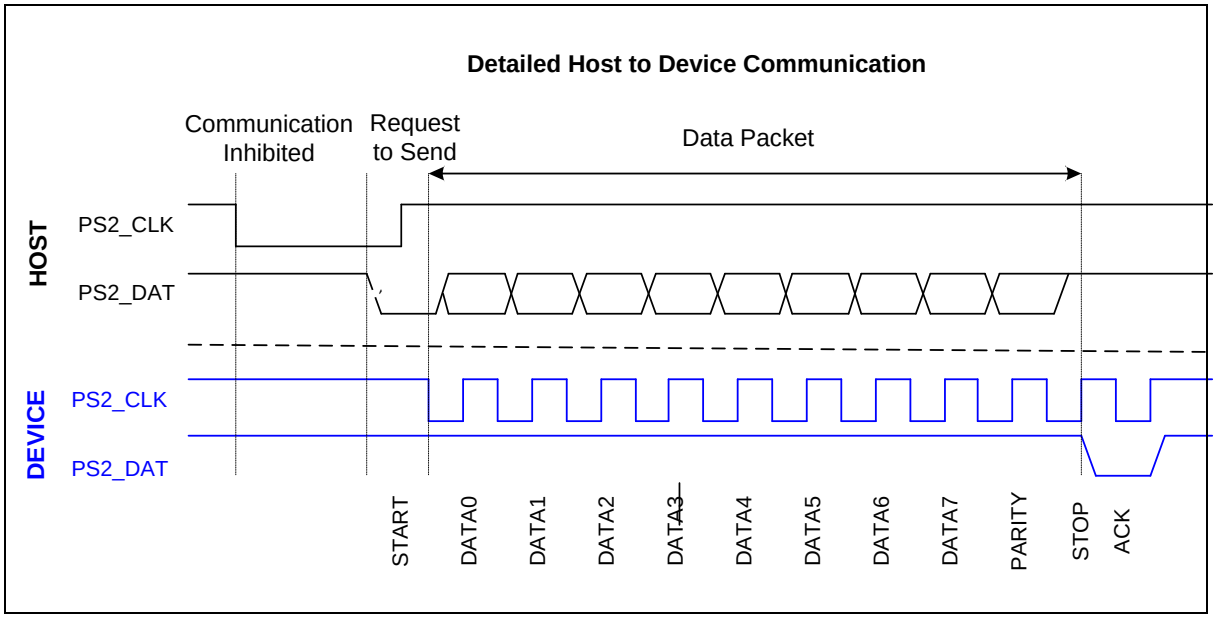


图 5-92 PS/2 Bit 数据格式

5.15.5.2 *PS/2 总线时序规范*

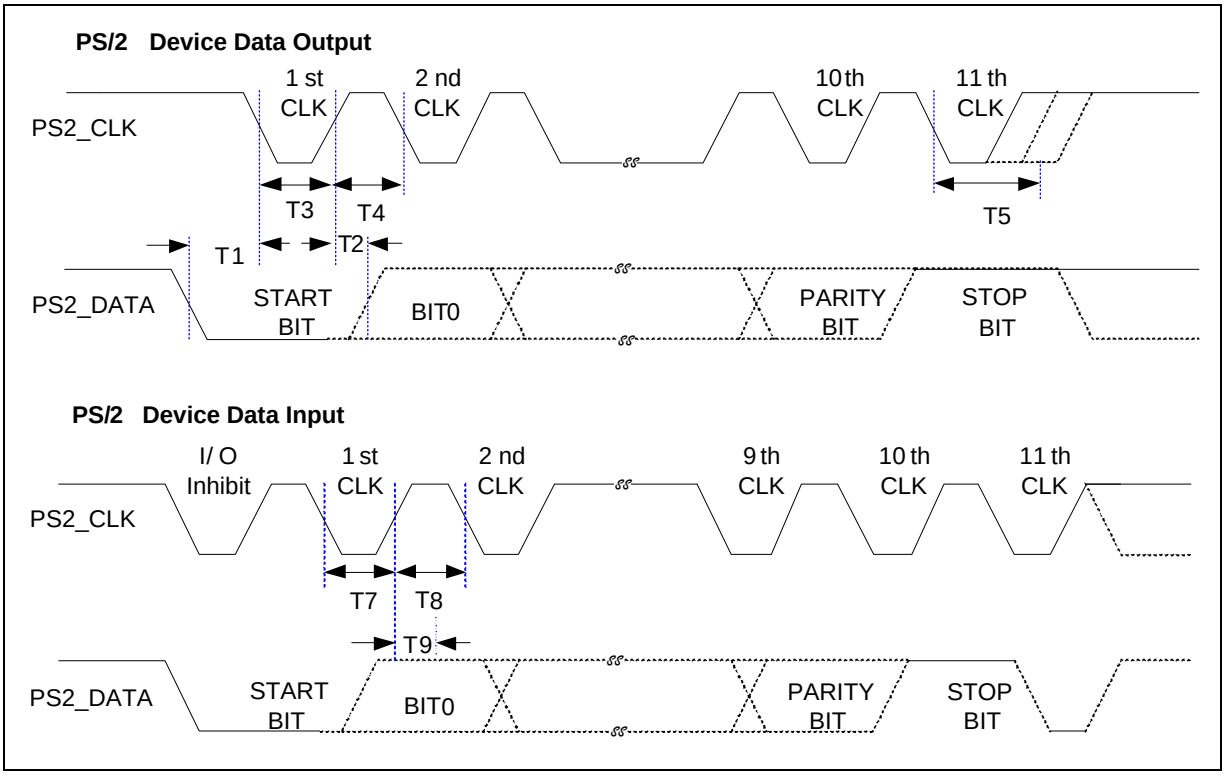


图 5-93 PS/2 总线时序

符号	时序参数	Min.	Max.

T1	PS2_DATA 转换到 PS2_CLK的下降沿	5us	25us
T2	PS2_CLK的上升沿 到PS2_DATA 转换	5us	T4-5us
T3	PS2_CLK 持续无效时间	30us	50us
T4	PS2_CLK持续有效时间	30us	50us
T5	11th clock 后 辅助设备禁止其他传输	>0	50us
T7	PS2_CLK 持续无效时间	30us	50us
T8	PS2_CLK持续有效时间	30us	50us
T9	PS2_CLK从无效到有效转换时间, 辅助设备取样时间	5us	25us

5.15.5.3 *TX FIFO 操作*

写数据到PS2TXDATA0寄存器将触发设备和主机通讯. 在向TX FIFO 写发送数据之前，软件需要定义TXFIFO 的长度. 在写PS2TXDATA0 寄存器之後100us ，第一个字节的start bit 才会被传送到总线上。如果要传送的数据超过四个字节，软件在第四个字节数据传输完成前可写剩余数据到PS2TXDATA1-3 。2个连续的字节之间将延时100us

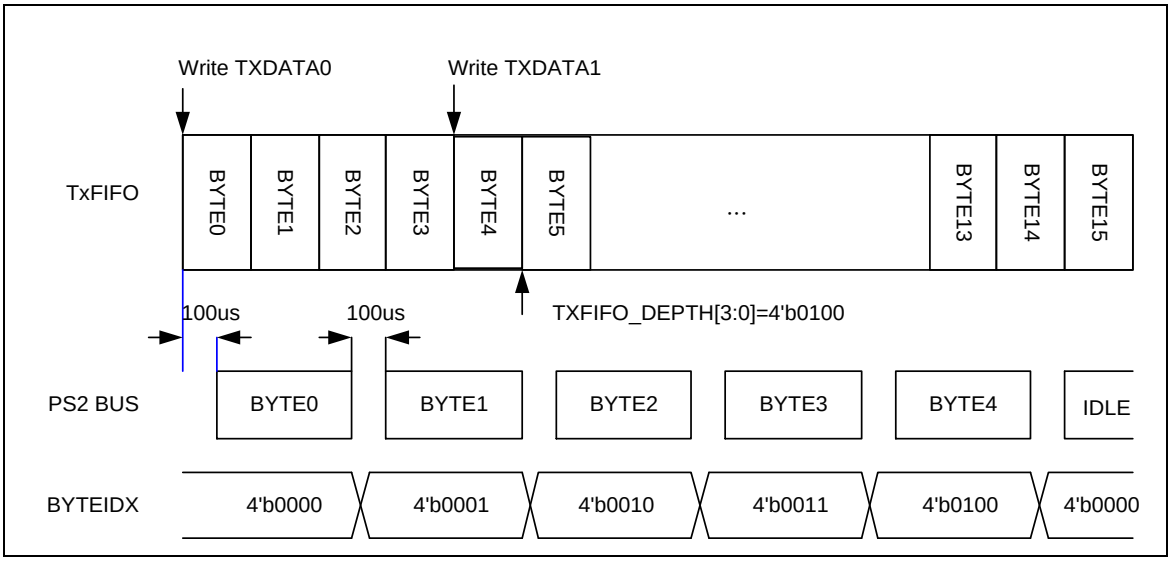


图 5-94 PS/2 数据格式

5.15.6 寄存器映射

R: 只读, **W**: 只写, **R/W**:读/写

寄存器	偏移地址	R/W	描述	复位值
PS/2 基地址: **PS2_BA = 0x4010_0000**				
PS2CON	PS2_BA+0x00	R/W	PS/2 控制寄存器	0x0000_0000
PS2TXDATA0	PS2_BA+0x04	R/W	PS/2 发送数据寄存器0	0x0000_0000
PS2TXDATA1	PS2_BA+0x08	R/W	PS/2 发送数据寄存器1	0x0000_0000
PS2TXDATA2	PS2_BA+0x0C	R/W	PS/2 发送数据寄存器2	0x0000_0000
PS2TXDATA3	PS2_BA+0x10	R/W	PS/2发送数据寄存器3	0x0000_0000
PS2RXDATA	PS2_BA+0x14	R	PS/2 接收数据寄存器	0x0000_0000
PS2STATUS	PS2_BA+0x18	R/W	PS/2 状态寄存器	0x0000_0083
PS2INTID	PS2_BA+0x1C	R/W	PS/2 中断指示寄存器	0x0000_0000

5.15.7 寄存器描述

PS/2 控制寄存器(PS2CON)

寄存器	偏移地址	R/W	描述	复位值
PS2CON	PS2_BA+0x00	R/W	PS/2 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				FPS2DAT	FPS2CLK	OVERRIDE	CLRFIFO
7	6	5	4	3	2	1	0
ACK	TXFIFO_DEPTH				RXINTEN	TXINTEN	PS2EN

位	描述	
[31:12]	保留	保留
[11]	FPS2DAT	**强制 PS2DATA 线** 若 OVERRIDE(PS2CON[9]) =1 ，它将强制PS2DATA 低或高，忽略设备控制器的内部状态 0 = 强制 PS2_DATA 低 1 = 强制PS2_DATA 高
[10]	FPS2CLK	**强制 PS2CLK 线** 若 OVERRIDE(PS2CON[9]) =1 ，I它将强制PS2CLK 线为低或高，忽略设备控制器的内部状态 0 = 强制PS2_CLK 低 1 = 强制PS2_CLK高.
[9]	OVERRIDE	**软件重写PS2 CLK/DATA 引脚状态** 0 = PS2_CLK 和 PS2_DATA 引脚由内部状态机控制 1 = PS2_CLK 和 PS2_DATA引脚由软件控制
[8]	CLRFIFO	**清 TX FIFO** 向该位写1将终止设备向主机传输. 无论数据缓冲器中是否有数据，TXEMPTY(PS2STATUS[7])位将被置1且指针BYTEIDX (PS2STATUS[11:8]) i将被清0，但数据缓冲器中的内容不会被清除. 0 = 无效 1 = 清楚 FIFO.
[7]	ACK	**应答使能位** 0 =主机到设备发送时，在12th 时钟总是发送应答位 1 =如果校验位错误或停止位未接收到，在12th 时钟将不发送应答.

[6:3]	TXFIFO_DEPTH	数据传输FIFO 长度 16 位缓冲器应用于数据传输，软件可根据应用可定义FIFO 长度，范围从1 ~ 16 字节 0 = 1 字节. 1 = 2 字节.. … 14 = 15字节.. 15 = 16字节.
[2]	**RXINTEN**	**接收中断使能位** 0 = 禁用数据接收完成中断 1 =使能数据接收完成中断
[1]	**TXINTEN**	**传送中断使能位** 0 =禁用数据传送完成中断 1 =使能数据传送完成中断
[0]	**PS2EN**	**PS/2 设备使能位** 0 = 禁用 1 = 使能.

PS/2发送数据寄存器 0-3 (PS2TXDATA0-3)

寄存器	偏移地址	R/W	描述	复位值
PS2TXDATA0	PS2_BA+0x04	R/W	PS/2 发送数据寄存器 0	0x0000_0000
PS2TXDATA1	PS2_BA+0x08	R/W	PS/2 T发送数据寄存器 1	0x0000_0000
PS2TXDATA2	PS2_BA+0x0C	R/W	PS/2发送数据寄存器 2	0x0000_0000
PS2TXDATA3	PS2_BA+0x10	R/W	PS/2发送数据寄存器 3	0x0000_0000

31	30	29	28	27	26	25	24
PS2TXDATAx[31:24]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PS2TXDATAx[23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PS2TXDATAx[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PS2TXDATAx[7:0]							

位	描述	
[31:0]	**PS2TXDATAx**	**发送数据** 如果总线在空闲转态.向寄存器写数据将启动设备到主机的通讯软件在写数据到TX 缓冲器前需使能PS2EN(PS2CON[0])

PS/2 接收数据寄存器 (PS2RXDATA)

寄存器	偏移地址	R/W	描述	复位值
PS2RXDATA	PS2_BA+0x14	R	PS/2 接收数据寄存器	0x0000_0000

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

位	描述	
[31:8]	**保留**	保留
[7:0]	**RXDATA**	**接收数据** 在主机到设备传输中，在应答位发送后，接收到的数据将从移位寄存器复制到PS2RXDATA寄存器．CPU 需在下一字节接收完成前读取此寄存器，否则数据将被改写，并且RXOVF（PS2STATUS[6]） 位将置1

PS/2 状态寄存器 (PS2STATUS)

寄存器	偏移地址	R/W	描述	复位值
PS2STATUS	PS2_BA+0x18	R/W	PS/2 转态寄存器	0x0000_0083

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				BYTEIDX[3:0]			
7	6	5	4	3	2	1	0
TXEMPTY	RXOVF	TXBUSY	RXBUSY	RXPARITY	FRAMERR	PS2DATA	PS2CLK

位	描述	
[31:12]	保留	保留
[11:8]	BYTEIDX	字节索引 表示哪个数据字节在发送移位寄存器中.当FIFO 所有数据传输完毕，该位清0. 该位只读 0000 = PS2TXDATA0[7:0]. 0001 = PS2TXDATA0[15:8]. 0010 = PS2TXDATA0[23:16]. 0011 = PS2TXDATA0[31:24]. 0100 = PS2TXDATA1[7:0]. 0101 = PS2TXDATA1[15:8]. 0110 = PS2TXDATA1[23:16]. 0111 = PS2TXDATA1[31:24]. 1000 = PS2TXDATA2[7:0]. 1001 = PS2TXDATA2[15:8]. 1010 = PS2TXDATA2[23:16]. 1011 = PS2TXDATA2[31:24]. 1100 = PS2TXDATA3[7:0]. 1101 = PS2TXDATA3[15:8]. 1110 = PS2TXDATA3[23:16]. 1111 = PS2TXDATA3[31:24].
[7]	TXEMPTY	发送FIFO 空 如果PS2EN(PS2CON[0])使能，当软件写任何数据到PS2TXDATA0-3时，将置TXEMPTY 位为0，当传输数据的字节数和TXFIFO_DEPTH (PS2CON[6:3])相等时，TXEMPTY 将置1. 0 = 有数据等待传输 1 = FIFO 为空 该位只读

[6]	**RXOVF**	**接收缓冲器覆盖** 0 = 不覆盖 1 = PS2RXDATA 寄存器中的数据被新收到的数据覆盖. 写1清该位
[5]	**TXBUSY**	**发送忙** 该位表示PS/2 驱动器当前正在发送数据. 0 = 空闲. 1 = 当前正在传送数据. 该位只读
[4]	**RXBUSY**	**接收忙** 该位表示PS/2 驱动器当前正在接收数据. 0 = 空闲 1 = 当前正在接收数据. 该位只读
[3]	**RXPARITY**	**接收校验位** 该位反应最后接收字节的校验位. (奇校验). 该位只读
[2]	**FRAMERR**	**帧错误** 对于主机到设备的通信，如果未接收到STOP位(逻辑 1)，该位置1。如果帧错误发生，PS/2_DATA I线在12th 时钟后保持低状态. 此时，软件应该驱动PS2CLK 一直发送时钟直到PS2DATA 释放成高状态后才停止发送时钟. 然后，设备发送“重传”命令到主机. 0 = 未发生帧错误. 1 = 出现帧错误. 写1清该位
[1]	**PS2DATA**	**数据引脚状态** 该位表示同步和取样后PS2_DATA 线的状态
[0]	**PS2CLK**	**时钟引脚状态** 该位表示同步后PS2_CLK 线的状态.

PS/2中断识别寄存器(PS2INTID)

寄存器	偏移地址	R/W	描述	复位值
PS2INTID	PS2_BA+0x1C	R/W	PS/2 中断识别寄存器	0x0000_0000

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

位	描述	
[31:3]	**保留**	保留
[1]	TXINT	**传送中断** 当STOP 位传输后，该位置1 ，如果TXINTEN(PS2CON[1]) 位置1，中断产生 0 = 无中断 1 = 发送中断发生 写1清该位为0.
[0]	**RXINT**	**接收中断** 在主机向设备传输时，当应答位发送，该位置位. 如果 RXINTEN(PS2CON[2]) 位置1，中断产生. 0 = 无中断 1 =接收中断发生. 写1清该位为0

5.16 I²C 总线控制器 (I²C)

5.16.1 概述

I²C为双线，双向串行总线，通过简单有效的连线方式实现器件间的数据交换。I²C标准是多主机总线，包括冲突检测和仲裁以防止在两个或多个主机尝试同时控制总线时发生的数据损坏。

5.16.2 特征

I²C通过I2Cn_SDA 及I2Cn_SCL两条线与连接在总线上的设备传输信息，总线的主要特征:

- 支持最多两个I²C总线控制器
- 支持主机/从机 模式
- 主从机之间双向数据传输
- 多主机总线支持 (无中心主机)
- 多主机间同时传输数据仲裁，避免总线上串行数据损坏
- 总线采用串行同步时钟，可实现设备之间以不同的速率传输
- 内建14位溢出定时器，当I2C总线中止且定时器溢出，产生I2C中断
- 时钟源可设以适用于不同速率控制
- 支持7位从地址模式(4个带掩码从地址)
- I²C 总线控制器支持多地址识别（4组从机地址带mask 选项）
- 支持唤醒功能

5.16.3 基本配置

I²C0的基本配置如下：

- I²C0管脚从寄存器GPA_MFP [9:8]配置
- 通过设置I2C0_EN (APBCLK [8])使能I²C0时钟
- 通过设置I2C0_RST(IPRSTC2 [8])复位I²C0控制器

I²C1的基本配置如下

- I²C1管脚从寄存器GPA_MFP [11:10]配置
- 通过设置I2C1_EN (APBCLK [9])使能I²C1时钟
- 通过设置I2C1_RST(IPRSTC2 [9])复位I²C1控制器

5.16.4 框图

I²C基本配置如下：

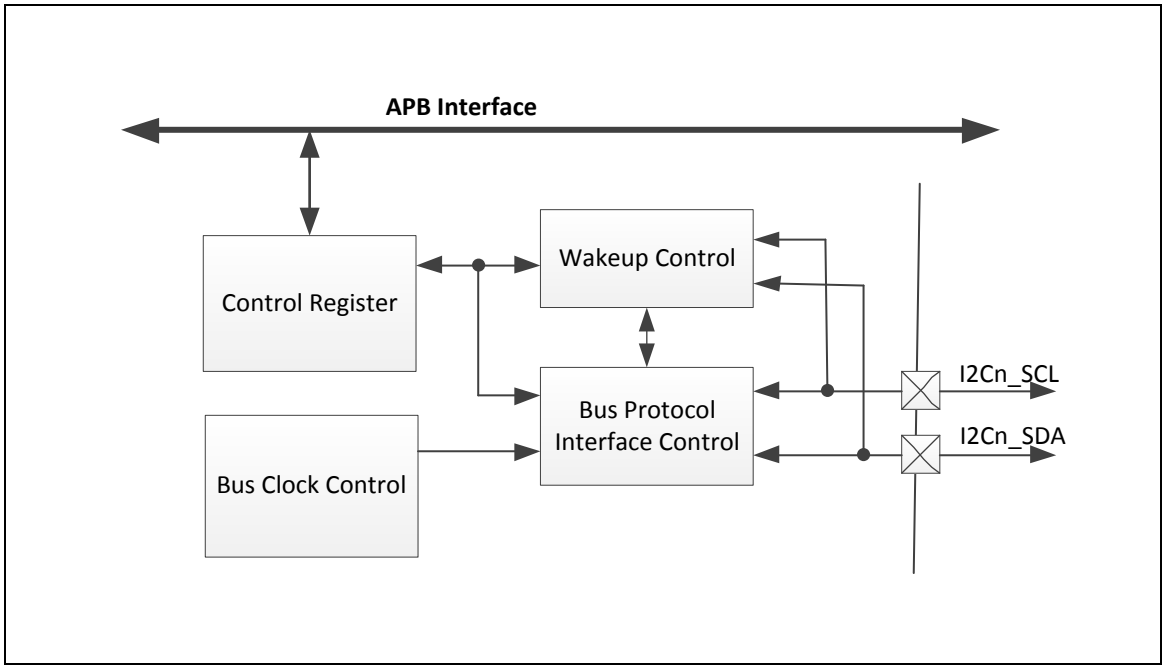


图 5-95 I²C 控制器模块框图

5.16.5 功能描述

I2C总线中，数据通过I2Cn_SCL和I2Cn_SDA在主从机间逐一字节同步传送。每个字节数据长度是8位. 一个SCL 时钟脉冲传输一个数据位，数据由最高位MSB 开始传输，每个传输字节后跟随一个应答位，每个位在SCL 为高时采样；因此，SDA 线只有在SCL 为低时才可以改变，在SCL 为高时SDA 保持稳定。当SCL 为高时，SDA 线上的跳变视为一个命令(START or STOP)。更多关于I2C总线时序的细节请参考 图6-43 。

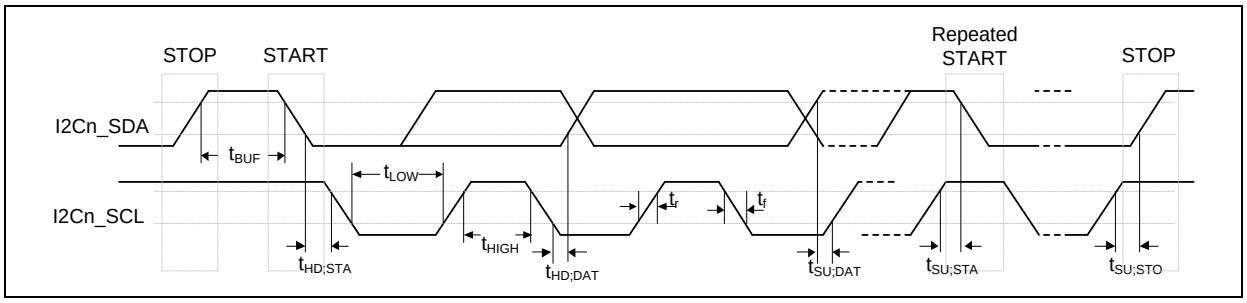


图 5-96 I²C 总线时序

片上I2C外设提供了一个符合I2C总线规范的串行接口。I2C端口自动处理字节传输，将ENS1(I2CON[6]) 设置为'1'，即可使能该端口。I2C 硬件接口通过I2Cn_SDA 与I2Cn_ SCL两个管脚连到I2C总线。当I/O 管脚作为I2C 端口使用时，用户必须事先设定I/O 管脚功能为I2C功能。

注意： I2Cn_SDA 和 I2Cn_SCL两个管脚需要上拉电阻，因为这个两个管脚为开漏脚。

5.16.5.1 *I²C协议*

标准I²C 协议如下图，通常标准通讯有以下4部分：

- 起始信号(START) 或者重复起始信号(Repeated START)
- 从机地址传输和R/W 位传输
- 数据传输
- 停止信号(STOP)

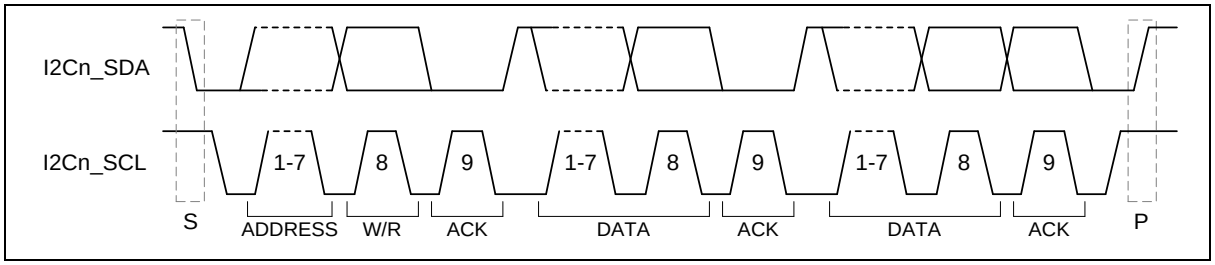


图 5-97 I^2C 协议

起始或重复起始信号

当总线处于空闲状态下，说明没有主机设备占用总线（I2Cn_SDA 与I2Cn_ SCL线同时为高），主机可以通过发送起始(START)信号来发起传输过程。起始信号，通常表示为S-bit，当SCL 线为高时，SDA 线上信号由高至低变化，就被定义为起始信号。起始信号表示一个新的数据传输的开始。

主机发送地址字节（地址和读/写位）后，可以发送任何数量的字节数据并带一个停止信号。也可以用另一个起始信号替代停止信号，随后是地址(包含读写位)和更多的数据。这个起始信号叫做重复起始。这个定义可以用来发送任意个起始信号，它的目的是在不释放总线情况下，对一个或多个设备能读/写操作，而不让操作被打断。用这个方法主机跟其他从机或同一个从机在不同方向传输（例如，写设备到读设备）不用释放总线。

停止（STOP）信号

主机可以通过产生一个停止信号来终止数据传送。停止信号，通常表示为P-bit，当I2Cn_SCL 线为高时，I2Cn_SDA线上信号由低至高变化，就被定义为停止信号。.

下图为起始，重复起始和停止信号：

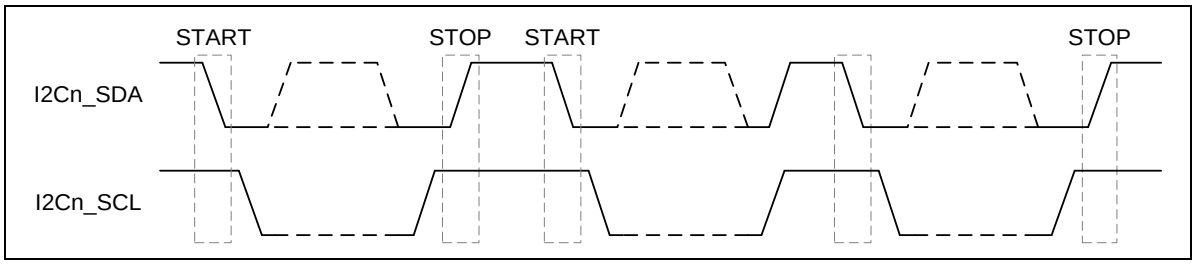


图5-98 起始(START) 和停止(STOP) 条件

从机地址传输

起始信号后立即传输的第一个字节就是从机地址(SLA)。这是一个7位设备地址跟随一个读/写(R/W)位。R/W 位标志从机的信号的数据传输方向。系统中没有两个从机有相同的地址，只有被主机寻址的从机设备才会通过在第9个I2Cn_SCL 时钟周期时将I2Cn_SDA 拉低作为应答

数据传输

当从机设备地址被成功识别，就可以根据R/W 位所决定的方向按一字节一字节方式进行数据传输。每个字节传输完后紧接着在第9个I2Cn_SC时钟周期会有一个应答信号位。如果从机上产生无应答信号(NACK)，主机可以产生停止信号来中止数据传输或者产生重复起始信号开始新一轮数据传输。

当主机作为接收设备时，发生无应答信号(NACK)，则从机释放I2Cn_SDA 线，让主机产生停止信号或重复起始信号。

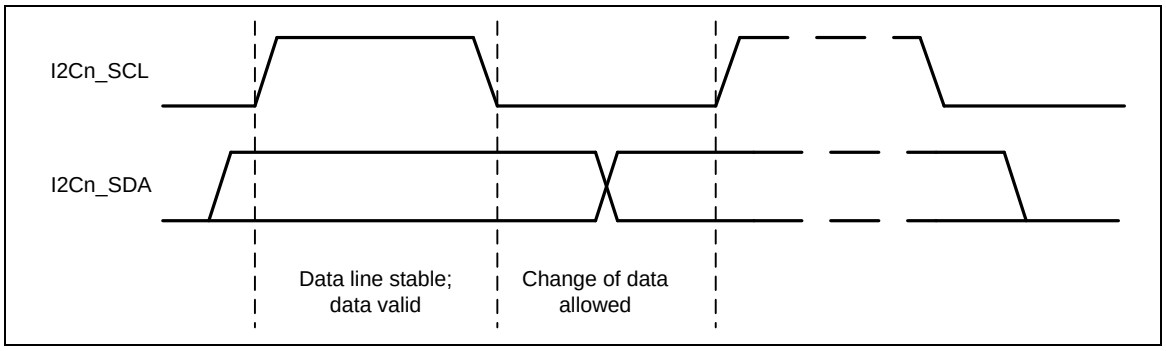


图 5-99 总线上的位传输

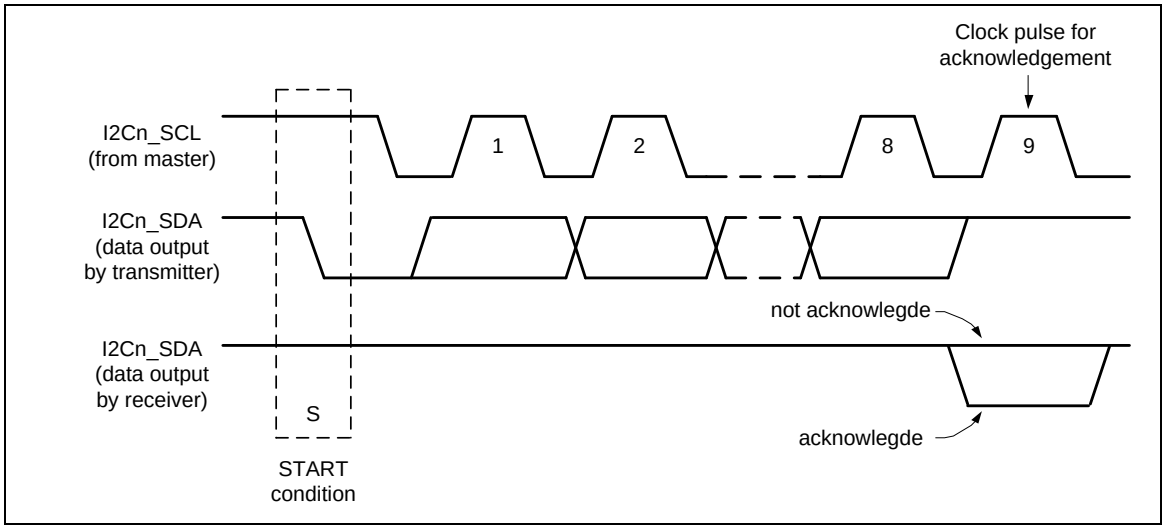


图 5-100 总线上的应答信号

I^2C总线上的数据传输

下图表示主机向从机传输数据。主机发出一个7-bit地址和1-bit写指令,表示主机想要传送数据给从机。从机回复应答给主机之后，主机继续传输数据。

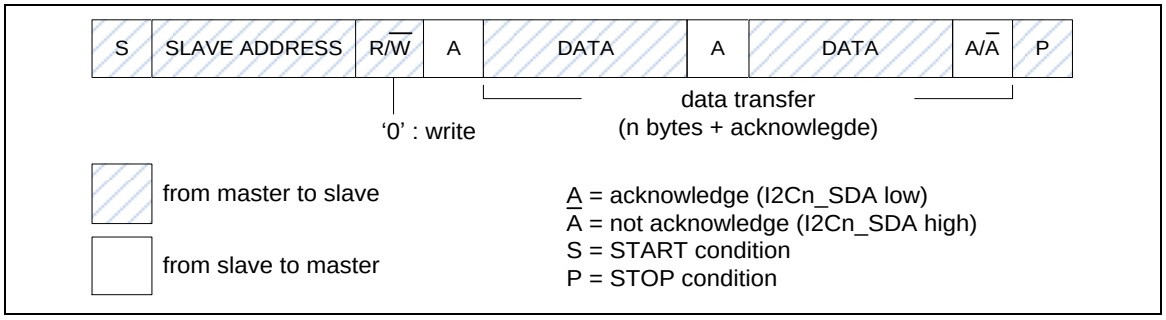


图 5-101 主机向从机传输数据

下图表示主机向从机读取数据。主机发7位地址寻址和1-bit读指示，表示主机要向从机读取数据，从机返回应答给主机后就开始传输数据。

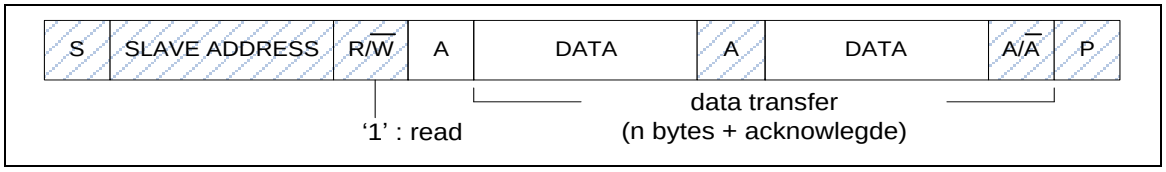


图 5-102 主机向从机读取数据

5.16.5.2 *操作模式*

片上I²C端口支持五种操作模式：主机传送，主机接收，从机传送，从机接收和广播呼叫模式。

应用中，I²C 端口可以作为主机和从机。在从机模式，I2C端口硬件查找自身从机地址或广播呼叫地址，如果这两个地址的任一个被检测到，并且从机打算从主机接收或向主机发送数据(通过设置AA位)，应答脉冲将会在第9个时钟被发出，此时，如果中断被使能，则在主机和从机设备上都会发生一次中断请求。在主控芯片要成为总线主机时，在进入主机模式之前，硬件等待总线空闲使合理的从机动作不会被打断，在主机模式下，如果总线仲裁丢失，I2C立即切换到从机模式，并可以在同一次串行传输过程中检测自身从机地址。

每种I²C总线传输模式中，用户都需要按照I2CSTATUS寄存器当前状态码来设置I2CON, I2CDAT。换句话说，每个I2C动作都要查询I2CSTATUS寄存器的当前状态，并设置I2CON, I2CDAT寄存器让总线动作。最后，通过I2CSTATUS检查响应状态。

SI (I2CON[3])标志清除后STA(I2CON[5]), STO(I2CON[4]) 和 AA(I2CON[2]) 用来控制I²C硬件的下一个状态。当完成一个新的动作，I2CSTATUS的状态代码将被更新，SI标志将被设置。如果I²C中断控制位EI (I2CON [7])被设置，新状态代码对应的动作或者软件将在中断服务程序中被执行。

下图显示当前I²C状态码是0x08，然后设置I2CDATA=SLA+W和(STA,STO,SI,AA) = (0,0,1,x)发送I²C总线地址。如果在总线上的从机地址相匹配则返回ACK，I2CSTATUS状态码更新为0x18。

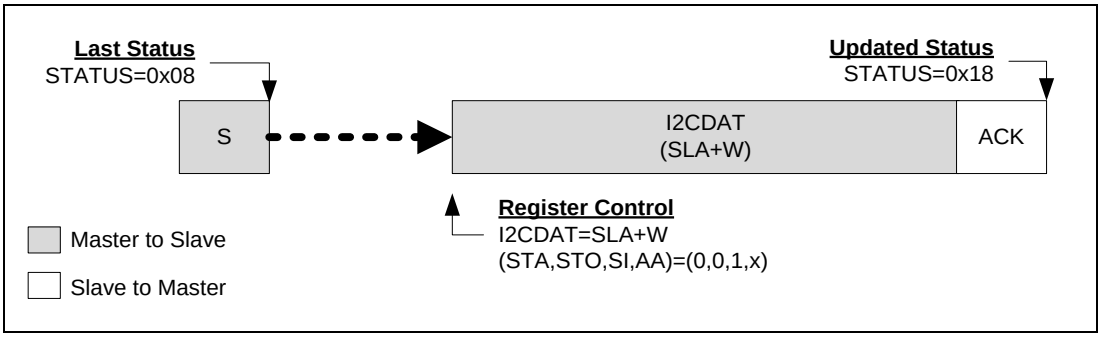


图 5-103 根据 I^2C 当前状态控制总线

主机模式

下图中，是I²C主机模式所有的协议。用户需要遵循恰当的流程来实现I²C协议。

换句话说，用户可以根据 (图6-51)发送一个起始信号到总线，I²C总线设置成主机传输模式，或根据(图6-53图 5-106)设置成主机接收模式。起始信号设置成功后新的状态码将是0x08。起始信号后，用户可以发送从机地址，读/写位，数据和重复起始，停止来执行I²C协议。

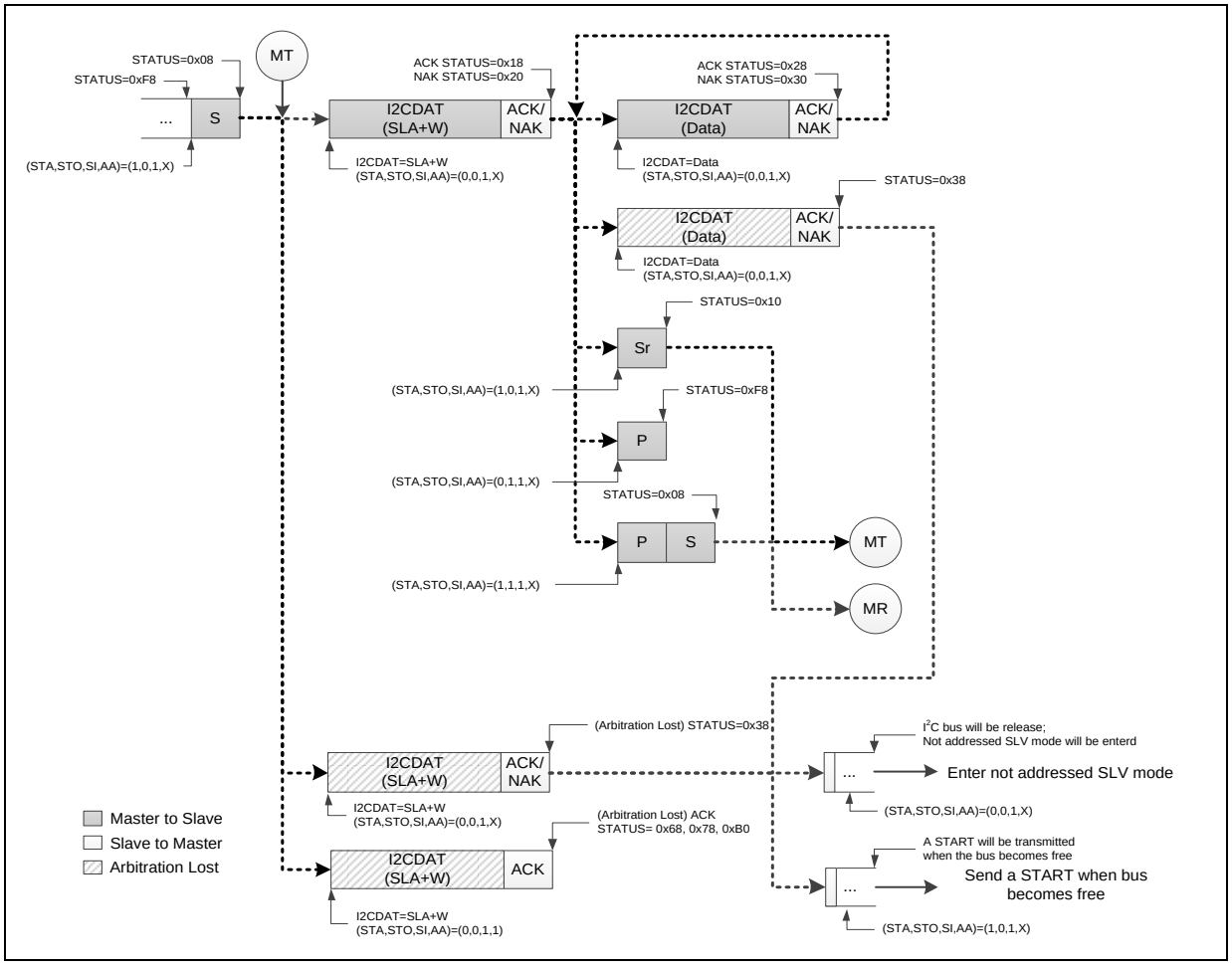


图 5-104 主机传输模式流程控制

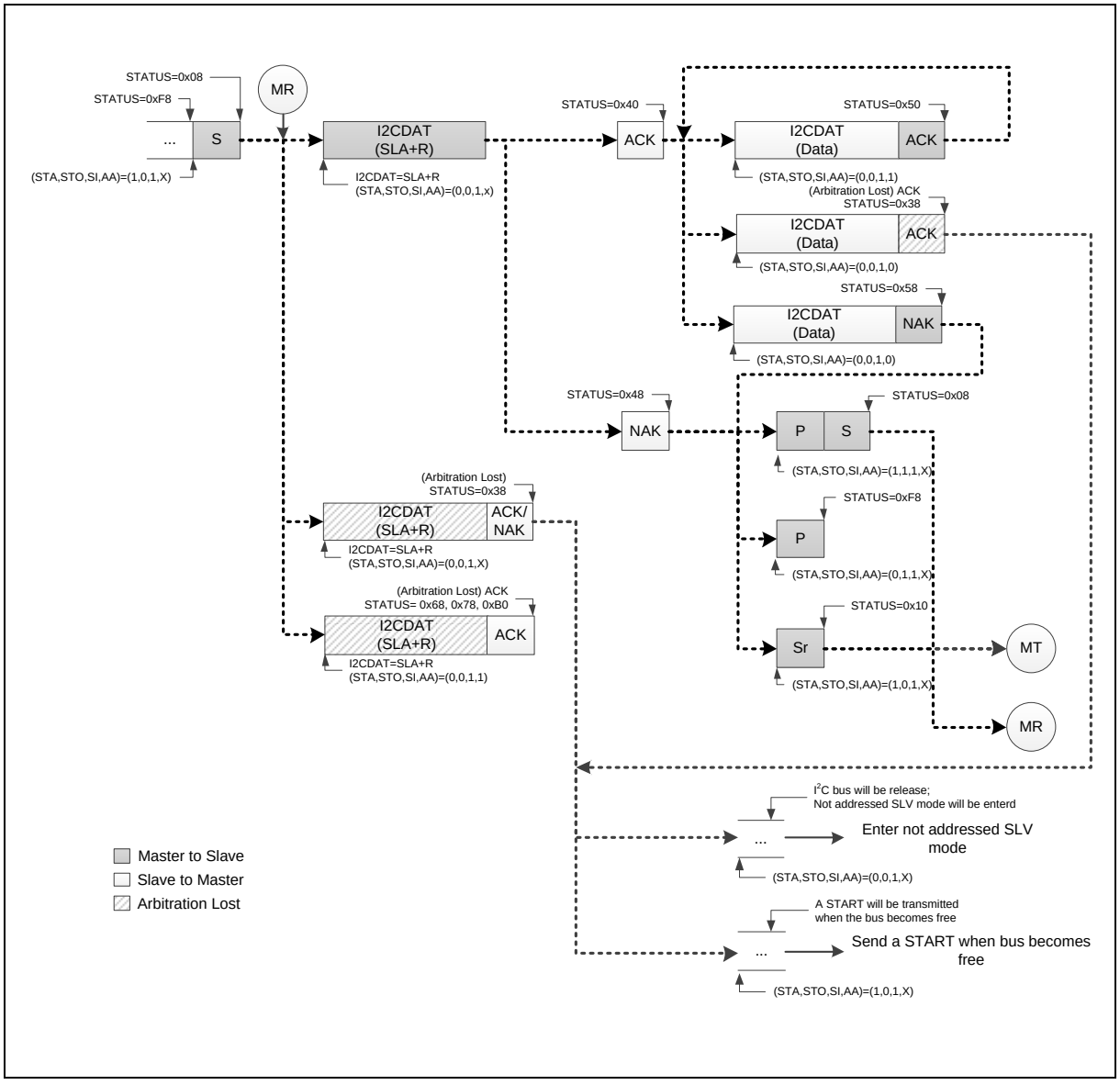


图 5-105 主机接收模式流程控制

如果I2C在主机模式并且仲裁丢失，状态码将置为0x38，在状态0x38时，用户可以设置(STA, STO, SI, AA) = (1, 0, 1, X)在总线空闲时发送起始信号来重新开始主机操作。另外，用户可以设置(STA, STO, SI, AA) = (0, 0, 1, X)来释放总线，并进入无地址从机模式。

从机模式

复位后默认情况下，I^{2}C不会被寻址，并且不会识别I^{2}C总线上的地址。用户可以通过I2CADDRx 和设置 (STA, STO, SI, AA) = (0, 0, 1, 1)来让I^{2}C识别主机发送的地址。图6-53所示，从机模式的所有流程。用户需要遵循（图6-53）的流程来实现他们的I^{2}C协议。

如果在主机模式仲裁丢失，I^{2}C端口立即切换到从机模式并且在同一串口传输中识别自有的从机地址。如果在仲裁丢失后识别到地址是SLA+W（主机写数据到从机），状态码是0X68。如果在仲裁丢失后识别到地址是SLA+R（主机向从机读数据），状态码是0xB0。

注意： I^{2}C通讯期间，当从机模式的SI标志被写‘1’(清除)，I2Cn_SCL clock将释放。

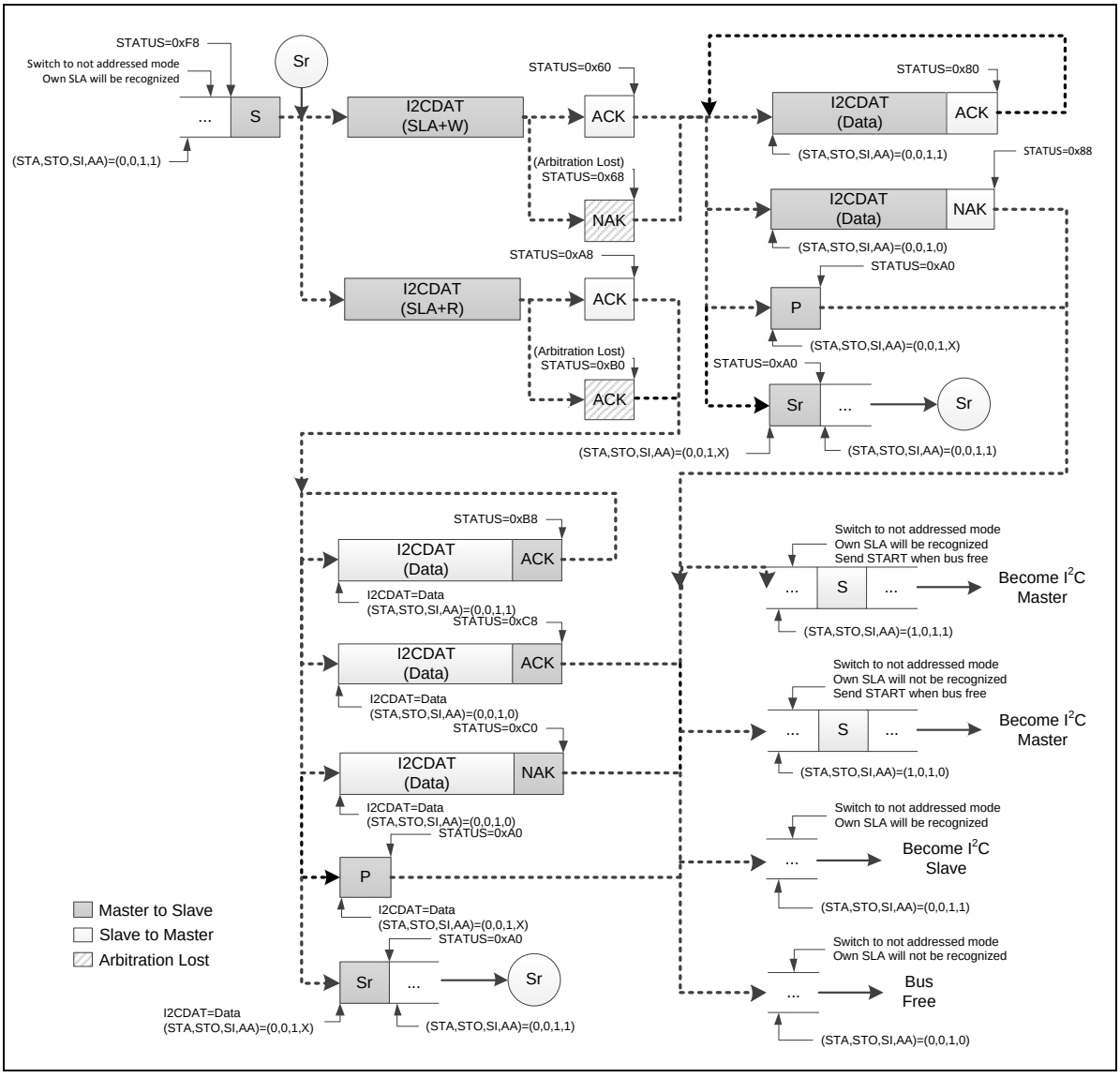


图 5-106 从机模式控制流程

如果I2C处在被寻址的从机接收数据模式却收到停止或重复起始信号，状态码是0xA0。当状态码是0xA0时用户可以遵循如上图状态码是0x88的操作。

如果I2C处在被寻址的从机传输数据模式却收到停止或重复起始信号，状态码是0xA0。当状态码是0xA0时，用户可以遵循如上图状态码是0xC8的操作。

注意： 从机获得0x88, 0xC8, 0xC0 和0xA0状态后,从机会切换到无地址模式，自身SLA不会被辨识。如果进入这种状态，从机不再从主机接收任何信号或地址。需要复位才能离开这种状态。

广播呼叫模式(GC)

如果 GC位(I2CADDRn [0]) 被设, I^{2}C端口硬件将响应广播呼叫地址(0x00).用户可以通过清GC位来禁止广播功能。当I^{2}C在从机模式时GC bit被设置，可以接收主机地址(0x00)的广播呼叫，将遵循广播模式状态。

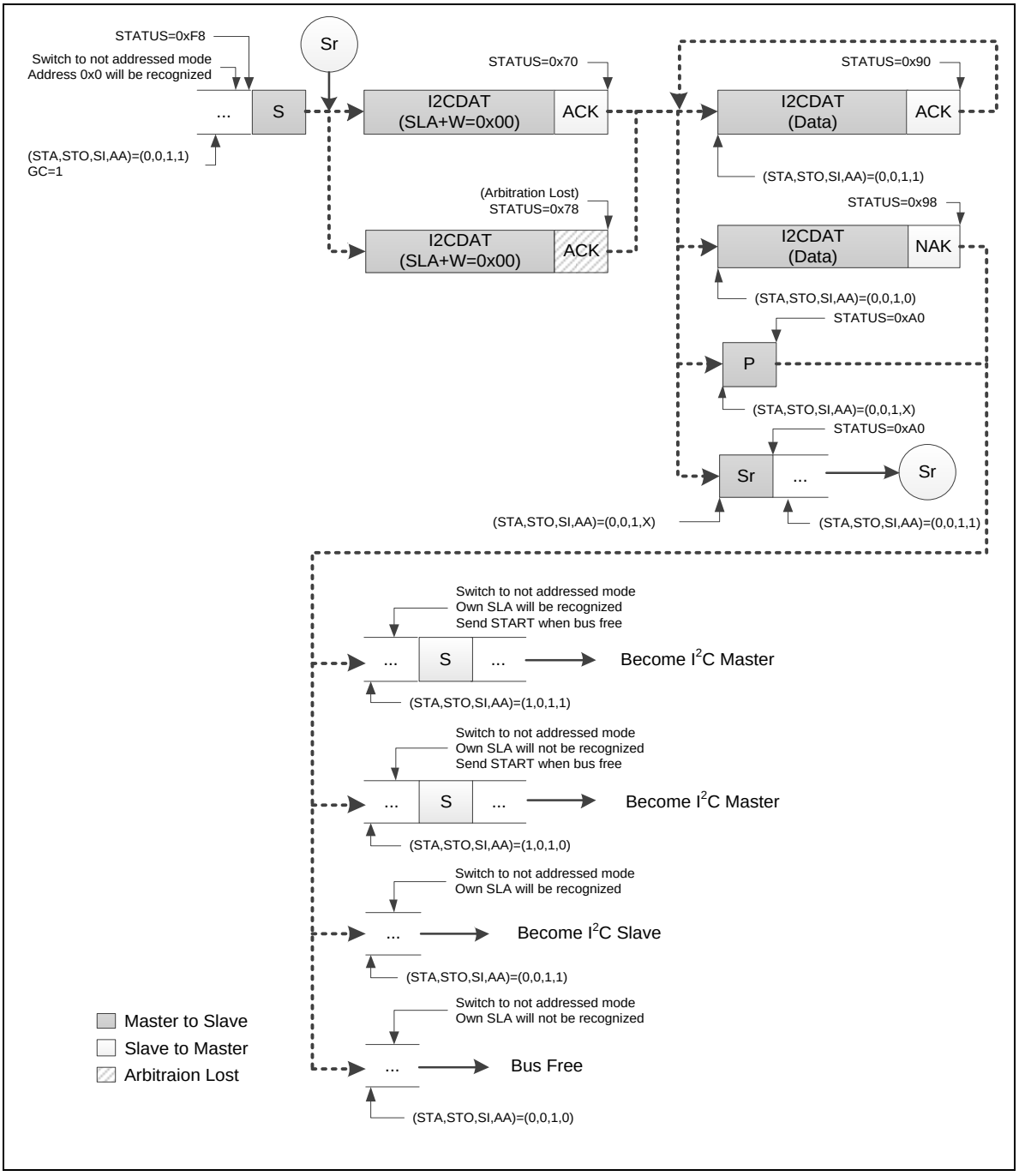


图 5-107 广播呼叫模式

如果I2C处在接收数据的广播呼叫模式，收到停止或重复起始信号，状态码是0xA0，用户可以遵循如上图状态码是0x98的流程处理。

注意： 从机获得0x98 和0xA0 状态后，从机会切换到无地址模式并且自身SLA将不被辨识。如果进入这种状态，从机不再从主机接收任何信号或地址。需要复位才能离开这种状态。

多主机模式

在一些应用中，一个I²C总线上有多个主机同时访问从机，并有可能同时在传送数据。I²C是支持多主机模式，并包含有冲突检测和仲裁，防止数据损坏。

如果两个主机同时发命令，通过仲裁来决定哪个优先并继续发命令。仲裁是在I2Cn_SCL为高时在I2Cn_SDA上执行的。每一个主机都会检测总线上的I2Cn_SDA信号是否符合它产生的I2Cn_SDA信号。如果检测到总线上2Cn_SDA为低，但应该为高，则这个主机将失去仲裁。设备在仲裁丢失后会产生I2Cn_SCL脉冲直到本字节结束。仲裁可以一直进行到所有数据被传输完。这样意味着多主机系统中主机必须监控总线和做相应的处理

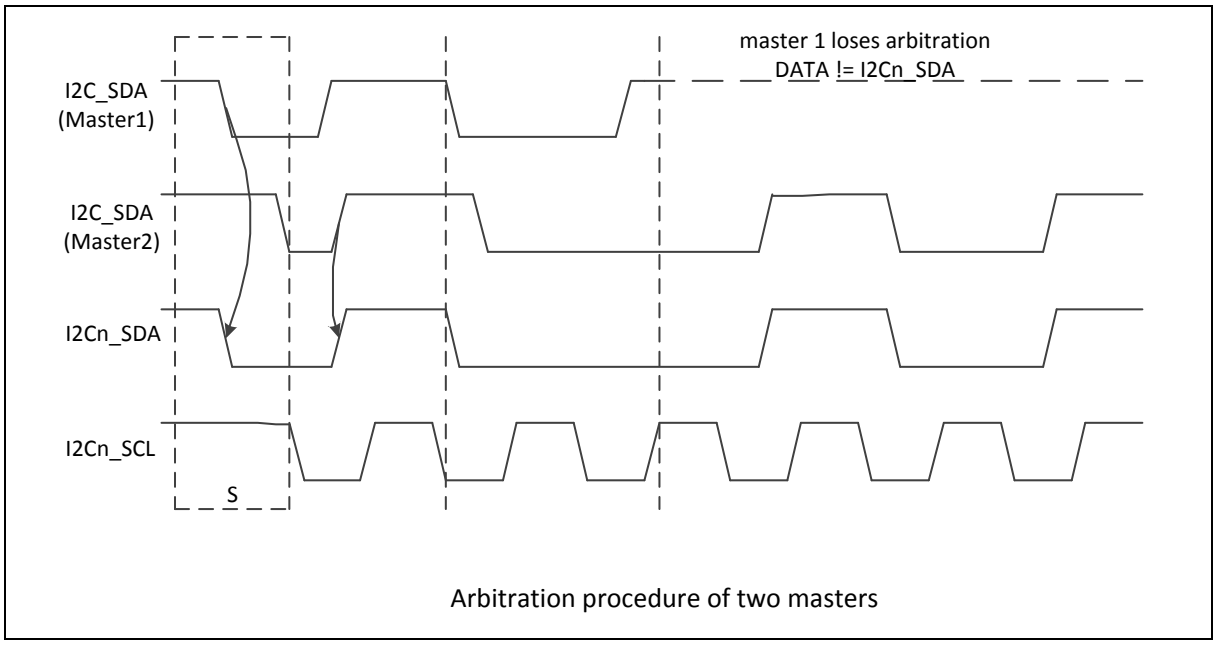


图 5-108 仲裁丢失

- 当I2CSTATUS = 0x38代表接收到一个仲裁丢失。仲裁丢失事件可能发生在发送起始位、数据位或者停止位期间。用户可以在总线空闲时设置(STA, STO, SI, AA) = (1, 0, 1, X)来再次发送起始位或者设置(STA, STO, SI, AA) = (0, 0, 1, X)返回到无地址从机模式。
- 当I2CSTATUS = 0x00，接收到一个“总线错误”，总线错误时STO(I2CON[4])应被设置、SI(I2CON[3])应将被清0，然后STO(I2CON[4])清0来释放总线
  - 设置(STA, STO, SI, AA) = (0, 1, 1, X)停止传输。
  - 设置(STA, STO, SI, AA) = (0, 0, 1, X)释放总线。

5.16.5.3 *I²C协议寄存器*

通过下列15个特殊功能寄存器来控制I²C端口：I2CON（控制寄存器）I2CSTATUS（状态寄存器），I2CDAT（数据寄存器），I2CADDRn（地址寄存器，n=0~3），I2CADMn（地址掩码寄存器，n=0~3），I2CLK（时钟速率寄存器），I2CTOC（超时寄存器），I2CWKCON(唤醒控制寄存器)，I2CWKSTS(唤醒状态寄存器)。

地址寄存器(I2CADDR)

I²C端口内建4个从机地址寄存器I2CADDRn (n=0~3)。当I2C作为主机时，这四个寄存器的内容不相关。在从机模式下，位字段I2CADDRn[7:1] 必须装载芯片自己的从机地址。当I2CADDRn 地址与接收到的从机地址符合时I²C硬件会作出反应。

I2C 端口支持“广播呼叫”功能。当GC位(I2CADDRn [0]) 被置，I2C端口硬件将应答广播呼叫的地址(00H)。清GC 位可禁用“广播呼叫”功能。

当GC 位被置且I2C处于从机模式时，主机发出广播呼叫地址到I2C总线后，从机可以通过地址00H接收广播呼叫地址，然后它将跟随GC 模式的状态。

从机地址掩码寄存器 (I2CADM)

I^2C总线控制器带有四个地址掩码寄存器I2CADMn (n=0~3) 支持多地址识别。当地址掩码寄存器被置1，意味着收到的相应地址位是"无效的".如果置0则收到相应地址位应跟地址寄存器完全一样。

数据寄存器 (I2CDAT)

该寄存器包含一个准备发送或刚接收到的一个字节的串行数据。当不在字节移位处理过程时，CPU可以直接读写访问I2CDAT[7:0]。当I2C 处于一个确定的状态并且串行中断标志(SI) 被置1，I2CDAT[7:0]中的数据保持稳定。当数据被移出，总线上的数据同时被移入。I2CDAT [7:0]的内容总是总线上传递的最后一个字节。

应答位由I^2C硬件控制，不能被CPU 访问。串行数据在I2Cn_SCL 线上串行时钟脉冲的上升沿移入I2CDAT [7:0] 。当一个字节被移入I2CDAT [7:0]，则I2CDAT [7:0] 中的串行数据是可用的，应答位(ACK 或NACK) 在第9个时钟脉冲由控制逻辑返回。为了在发送数据时监控总线的状态，当发送I2CDAT [7:0]到总线时，总线数据将同时被移入I2CDAT [7:0]。在发送数据过程中，串行数据在I2Cn_SCL 时钟脉冲的下降沿从I2CDAT [7:0] 移出，在I2Cn_SCL时钟脉冲的上升沿数据移入I2CDAT [7:0]。

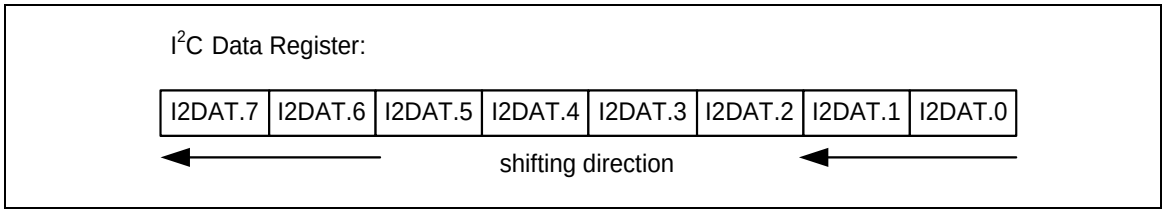


图 5-109 I^2C 数据移位方向

控制寄存器 (I2CON)

CPU 可以直接读或写寄存器I2CON，当I^2C端口通过设置ENS1 (I2CON [6])为高使能，内部状态由I2CON 和I^2C逻辑硬件控制。

有两个位会受硬件影响：当I2C硬件产生中断时SI(I2CON[3])被置位，当总线上产生停止信号时，STO位被清零，当ENS1(I2CON[6])=0时STO(I2CON[4])也会被清零

一旦有新的状态码产生并存储在I2CSTATUS，I^2C中断标志位SI将被自动置位。若此时使能中断EI (I2CON [7])位被设置，I^2C中断将产生。I2CSTATUS[7:0]用来存储内部状态码，SI(I2CON[3])被软件清除前内容一直保持。

状态寄存器 (I2CSTATUS)

I2CSTATUS [7:0] 是一个8-位只读寄存器。I2CSTATUS [7:0]共有26 个可能的状态码。当I2CSTATUS [7:0] 的值为0xF8时，没有串行中断请求。所有其他的I2CSTATUS [7:0]的值对应I^2C 的状态。当进入其中任一状态时，就会产生状态中断请求(SI (I2CON[3])= 1)。在SI 被硬件置位或SI 被

软件复位后一个机器周期，有效状态码出现在I2CSTATUS [7:0] 中。

此外，状态0x00表示总线错误，这会出现在起始 或 停止条件处在不正确的I^2C格式帧。总线错误会发生在一个地址字节、一个数据字节或一个应答位的串行传输。恢复错误总线时STO (I2CON[4])应被置位，SI(I2CON[3])应被清除，进入无地址从机模式。然后STO(I2CON[4])清除释放总线并等待下一个传输。出现总线错误操作期间I^2C总线不能识别停止条件。

主机模式		**从机模式**	
状态	**描述**	**状态**	**描述**
0x08	开始	0xA0	从机发送重新开始或停止
0x10	主机重复开始	0xA8	从机发送地址ACK
0x18	主机发送地址ACK	0xB0	从机发送仲裁丢失
0x20	主机发送地址NACK	0xB8	从机发送数据ACK
0x28	主机发送数据ACK	0xC0	从机发送数据NACK
0x30	主机发送数据NACK	0xC8	从机发送最后数据ACK
0x38	t主机仲裁丢失	0x60	从机接收地址 ACK
0x40	主机地址接收ACK	0x68	从机接收仲裁丢失
0x48	主机地址接收NACK	0x80	从机接收数据 ACK
0x50	主机数据地址接收 ACK	0x88	从机接收数据NACK
0x58	主机数据接收 NACK	0x70	广播模式地址ACK
0x00	总线错误	0x78	广播模式仲裁 Lost
		0x90	广播模式数据 ACK
		0x98	广播模式数据 NACK
0xF8	总线释放 **注意**：主/从模式的“0xF8”状态，都不会产生中断		

表 5-23 I^2C 状态码描述

时钟波特率位(I2CLK)

当I²C 在主机模式下，I²C数据的波特率由I2CLK[7:0] 寄存器设定。其在从机模式下时是不重要的。在从机模式下，I²C 将自动与主机I²C设备时钟频率同步。

I²C 数据波特率的设定：I²C数据波特率= (系统时钟) / (4x (I2CLK [7:0] +1))。如果系统时钟 = 16 MHz, the I2CLK [7:0] = 40 (28H), 那么I²C数据波特率= 16 MHz/ (4x (40 +1)) = 97.5 Kbits/sec。

I²C超时计数器寄存器((I2CTOC)

MCU 提供一个14 位超时的计数器来处理当I2C总线锁死时的情况。当计数功能使能后，计数器开始计数直至溢出(TIF=1) 并要求CPU 产生I2C中断或者清除ENTI 为0 关闭计数功能。当超时计数器使能，对SI 标志置高会使计数器复位，再对SI 清零会重新开始计数。如果I2C总线锁死，会使I2CSTATUS 及SI 标志不再更新，该14-位超时计数器会发生溢出从而产生I2C中断通知CPU。参考下图14-位超时计数器。用户可以写1 清TIF(I2C_TOC[0])为0。

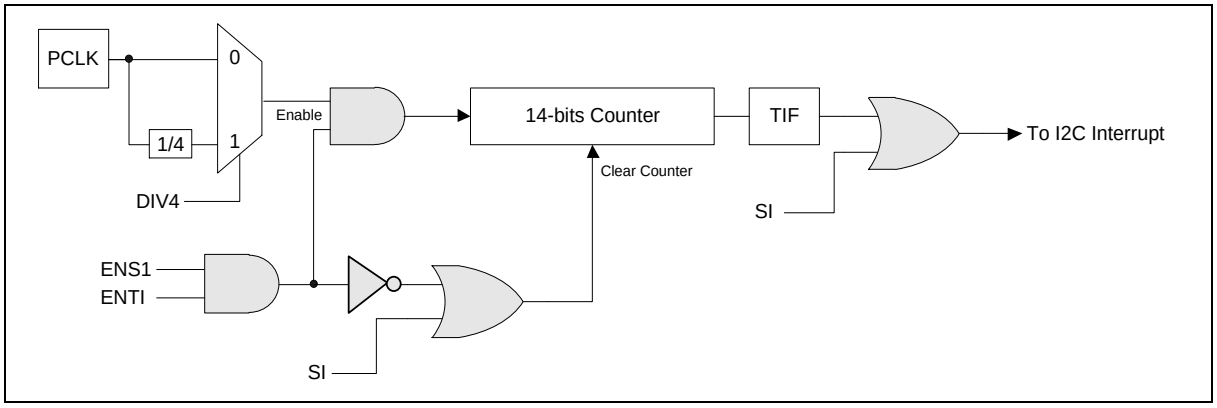


图 5-110 I²C 超时计数模块框图

唤醒控制寄存器(I2CWKUPCON)

当进入掉电模式，其他的I²C主机可以通过寻址I²C设备唤醒我们的芯片。用户必须在进入掉电模式之前设置。当芯片四个地址寄存器相匹配的其中一个唤醒，此时下一个数据将被作废。

唤醒状态寄存器(I2CWKUPSTS)

当系统被其他的I²C主机设备唤醒，WKUPIF(I2CWKUPSTS[0])被置位表示该事件发生。用户需要写“1”来清除此位。

5.16.6 EEPROM 随机读取例子

下面配置I²C0相关寄存器流程用于I²C从EEPROM读取数据。

1. 在“GPA_MFP”寄存器中将多功能管脚设置成 I2C0_SCL 和 I2C0_SDA 管脚。
2. 通过 I2C0_EN(APBCLK[8])使能 I2C APB 时钟。
3. 通过设置I2C0_RST (IPRSTC2 [8]) = 1复位I²C控制器，然后通过I2C0_RST(IPRSTC2 [8]) = 0将I²C控制器设置成普通操作。
4. 通过设置ENS1(I2CON[6])=1使能I²C0控制器。
5. .通过在I2CLK寄存器写 I²C时钟分频值。

6. 在“NVIC_ISER”寄存器中设置SETENA(NVIC_ISER[31:0])=0x00040000来设置I²C0 IRQ.
7. 设置EI(I2CON[7])=1使能I²C0中断
8. 设置I²C0地址寄存器“I2CADDR0~I2CADDR3”。

随机读操作是访问EEPROM的其中一种方法。这个方法是允许主机访问EEPROM的任何一个地址。下图显示EEPROM随机读取操作。

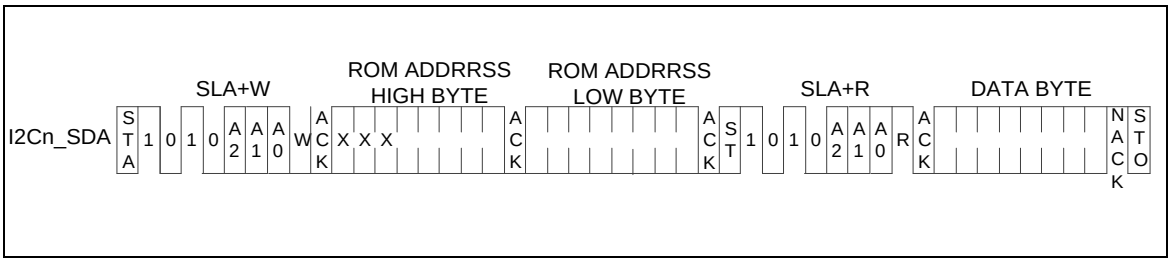


图 5-111 EEPROM 随机读

下图显示如何使用I²C控制器来实现EEPROM随机读取协议。

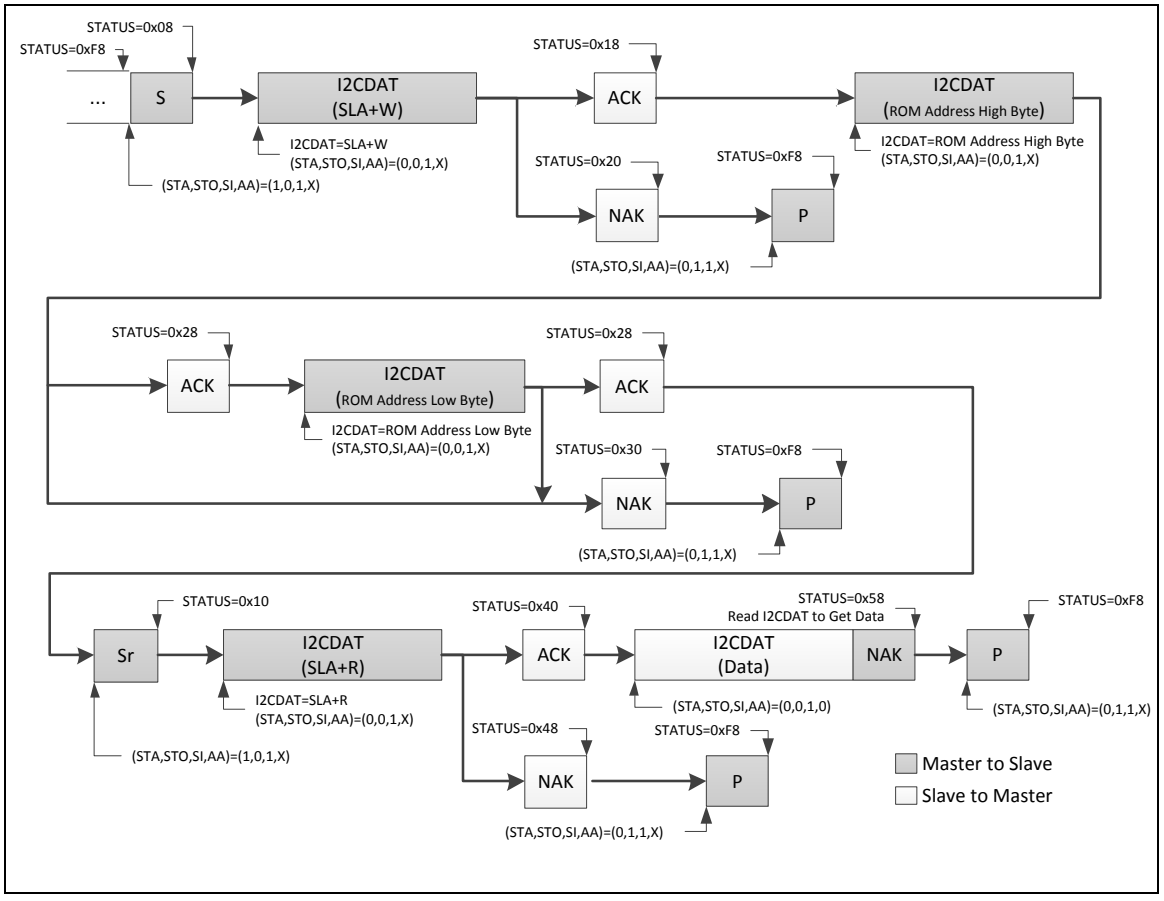


图 5-112 EEPROM 随机读协议

I²C控制器作为主机发送起始信号到总线，然后发送SLA+W (从机地址 + 写 位)到EERPOM，跟着由两个字节数据地址来设置EEPROM被读的地址。最后，重复起始信号跟着SLA+R被发送来向EEPROM 读取数据。

5.16.7 寄存器映射

R: 只读, **W:** 只写, **R/W:**读/写

寄存器	偏移地址	R/W	描述	复位值
I²C 基地址: **I2C0_BA = 0x4008_0000** **I2C1_BA = 0x4008_1000**				
I2CON **n=0,1**	I2Cn_BA+0x00	R/W	I²C控制寄存器	0x0000_0000
I2CADDR0 **n=0,1**	I2Cn_BA+0x04	R/W	I²C从机地址寄存器0	0x0000_0000
I2CDAT **n=0,1**	I2Cn_BA+0x08	R/W	I²C数据寄存器	0x0000_0000
I2CSTATUS **n=0,1**	I2Cn_BA+0x0C	R	I²C状态寄存器	0x0000_00F8
I2CLK **n=0,1**	I2Cn_BA+0x10	R/W	I²C时钟分频寄存器	0x0000_0000
I2CTOC **n=0,1**	I2Cn_BA+0x14	R/W	I²C超时控制寄存器	0x0000_0000
I2CADDR1 **n=0,1**	I2Cn_BA+0x18	R/W	I²C从机地址寄存器1	0x0000_0000
I2CADDR2 **n=0,1**	I2Cn_BA+0x1C	R/W	I²C从机地址寄存器2	0x0000_0000
I2CADDR3 **n=0,1**	I2Cn_BA+0x20	R/W	I²C从机地址寄存器3	0x0000_0000
I2CADM0 **n=0,1**	I2Cn_BA+0x24	R/W	I²C从机地址掩码寄存器0	0x0000_0000
I2CADM1 **n=0,1**	I2Cn_BA+0x28	R/W	I²C从机地址掩码寄存器1	0x0000_0000
I2CADM2 **n=0,1**	I2Cn_BA+0x2C	R/W	I²C从机地址掩码寄存器2	0x0000_0000
I2CADM3 **n=0,1**	I2Cn_BA+0x30	R/W	I²C从机地址掩码寄存器3	0x0000_0000
I2CWKUPCON **n=0,1**	I2Cn_BA+0x3C	R/W	I²C 唤醒控制寄存器	0x0000_0000
I2CWKUPSTS **n=0,1**	I2Cn_BA+0x40	R/W	I²C唤醒状态寄存器	0x0000_0000

5.16.8 寄存器描述

I²C 控制寄存器(I2CON)

寄存器	偏移地址	R/W	描述	复位值
I2CON n=0,1	I2Cn_BA+0x00	R/W	I²C 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
EI	ENS1	STA	STO	SI	AA	保留	

位	描述	
[31:8]	保留	保留.
[7]	EI	**使能中断** 1 = 使能 I²C 中断 0 = 禁用 I²C 中断
[6]	ENS1	**I²C 控制器使能位** 1 = 使能 0 = 禁用 设置使能 I²C 串行控制器功能。当 ENS1=1，I²C 串行功能使能。I2Cn_SDA和I2Cn_SCL对应的多功能管脚功能必须先设置为 I²C 功能。
[5]	STA	**I²C起始控制位** 设置 STA 为 1，进入主机模式，如果总线处于空闲状态，I²C 硬件会送出 START 或 重复 START 条件。
[4]	STO	**I²C 停止控制位** 在主机模式，设置 STO 来传送一个 STOP 条件到总线，然后 I²C 硬件将会检查总线状况，如果检测到一个 STOP 状况，这个标志会被硬件自动清除。在 I²C 从机模式，设置 STO 复位 I²C 硬件来定义“无地址”从机模式，这表示在从机接收模式不再接收从主机设备发送的数据。
[3]	SI	**I²C 中断标志** 当一个新的 I²C 状态出现在寄存器 I2CSTATUS 时，SI 标志由硬件置位，并且如果 EI (I2CON [7]) 位被置位，则产生 I²C 中断请求。SI 必须由软件通过向该位写 ’1’ 清零。
[2]	AA	**应答控制位** 当 AA=1 先于地址或数据接收，在以下两种情况：1.) 从机正在应答主机发送的地址。2.) 接收设备正在应答发送设备发送的数据，在I2Cn_SCL线上的应答时钟脉冲期间将返回一个应答（I2Cn_SDA上的低电平）。当 AA = 0 先于地址或数据接收，则在I2Cn_SCL线上的应答时

		钟脉冲期间将返回一个非应答（I2Cn_SDA上的高电平）。
[1:0]	**Reserved**	保留.

I²C数据寄存器 (I2CDAT)

寄存器	偏移地址	R/W	描述	复位值
I2CDAT **n=0,1**	I2Cn_BA+0x08	R/W	I²C数据寄存器	0x0000_0000

31	30	29	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**
I2CDAT[7:0]							

位	描述	
[31:8]	**Reserved**	保留.
[7:0]	**I2CDAT**	**I²C 数据寄存器** 该区域 8 位 I²C 串行端口传输数据

I²C 状态寄存器 (I2CSTATUS)

寄存器	偏移地址	R/W	描述	复位值
I2CSTATUS n=0,1	I2Cn_BA+0x0C	R	I²C状态寄存器	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**
I2CSTATUS[7:0]							

位	描述	
[31:8]	**Reserved**	保留.
[7:0]	**I2CSTATUS**	**I²C 状态寄存器** 共有26个可能状态码。 当I2CSTATUS值是0xF8没有串行中断请求。 所有其他的 I2CSTATUS的值对应 I²C 的状态。当进入其中任一状态时，就会产生状态中断请求 (SI(I2CON[3]) = 1)。在 SI 被硬件置位或 SI 被软件复位后一个机器周期，有效状态码出现在 I2CSTATUS中。 此外，00H 状态表示总线错误。总线错误发生在 START 或 STOP 条件出现在帧结构不正确的位置。不正确的位置比如是在串行传输地址字节、数据字节或应答位期间。

I²C 时钟分频寄存器(I2CLK)

寄存器	偏移地址	R/W	描述	复位值
I2CLK **n=0,1**	I2Cn_BA+0x10	R/W	I²C时钟分频寄存器	0x0000_0000

31	30	29	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**
I2CLK[7:0]							

位	描述	
[31:8]	**Reserved**	保留
[7:0]	**I2CLK**	**I²C 时钟分频寄存器** I²C 数据波特率 = (system clock) / (4x (I2CLK+1)). **注意:** I2CLK 的最小值是 4.

I²C 超时计数器寄存器 (I2CTOC)

寄存器	偏移地址	R/W	描述	复位值
I2CTOC n=0,1	I2Cn_BA+0x14	R/W	I²C超时计数器寄存器	0x0000_0000

31	30	29	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					**ENTI**	**DIV4**	**TIF**

位	描述	
[31:3]	**Reserved**	保留.
[2]	**ENTI**	**超时计数器使能/禁用** 1 = 使能 0 = 禁用 当使能该位，则14-位溢出定时器将会在 SI(I2CON[3]) 清零后开始计数。设置 SI(I2CON[3]) 位为高将会复位计数器，在 SI(I2CON[3])位清零之后，计数器会重新开始计数。
[1]	**DIV4**	**溢出定时器输入时钟除以 4** 1 = 使能 0 = 禁用 该位使能后，超时时间扩大 4 倍。
[0]	**TIF**	**定时溢出标志** 当超时发生时，该位由 H/W 置位，如果此时 I²C 的中断使能位 EI(I2CON[7]) 置为1，则可引发 CPU 的中断。 **注意**：写 1 清除该位。

I²C 从机地址寄存器 (I2CADDRx)

寄存器	偏移地址	R/W	描述	复位值
I2CADDR0 n=0,1	I2Cn_BA+0x04	R/W	I²C 从机地址寄存器0	0x0000_0000
I2CADDR1 n=0,1	I2Cn_BA+0x18	R/W	I²C 从机地址寄存器1	0x0000_0000
I2CADDR2 n=0,1	I2Cn_BA+0x1C	R/W	I²C 从机地址寄存器2	0x0000_0000
I2CADDR3 n=0,1	I2Cn_BA+0x20	R/W	I²C 从机地址寄存器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**
I2CADDR[7:1]							**GC**

位	描述	
[31:8]	**Reserved**	保留
[7:1]	**I2CADDR**	**I²C 地址寄存器** 主机模式下，该寄存器内容没有意义。从机模式下，高七位作为芯片本身的地址。如果地址符合，I²C 硬件将会自动应答。
[0]	**GC**	**广播呼叫功能** 0 = 禁用广播呼叫功能 1 = 使能广播呼叫功能

I²C 从机地址掩码寄存器 (I2CADMx)

寄存器	偏移地址	R/W	描述	复位值
I2CADM0 n=0,1	I2Cn_BA+0x24	R/W	I²C 从机地址掩码寄存器0	0x0000_0000
I2CADM1 n=0,1	I2Cn_BA+0x28	R/W	I²C 从机地址掩码寄存器1	0x0000_0000
I2CADM2 n=0,1	I2Cn_BA+0x2C	R/W	I²C 从机地址掩码寄存器2	0x0000_0000
I2CADM3 n=0,1	I2Cn_BA+0x30	R/W	I²C 从机地址掩码寄存器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**
I2CADM[7:1]							**Reserved**

位	描述	
[31:8]	**Reserved**	保留
[7:1]	**I2CADM**	**I²C 地址掩码寄存器** 1 = 使能掩码（接收到的相应地址位不予辨识） 0 = 禁用掩码（接收到的相应地址必须完全符合地址寄存器） I²C 总线控制器有4个地址掩码寄存器，支持多地址识别。当地址掩码寄存器的某位被置 '1'，表示接收到的地址的相应位可忽略。如果该位被置 '0'，则表示接收到的地址的相应位必须和地址寄存器中的完全一致。
[0]	**Reserved**	保留.

I²C唤醒控制 寄存器 (I2CWKUPCON)

寄存器	偏移地址	R/W	描述	复位值
I2CWKUPCON n=0,1	I2Cn_BA+0x3C	R/W	I²C唤醒控制寄存器	0x0000_0000

31	30	29	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							**WKUPEN**

位	描述	
[31:1]	**Reserved**	保留.
[0]	**WKUPEN**	**I²C唤醒功能使能** 1 = I²C 唤醒功能使能. 0 = I²C 唤醒功能禁止.

I²C 唤醒状态寄存器 (I2CWKUPSTS)

寄存器	偏移地址	R/W	描述	复位值
I2CWKUPSTS n=0,1	I2Cn_BA+0x40	R/W	I²C唤醒状态寄存器	0x0000_0000

31	30	29	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							**WKUPIF**

位	描述	
[31:1]	**Reserved**	保留.
[0]	**WKUPIF**	**I2C唤醒标志** 0 =芯片不是在掉电模式下由I2C中断唤醒 1 =芯片是在掉电模式下由I2C中断唤醒 **注意：**该位软件写1清0

5.17 串行外围设备接口 (SPI)

5.17.1 概述

串行外围设备接口(SPI)是一个工作于全双工模式的同步串行数据通讯协议。设备可工作在主/从模式，利用4线双向接口相互通讯。NuMirco NUC200系列包含4组SPI控制器，当从一个外围设备接收数据时，SPI执行串-并的转换，而在数据向外围设备发送时执行并-串的转换。每组SPI控制器可以配置为主设备或从设备。

SPI控制器支持可变串行时钟以适应特殊的应用，也支持2位传输模式，可同时连接两个片外从机设备。SPI控制器也支持使用PDMA功能访问数据缓冲区和也支持双I/O传输模式。

5.17.2 特性

- 多达4组SPI控制器
- 支持主机和从机工作模式
- 支持2位传输模式
- 支持双I/O传输模式
- 一个事务传输的数据长度可配置为8到32位
- 提供独立的8级深度发送和接收FIFO缓存
- 支持MSB或LSB优先传输
- 主机模式下支持2条从机选择线
- 支持字节重排序功能
- 支持字节或者字休眠模式
- 在主机模式下，支持可变串行时钟频率
- 支持PDMA传输
- 支持三线，没有从机选择信号的双向接口

5.17.3 框图

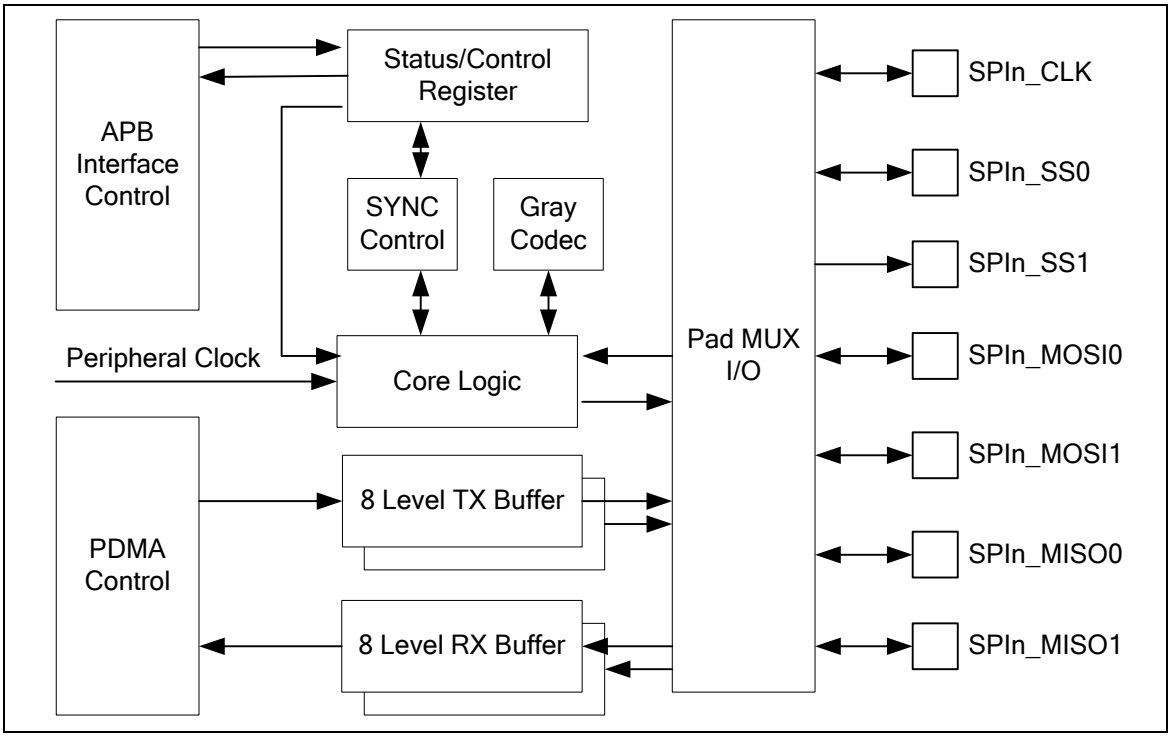


图 5-113 SPI 框图

5.17.4 基本配置

SPI0的基本配置如下：

- SPI0管脚功能由寄存器ALT_MFP，GPB_MFP和GPC_MFP配置。
- 通过寄存器CLKSEL1的第4位SPI0_S选择SPI0外设时钟源
- 通过寄存器APBCLK的第12位SPI0_EN使能SPI0的外设时钟
- 通过寄存器IPRSC2的第12位SPI0_RST复位SPI0控制器

SPI1的基本配置如下：

- SPI1管脚功能由寄存器ALT_MFP，GPB_MFP和GPC_MFP配置。
- 通过寄存器CLKSEL1的第5位SPI1_S选择SPI1外设时钟源
- 通过寄存器APBCLK的第13位SPI1_EN使能SPI1的外设时钟
- 通过寄存器IPRSC2的第13位SPI0_RST复位SPI1控制器

SPI2的基本配置如下：

- SPI2管脚功能由寄存器ALT_MFP，ALT_MFP1，GPA_MFP和GPD_MFP配置。
- 通过寄存器CLKSEL1的第6位SPI2_S选择SPI2外设时钟源
- 通过寄存器APBCLK的第14位SPI2_EN使能SPI2的外设时钟
- 通过寄存器IPRSC2的第14位SPI2_RST复位SPI2控制器

SPI3的基本配置如下：

- SPI3管脚功能由寄存器ALT_MFP， GPB_MFP和GPD_MFP配置。
- 通过寄存器CLKSEL1的第7位SPI3_S选择SPI3外设时钟源
- 通过寄存器APBCLK的第15位SPI3_EN使能SPI3的外设时钟
- 通过寄存器IPRSC2的第15位SPI3_RST复位SPI3控制器

5.17.5 功能描述

5.17.5.1 *术语*

SPI外设时钟和SPI总线时钟

SPI控制器需要SPI外设时钟来驱动SPI逻辑单元执行数据传输。SPI总线时钟是SPIn_CLK管脚上的时钟。

SPI外设时钟速率决定于时钟源，BCn选项和时钟分频器的设置。寄存器CLKSEL1的SPIn_S位决定SPI外设时钟的时钟源。时钟源可以是HCLK或是PLL输出时钟。设置寄存器SPI_CNTRL2的BCn位为0，可兼容早先产品的SPI时钟速率计算。寄存器SPI_DIVIDER的0到7位DIVIDER的设定值决定时钟速率计算时的分频值。

在主机模式下，如果可变时钟功能禁止，SPI总线时钟输出管脚的输出频率等于SPI外设时钟速率。在通常情况下，SPI总线时钟表示为SPI时钟。在从机模式下，SPI总线时钟由片外主机设备提供。从机设备的SPI外设时钟速率必须快于连接在一起的主机设备的SPI总线时钟速率。不论工作在主机还是从机模式，SPI外设的时钟频率不能快于APB时钟速率。

主机/从机模式

SPI控制器可通过设置寄存器SPI_CNTRL的第18位SLAVE配置成主机或从机模式，来与片外SPI从机或者主机通信。主机模式与从机模式的应用框图如下：

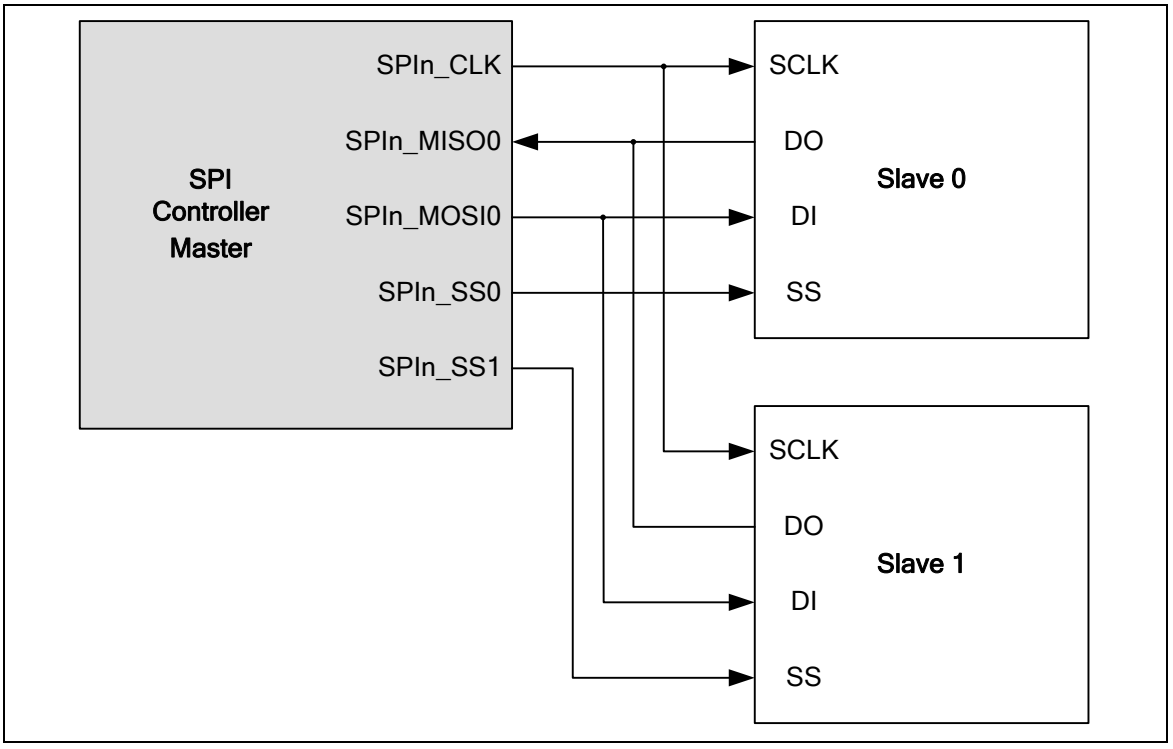


图 5-114 SPI 主机模式应用框图

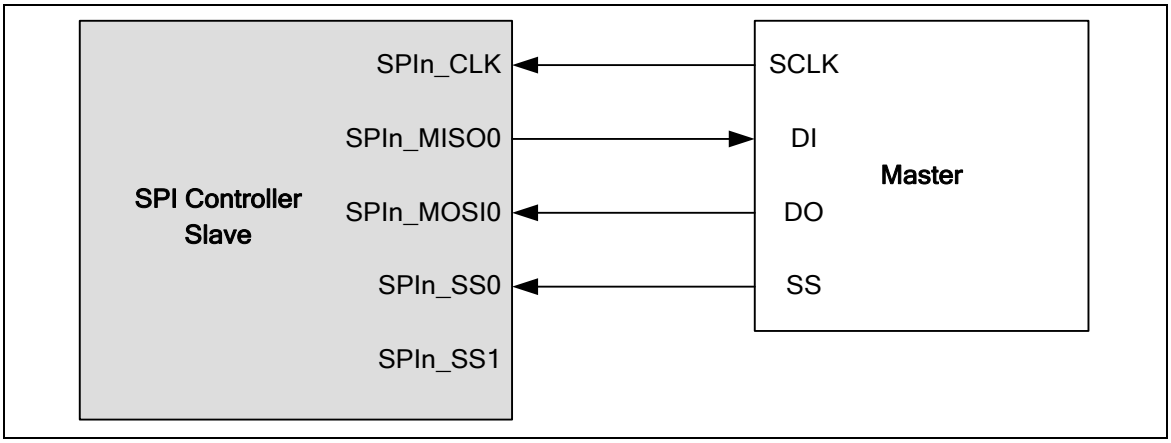


图 5-115 SPI 丛机模式应用框图

时钟极性

寄存器SPI_CTL的第11位CLKP定义总线时钟的空闲状态。如果CLKP=1，SPIn_CLK输出为空闲高电平状态，否则如果CLKP=0，SPIn_CLK输出为空闲低电平状态。

发送/接收位长

一个事务的位长由寄存器SPI_CNTRL的第3到7位TX_BIT_LEN来定义。对于发送和接收，在一个事务中，位长可配置为多达32位。

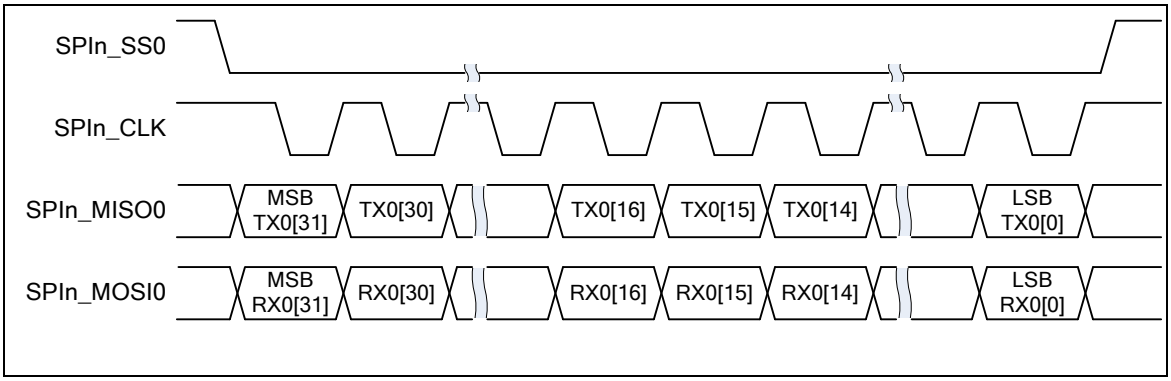


图 5-116 一次传输32-位长度

LSB/MSB 优先

寄存器SPI_CNTRL的第10位LSB定义一个事务中位传输序列。如果设定LSB位为1，传输序列为LSB优先。位0首先将被传输。如果清除LSB位为0，传输序列是MSB优先。

发送边沿

寄存器SPI_CNTRL的第2位TX_NEG定义数据在SPI总线时钟的下降沿还是上升沿被发送出去。

接收边沿

寄存器SPI_CNTRL的第1位RX_NEG定义数据在SPI总线时钟的下降沿还是上升沿被接收。

注意: TX_NEG和RX_NEG的设置是相互排斥的，换话说就是不要在相同的时钟沿发送和接收数据。

字休眠

在主机模式下，寄存器SPI_CNTRL的第12到15位SP_CYCLE提供一个在两个连续事务之间可配置为0.5～15.5个SPI时钟周期的休眠间隔。休眠间隔指的是从前一个事务的最后一个时钟沿到下一个传输事务的第一时钟沿。SP_CYCLE的默认值是0x3(3.5 SPI总线时钟周期)。如果软件禁止FIFO模式，该SP_CYCLE设定值对字休眠间隔不起作用。

如果寄存器SPI_CNTRL的23位VARCLK_EN和寄存器SPI_CNTRL的第21位FIFO都设置为1，最小字休眠间隔是(6.5+SP_CYCLE)*SPI 时钟周期。

从机选择

在主机模式下，该SPI控制器能通过从机选择输出脚SPIn_SS0和SPIn_SS1来驱动多达两个片外从机设备。在从机模式，片外主机设备通过SPIn_SS0输入端口驱动从机选择信号到SPI控制器。在主机和从机模式下，从机选择信号的有效状态可以通过编程寄存器SPI_SSR的第2位SS_LVL来设定为低有效或高有效，寄存器SPI_SSR的第4位SS_LTRIG来设定从机选择信号SPIn_SS0/1为电平触发还是边沿触发。触发条件的选择取决于所连设备的从机/主机的类型。

在从机模式下，如果SS_LTRIG位被配置成电平触发，则寄存器SPI_SSR的第5位LTRIG_FLAG用来表示在一个事务中接收到的数据位是否满足寄存器SPI_CNTRL的第3到7位TX_BIT_LEN的设定值。

电平触发/边沿触发

在从机模式下，从机选择信号可以被配置为电平触发或边沿触发。边沿触发模式，数据从一个有效的从机选择信号边沿开始，到一个无效的从机选择信号边沿结束。当检测到一个无效边沿，寄存器SPI_CNTRL的16位单元传输中断标志将设置为1。如果主机没有发送一个无效边沿给从机，传输过程不会完成，从机的单元传输中断标志不会被置位。电平触发模式，当有下面的两个条件中的一个发生，从机的单元传输中断标志将会被置位。第一条件是传输的数据位与TX_BIT_LEN的设定值相匹配时，从机的单元传输中断标志将会被置位。同样第二个条件，如果主机正在传输期间的时候设置从机选择管脚为无效电平，将会强制从机设备终结当前传输，而不管已经传输了多少位数据，从机的单元传输中断标志将会被置位。用户可以读取LTRIG_FLAG状态位来检查数据是否已经全部传完。

5.17.5.2 *自动从机选择*

在主机模式下，如果寄存器SPI_SSR的第3位AUTOSS设置为1，从机选择信号将会自动产生，并根据寄存器SPI_SSR的第0位SSR[0]和第1位SSR[1]是否使能，将从机选择信号输出到SPInSS0与SPInSS1管脚上。这意味着当通过设置寄存器SPI_CNTRL的0位GO_BUSY来开始数据传输时，在寄存器SPI_SSR的0到1位中使能的从机选择信号将由SPI控制器自动设置为有效状态，在数据传输结束后自动被设置为无效状态。如果AUTOSS被清零时，从机选择输出信号需要通过手工置位/清除寄存器SPI_SSR的0到1位的相关位，从而使从机进入激活或非激活状态。从机选择输出信号的激活电平状态由寄存器SPI_SSR的第2位SS_LVL来定义。

在主机模式下，如果寄存器SP_CYCLE的0到3位的值小于3且AUTOSS被设置为1，在连个连续的事

务之间，从机选择信号将保持有效。

在从机模式下，为了识别从机选择信号的无效状态，在两个连续的事务之间，从机选择信号无效状态持续时间必须大于或等于6个外设时钟周期。

5.17.5.3 *可变总线时钟频率*

在主机模式下，如果寄存器SPI_CNTRL的第23位VARCLK_EN被设置为1，SPI时钟输出可以被编程为可变频率模式。每个SPI时钟周期的长度取决于SPI_VARCLK寄存器的设定值。当可变时钟功能使能，TX_BIT_LEN必须设置为0x10来配置数据传输为16位传输模式。寄存器VARCLK的31位决定第一个时钟周期的长度。如果寄存器VARCLK的31位值为0，第一个时钟周期的长度取决于寄存器DIVIDER的设定值；如果值为1，一个时钟周期的长度取决于寄存器DIVIDER2的设定值.在寄存器VARCLK的1到30位中两个连续的位定义一个时钟周期。如果两个连续位的值为00，时钟周期取决于寄存器DIVIDER的设置；如果值是11，时钟周期取决于寄存器DIVIDER2的设置。位域寄存器VARCLK[30:29]定义一个事务中SPI时钟的第二个周期，位域寄存器VARCLK[28:27]定义第三个周期，等等。寄存器VARCLK[0]无意义。SPI总线时钟，寄存器VARCLK设置，寄存器DIVIDER设置和寄存器DIVIDER2设置之间的时序关系如下图所示。

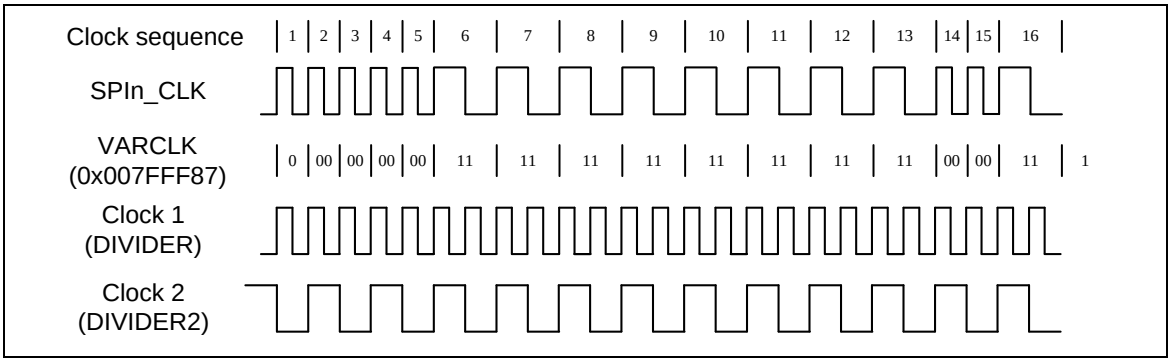


图 5-117 可变总线时钟频率

5.17.5.4 *字节重排序功能*

当设置为 MSB 优先 (LSB = 0) 且使能 REORDER，在TX_BIT_LEN =0， 32-位模式时，存储在 TX 缓存与 RX 缓存的数据将重新按 [BYTE0, BYTE1, BYTE2, BYTE3] 的顺序排列，数据将以 BYTE0, BYTE1, BYTE2, 和 BYTE3 的顺序进行发送/接收。如果 TX_BIT_LEN 设为 24-位模式，存储在 TX 缓存与 RX 缓存的数据将重新按 [unknown byte, BYTE0, BYTE1, BYTE2] 的顺序排列。SPI 控制器将按照 BYTE0, BYTE1, BYTE2 的顺序发送/接收数据，每个字节 MSB 优先发送/接收。16-位模式的规则与上面相同。字节重排序功能只在 TX_BIT_LEN 为16, 24, and 32 位时适用。

注意: 当可变串行时钟功能使能，不支持重排序功能。

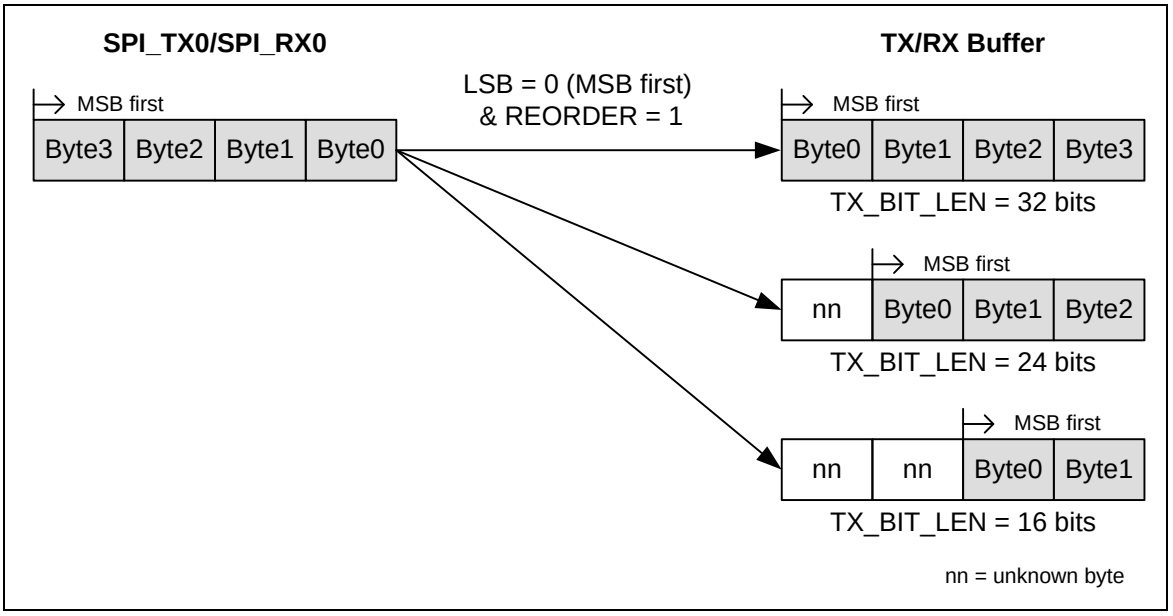


图 5-118 字节重排序功能

5.17.5.5 *字节休眠功能*

主机模式下，如果 SPI_CNTRL[19] 被设为 1，硬件将会在两个连续传输字节之间插入一个0.5 ~ 15.5个串行时钟周期的休眠间隔。字节休眠的设置与字休眠的设置一样都是用 SP_CYCLE(寄存器SPI_CONTRL[15:12])。

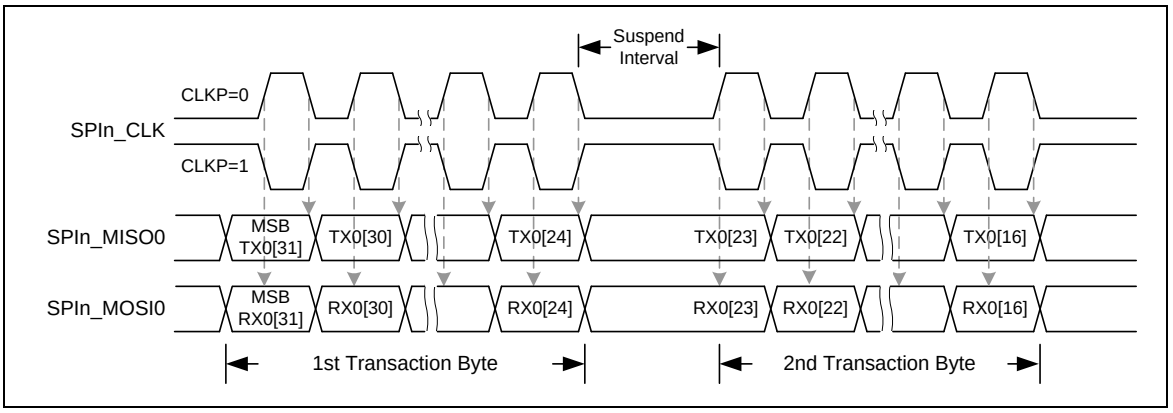


图 5-119 字节休眠时序波形

5.17.5.6 *从机3-线模式*

当NOSLVSEL(SPI_CNTRL2[8])位软件置1使能从机3线模式时，在从机模式下SPI控制器可以工作在没有从机选择信号。NOSLVSEL位仅在从机模式有效。在与一个主机通讯时，仅需要三个管脚，SPICLK, SPI_MISO, 和 SPI_MOSI。SPISS脚可以配置成一个GPIO。当 NOSLVSEL 位被设为 1 时，在 GO_BUSY 位被置为1后SPI从机将开始准备发送和接收数据。当接收数据位的个数满足定义长度TX_BIT_LEN(SPI_CNTRL[7:3])时，单元传输中断标志IF(SPI_CNTRL[16])将会置位。

注意:在从机3-线模式，SS_LTRIG, SPI_SSR[4]，必须设为 1。

5.17.5.7 *2-位传输模式*

当设置 TWOB 位 (SPI_CNTRL[22]) 为1时，SPI 控制器支持两-位 (two-bit) 传输模式。在2-位模式，SPI控制器执行全双工数据传输，换句话说，可以同时发送和接收两-位串行数据。

例如，在主机模式下，保存在寄存器 SPI_TX0 和寄存器 SPI_TX1 中的数据将会分别从 MOSIx0 管脚和 MOSIx1 管脚发送。同时，寄存器 SPI_RX0 和寄存器 SPI_RX1 将会分别保存从 MISOx0 管脚和 MISOx1 管脚接收到数据。

在从机模式下，保存在寄存器 SPI_TX0 和寄存器 SPI_TX1 中的数据将会分别从 MISOx0 管脚和 MISOx1 管脚发送。同时，寄存器 SPI_RX0 和寄存器 SPI_RX1 将会分别保存从 MOSIx0 管脚和 MOSIx1 管脚接收到数据。

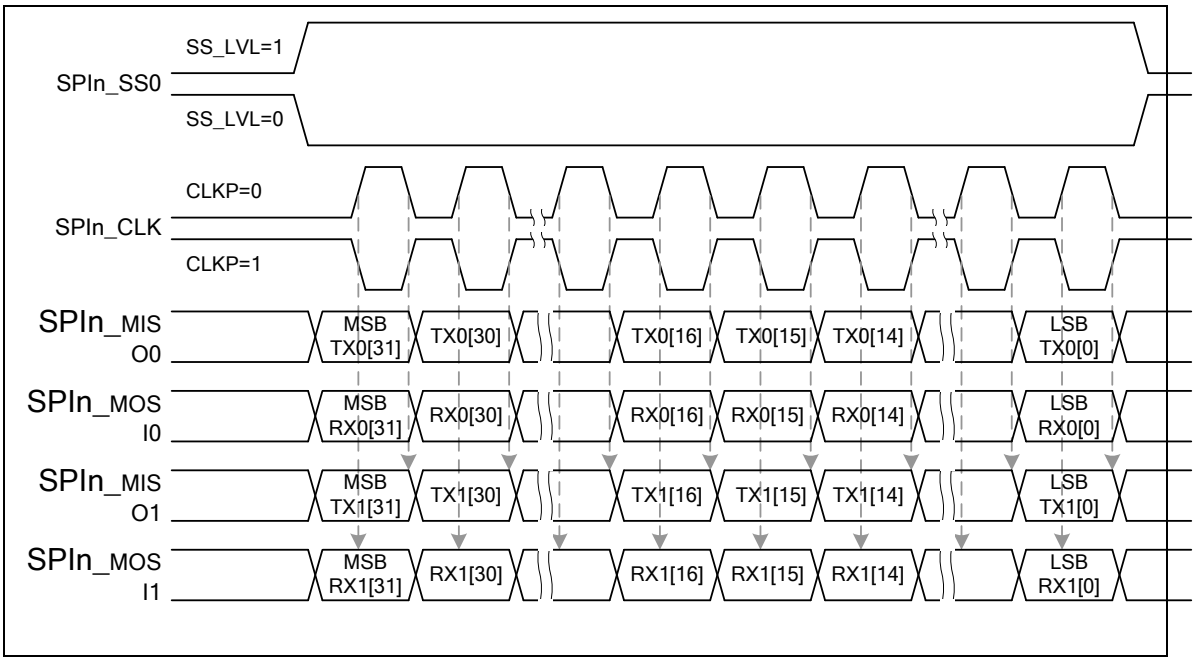


图 5-120 2-bit 传输模式(从模式)

5.17.5.8 *双 I/O 模式*

当设置DUAL_IO_EN bit (SPI_CNTRL2[13]) 为 1，SPI控制器支持双I/O传输。许多通用的SPI flash支持双I/O传输。DUAL_IO_DIR bit (SPI_CNTRL2[12])用来定义传输数据的方向。当DUAL_IO_DIR bit 设置为1，控制器会发送数据到外部设备。当DUAL_IO_DIR bit 设置为0，控制器会从外部设备读数据。该功能支持8, 16, 24, 和 32-位长度。

当从机3线模式或字节重排序功能使能，不支持双I/O模式。

如果DUAL_IO_EN 和 DUAL_IO_DIR都设置成1，SPI_MOSI0是偶数位数据输出，SPI_MISO0是奇数位数据输出。如果DUAL_IO_EN置1，DUAL_IO_DIR置0，SPI_MISO0 和 SPI_MOSI0都会设置为数据输入口。

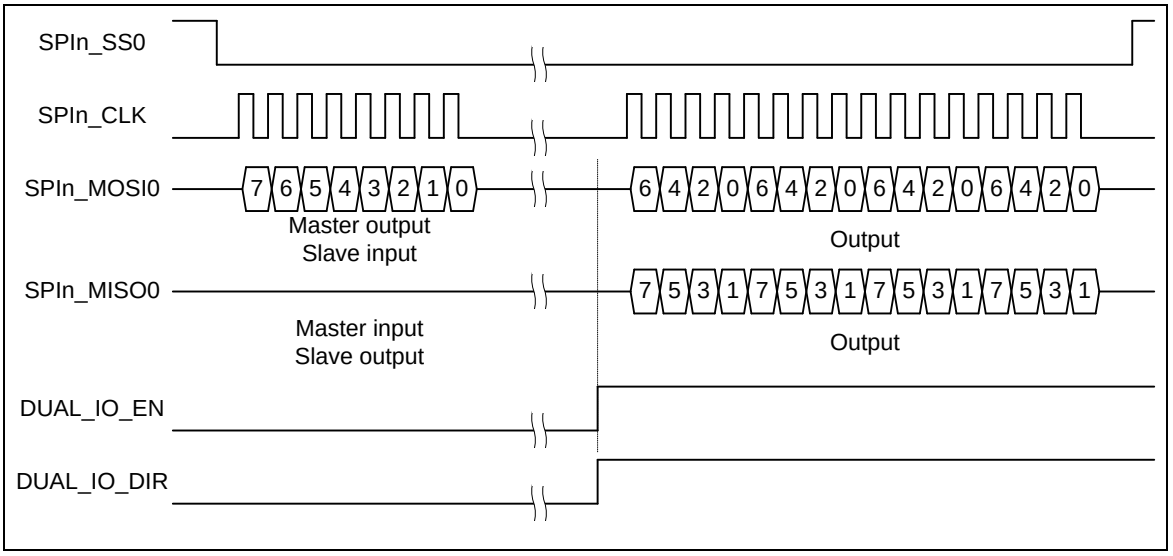


图 5-121 双输出模式的位序列

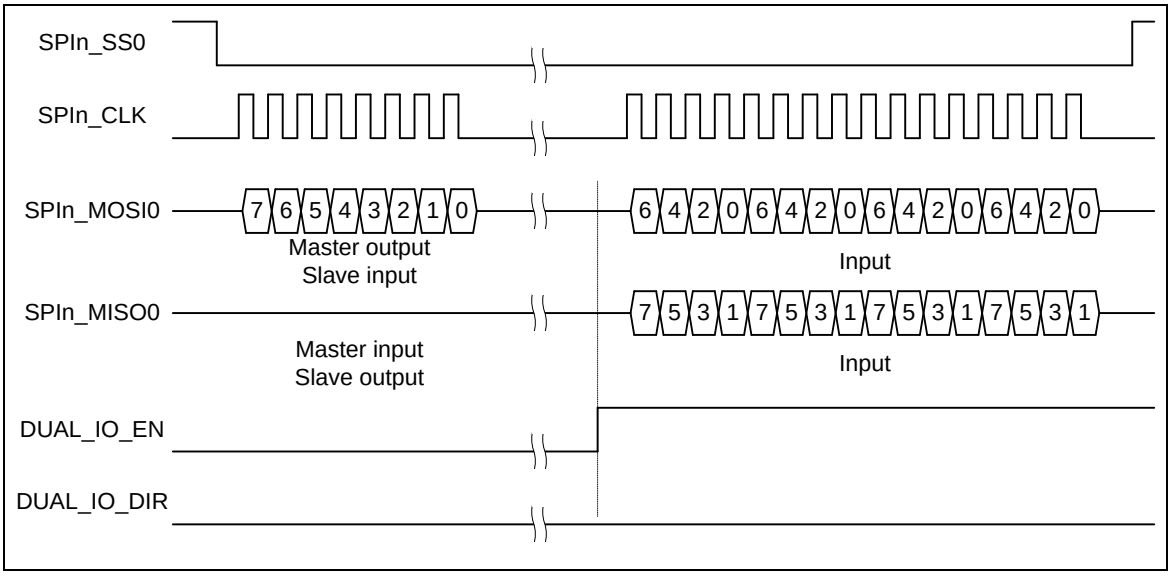


图 5-122 双输入模式的位序列

5.17.5.9 *FIFO 模式*

当 FIFO (SPI_CNTRL[21]) 被设为1时，SPI控制器支持FIFO模式。SPI控制器配备了32-bit宽的发送和接收FIFO缓存各8个

发送FIFO缓存是一个8层深度，32位宽，先进先出寄存器缓存。数据可以通过软件写SPI_TX0寄存器写入发送FIFO缓存。存储在发送FIFO缓存的数据会被传输控制逻辑读取并发送出去。如果8层发送FIFO缓存满了，TX_FULL位会被置1。当SPI传输逻辑单元抽出发送FIFO缓存的最后一个数据，8层发送FIFO缓存为空，TX_EMPTY位被置1。注意TX_EMPTY标志被置1时，最后一笔传输还在进行。在主机模式，软件应该检查GO_BUSY 位和 TX_EMPTY 位确认SPI是否空闲。

接收FIFO缓存也是一个8层深度，32位宽，先进先出寄存器缓存。接收控制逻辑存储接收到的数据到该缓存。FIFO缓存数据可以通过软件从SPI_RX0 寄存器读取。有FIFO相关的状态位，像

RX_EMPTY 和 RX_FULL，来表明当前FIFO缓存的状态。

在FIFO模式，发送和接收阀值可以通过软件设置TX_THRESHOLD 和 RX_THRESHOLD来设定。当存储在发送FIFO缓存的有效数据计数少于或等于TX_THRESHOLD设定，TX_INTSTS位会被置1。当存储在接收FIFO缓存的有效数据计数大于RX_THRESHOLD设定，RX_INTSTS位会被置1。

在 FIFO模式，8个数据可以事先通过软件写入SPI发送FIFO缓存。当SPI控制器工作在FIFO模式，SPI_CNTRL 寄存器中的GO_BUSY bit由硬件控制，SPI_CNTRL 寄存器的内容不应该被软件修改，除非FIFO位被清0使FIFO模式禁止。

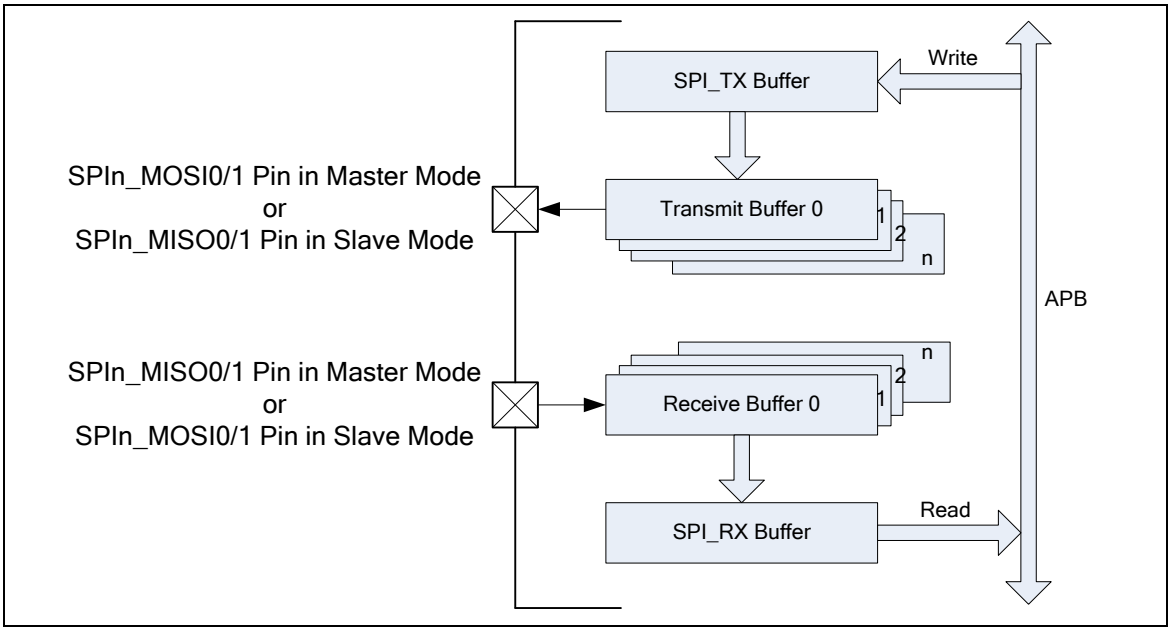


图5-123 FIFO模式框图

在主机发送操作中，当 FIFO 位被设置为 1，软件写第一个数据到 SPI_TX0 寄存器，TX_EMPTY 标志将会被清除为 0。只要发送 FIFO 缓存非空，发送操作立即开始，用户可以立即写接下来的数据到 SPI_TX0 寄存器。在 FIFO 模式下，SPI 控制器将会在两个连续的事务之间插入一个休眠间隔，休眠间隔的长度由SP_CYCLE (SPI_CNTRL [15:12]) 的设定值决定。只要 TX_FULL 标志为 0，用户就可以写数据到 SPI_TX0 寄存器。

如果要发送的数据更新及时，接下来的事务将会被自动触发。如果在所有数据传输完成之后，SPI_TX0 寄存器没有被更新，则传输停止。

在主机接收操作中，串行数据从 MISOx 管脚接收并被存储在接收 FIFO 缓存。当接收 FIFO 缓存中包含未读的数据时，RX_EMPTY 将会被清除为 0。只要RX_EMPTY 标志为 0，软件就可以从 SPI_RX0 寄存器读取接收到的数据。如果接收 FIFO 缓存包含 8 个未读数据， RX_FULL 标志将会被设置为1。此时，SPI控制器将会停止接收数据，直到软件读取SPI_RX0寄存器。

在从机模式下，当 FIFO 位被设置为 1，GO_BUSY 位将会被硬件自动设置为 1

在从机发送操作中，当软件写数据到 SPI_TX0 寄存器，数据将会被加载到发送 FIFO 缓存，且 TX_EMPTY 标志将会被设置为 0。当从设备从主机接收到时钟信号，发送操作将会开始。只要 TX_FULL 标志为 0，软件就可以写数据到 SPI_TX0 寄存器。在所有数据都被 SPI 发送逻辑单元发送出去，且软件没有更新 SPI_TX0 寄存器， TX_EMPTY 标志将会被设置为 1.

在从机接收操作中，串行数据从SPI_MOSI0/1管脚被接收，并被存储到 SPI_RX0 寄存器。接收机

制与主机模式接收操作类似。

5.17.5.10 *中断*

- SPI 单位传输中断

当SPI控制器完成一个单位传输，单位传输中断标志IF (SPI_CNTRL[16]) 将会被置位。如果中断使能位 IE (SPI_CNTRL[17]) 被置位，则单位传输中断事件将会给CPU产生中断。单位传输中断标志位只能写 1 清零。

- SPI 从机 3-线模式开始中断

在 3 线模式下，当从机检测到 SPI 时钟信号时，3 线模式会产生开始中断标志，SLV_START_INTSTS将会被置为 1。如果 SSTA_INTEN 被设置为 1，SPI控制器将会触发一个中断。如果接收到的数据位小于 TX_BIT_LEN 的设定要求，在由用户定义的期望的时间内再没有串行时钟输入，用户可以设置 SLV_ABORT 位来中止当前传输。如果软件设置 SLV_ABORT 位为1，单元传输中断标志，IF，将会被置位。

- 接收 FIFO 超时中断

在 FIFO 模式，有超时功能通知用户。如果超时中断使能位FIFO_CTL[21]置1，在FIFO里有一个接收到的数据，并且没有被软件读取主机模式下超过64个SPI引擎时钟周期，或从机模式下超过576个SPI引擎时钟周期，会发出一个超时中断。

- 发送 FIFO 中断

在FIFO模式，如果发送FIFO缓存的有效数据计数少于或等于TX_THRESHOLD的设定值，发送FIFO中断标志会被置1。如果SPI_FIFO_CTL[3]置1，发送FIFO中断使能，SPI控制器会产生一个发送FIFO中断到系统。

- 接收 FIFO 中断

在 FIFO 模式，如果接收FIFO缓存的有效数据计数大于RX_THRESHOLD的设定值，接收FIFO中断标志会被置1。如果SPI_FIFO_CTL[2]置1，接收FIFO中断使能，SPI控制器会产生一个接收FIFO中断到系统。

5.17.6 时序图

从机选择信号的有效状态可以由 SS_LVL 位 (SPI_SSR[2]) 和 SS_LTRIG 位 (SPI_SSR[4])的设定定义。串行时钟 (SPICLK) 的空闲状态可以通过 CLKP 位 (SPI_CNTRL[11]) 配置为高电平或低电平。传输字段长度在 TX_BIT_LEN (SPI_CNTRL[7:3])中定义，发送/接收数据是以 MSB 或 LSB 优先由 LSB 位 (SPI_CNTRL[10]) 定义。用户可以通过设置 TX_NEG/RX_NEG (SPI_CNTRL[2:1]) 寄存器来选择发送/接收数据时串行时钟的边沿。四个 SPI 发送/接收及相关设置如下。

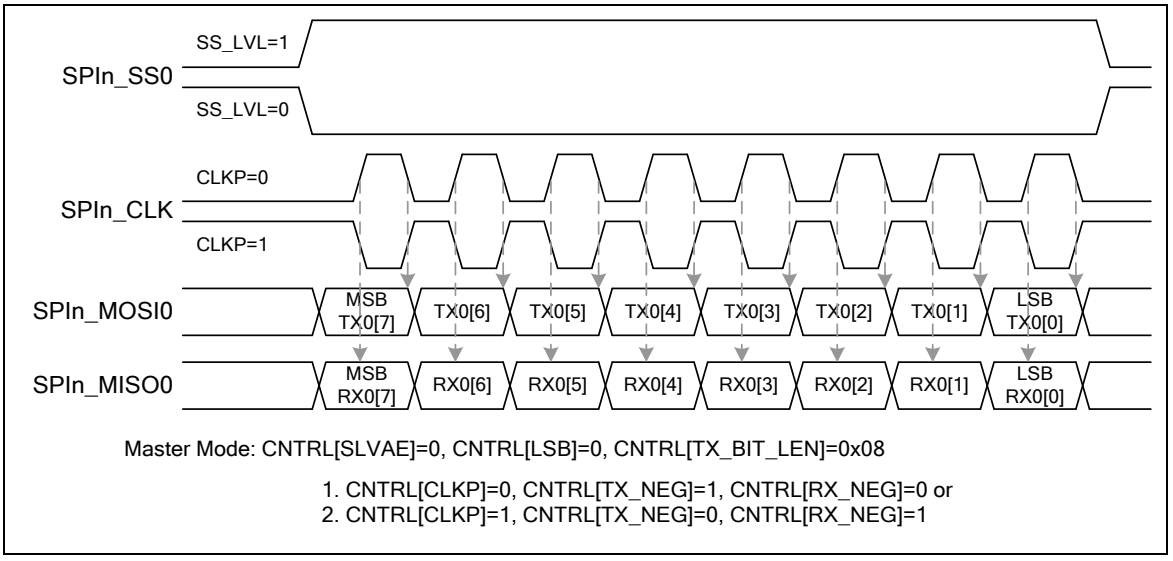


图 5-124SPI 主机模式下的时序

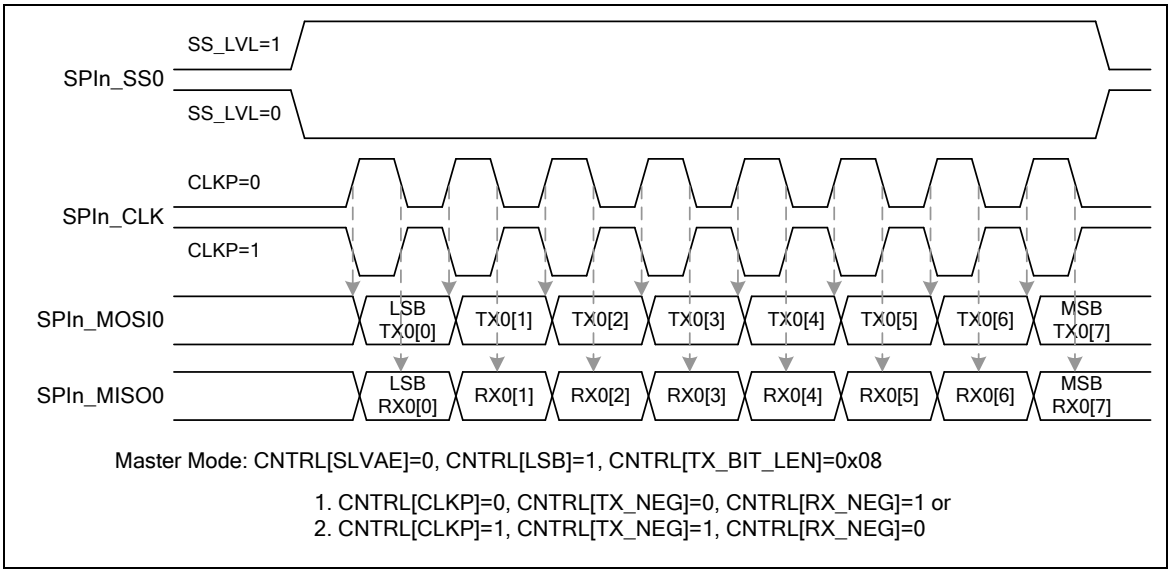


图 5-125 SPI 主机模式下的时序(交替 SPI 时钟相位)

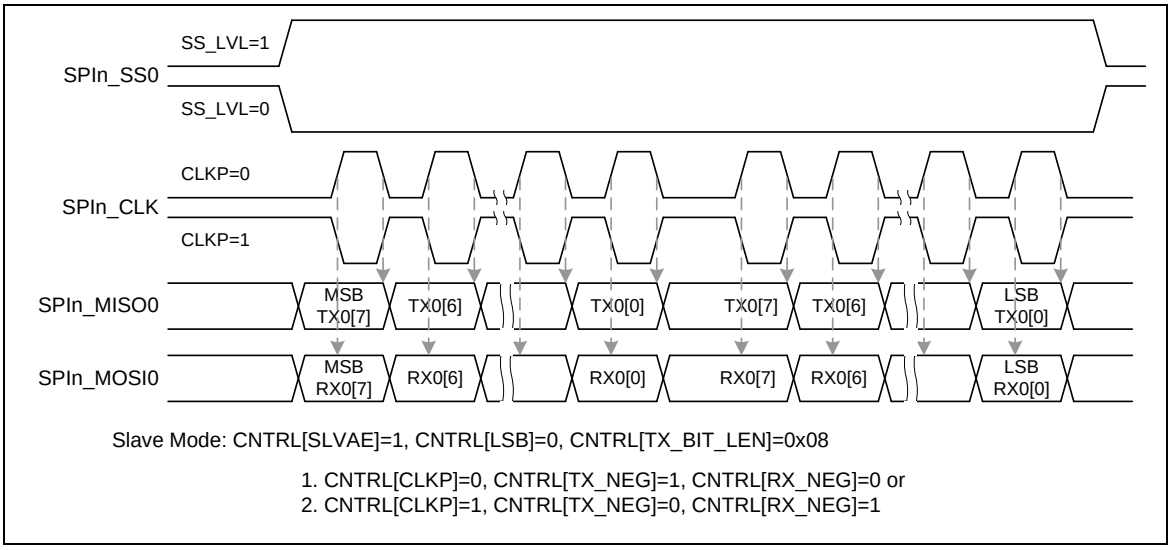


图 5-126 SPI 从机模式下的时序

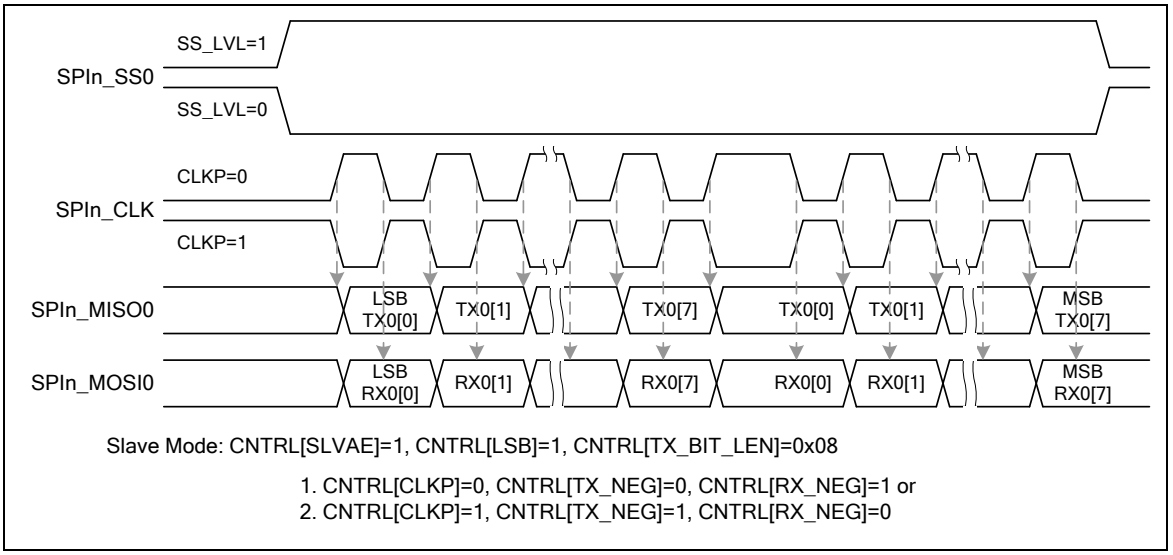


图 5-127SPI 从机模式下的时序(交替 SPI 时钟相位)

5.17.7 编程例子

例 1, SPI 控制器作为主机去访问一个片外从机设备，过程如下：

- 数据在串行时钟上升沿锁存
- 数据在串行时钟下降沿传输
- MSB 先传输
- SPICLK 空闲模式为低电平状态
- 每次发送/接收只有一个字节
- 使用第一个 SPI 从机选择管脚和一个片外从机相连。从机选择信号为低电平有效

操作流程如下：

1) 设置 DIVIDER (SPI_DIVIDER [7:0]) 寄存器来决定串行时钟输出频率。
2) 写一个合适的值到 SPI_SSR 寄存器来对主模式相关配置进行设定:
    1. 清除自动从选择位 AUTOSS (SPI_SSR[3] = 0).
    2. 在从机选择有效电平位 SS_LVL (SPI_SSR[2] = 0) 和从机选择电平触发位 SS_LTRIG (SPI_SSR[4] = 1)设定从机选择为低电平触发输出。
    3. 通过设定从机选择寄存器位SSR[0] (SPI_SSR[0]) 选择哪一个从机选择信号会在相应的 IO 管脚上输出，以激活片外从设备。
3) 通过设置 SPI_CNTRL 寄存器来控制 SPI 主机的行为：
    1. 通过设置 SLAVE 位 (SPI_CNTRL[18] = 0) 将 SPI 控制器设为主机设备。
    2. 通过设置 CLKP 位 (SPI_CNTRL[11] = 0) 将串行时钟的空闲状态设为低电平。.
    3. 通过设置 TX_NEG 位 (SPI_CNTRL[2] = 1) 选择数据在串行时钟的下降沿传输。
    4. 通过设置 RX_NEG 位 (SPI_CNTRL[1] = 0) 选择数据锁存在串行时钟的上升沿。.
    5. 通过设置 TX_BIT_LEN 位域 (SPI_CNTRL[7:3] = 0x08) 设定一次字传输的长度为8-位。
    6. 通过设置 MSB 位 (SPI_CNTRL[10] = 0) 设定为 MSB 传输优先。
4) 如果 SPI 主机要发送(写)一个字节的数据到片外从机设备，则将所要发送到数据写入SPI_TX0 寄存器。
5) 如果 SPI 主机只是要从片外从机设备接收（读）一个字节的数据，不必管被传输出去的数据是什么，寄存器SPI_TX0 不需要通过软件更新。
6) 使能 GO_BUSY 位 (SPI_CNTRL [0] = 1) 开始 SPI 接口的数据传输。
7) 等待 SPI 中断发生（如果中断使能位 IE 使能）或轮询检测 GO_BUSY 位直到被硬件自动清零。
8) 从寄存器 RX0 [7:0] (SPI_RX0[7:0]) 中读取接收到的一个字节数据。
9) 重复步骤 4) 继续其他数据传输或设置 SSR [0] 为 0 来停止片外从机设备。

例 2, SPI 控制器作为从机设备，和一片外主机设备相连，外设通过 SPI 接口与片上 SPI 从机通信。过程如下：

- 数据在串行时钟上升沿锁存
- 数据在串行时钟下降沿传输
- LSB 先传输
- SPICLK 空闲状态为高电平
- 每次发送/接收一个字节
- 从机选择信号为高电平触发

操作流程如下：

1) 将从机模式的相应配置值写入 SPI_SSR 寄存器

   设置从机选择有效电平位 SS_LVL (SPI_SSR[2] = 1) 和从机选择触发电平位 SS_LTRIG (SPI_SSR[4] = 1) 来选择高电平触发作为从机选择信号。

2) 通过设置 SPI_CNTRL 寄存器来控制 SPI 从机的行为。

   1. 通过设置 SLAVE 位 (SPI_CNTRL[18] = 1) 将 SPI 控制器设为从机设备
   2. 通过设置 CLKP 位 (SPI_CNTRL[11] = 1) 将串行时钟的空闲状态设为高电平
   3. 通过设置 TX_NEG 位 (SPI_CNTRL[2] = 1) 选择数据在串行时钟的下降沿传输
   4. 通过设置 RX_NEG 位 (SPI_CNTRL[1] = 0) 选择数据锁存在串行时钟的上升沿。
   5. 通过设置 TX_BIT_LEN 位域 (SPI_CNTRL[7:3] = 0x08) 设定一次字传输的长度为8-位。
   6. 通过设置 LSB 位 (SPI_CNTRL[10] = 1) 设定为 LSB 传输优先。

3) 如果 SPI 从机要发送一个字节的数据到外设主机，则将所要发送的数据写入到SPI_TX0 寄存器。

4) 如果 SPI 从机只是要从外设主机接收一字节数据，用户不必关心什么数据将被传输，不需要软件更新SPI_TX0 寄存器。

5) 使能 GO_BUSY 位 (SPI_CNTRL[0] = 1) 来等待片外主机设备的从机选择触发输入和串行时钟输入，以便开始在 SPI 接口的数据传输。 .

6) 等待 SPI 中断发生（如果中断使能位 IE 使能）或轮询检测 GO_BUSY 位直到被硬件自动清零。

7) 从寄存器 RX0 [7:0] (SPI_RX0[7:0]) 中读取接收到的一个字节数据。

8) 重复步骤 3) 继续其他数据传输或停止数据传输。

5.17.8 寄存器映射

R: 只读, **W**: 只写, **R/W**: 读/写

寄存器	偏移地址	R/W	描述	复位值
SPI 基地址: **SPI0_BA = 0x4003_0000** **SPI1_BA = 0x4003_4000** **SPI2_BA = 0x4013_0000** **SPI3_BA = 0x4013_4000**				
SPI_CNTRL **n=0,1,2,3**	SPIn_BA+0x00	R/W	控制状态寄存器	0x0500_3004
SPI_DIVIDER **n=0,1,2,3**	SPIn_BA+0x04	R/W	时钟分频寄存器	0x0000_0000
SPI_SSR **n=0,1,2,3**	SPIn_BA+0x08	R/W	从机选择寄存器	0x0000_0000
SPI_RX0 **n=0,1,2,3**	SPIn_BA+0x10	R	数据接收寄存器0	0x0000_0000
SPI_RX1 **n=0,1,2,3**	SPIn_BA+0x14	R	数据接收寄存器1	0x0000_0000
SPI_TX0 **n=0,1,2,3**	SPIn_BA+0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1 **n=0,1,2,3**	SPIn_BA+0x24	W	数据发送寄存器1	0x0000_0000
SPI_VARCLK **n=0,1,2,3**	SPIn_BA+0x34	R/W	可调时钟类型寄存器	0x007F_FF87
SPI_DMA **n=0,1,2,3**	SPIn_BA+0x38	R/W	SPI DMA 控制寄存器	0x0000_0000
SPI_CNTRL2 **n=0,1,2,3**	SPIn_BA+0x3C	R/W	控制与状态寄存器2	0x0000_1000
SPI_FIFO_CTL **n=0,1,2,3**	SPIn_BA+0x40	R/W	SPI FIFO 控制寄存器	0x4400_0000
SPI_STATUS **n=0,1,2,3**	SPIn_BA+0x44	R/W	SPI 状态寄存器	0x0500_0000

5.17.9 寄存器描述

SPI 控制与状态寄存器(SPI_CNTRL)

寄存器	偏移地址	R/W	描述	复位值
SPI_CNTRL	SPIn_BA+0x00	R/W	控制及状态寄存器	0x0500_3004

31	30	29	28	27	26	25	24
Reserved				TX_FULL	TX_EMPTY	RX_FULL	RX_EMPTY
23	22	21	20	19	18	17	16
VARCLK_EN	TWOB	FIFO	Reserved	REORDER	SLAVE	IE	IF
15	14	13	12	11	10	9	8
SP_CYCLE				CLKP	LSB	Reserved	
7	6	5	4	3	2	1	0
TX_BIT_LEN					TX_NEG	RX_NEG	GO_BUSY

位	描述	
[31:28]	**Reserved**	保留
[27]	**TX_FULL**	**发送 FIFO缓存满标志**（只读） 该位是SPI_STATUS[27]的相互镜像位. 1 = 表示 FIFO 上的发送数据缓存满了 0 = 表示发送数据缓存没满
[26]	**TX_EMPTY**	**发送 FIFO缓存空 标志**（只读） 该位是SPI_STATUS[26]的相互镜像位 1 = 表示发送数据缓存为空 0 = 表示发送数据缓存非空
[25]	**RX_FULL**	**接收 FIFO 缓存满标志**（只读） 该位是SPI_STATUS[25]的相互镜像位 1 = 表示 FIFO 上的接收数据缓存满了 0 = 表示接收数据缓存没满
[24]	**RX_EMPTY**	**接收 FIFO 缓存空标志**（只读） 该位是SPI_STATUS[24]的相互镜像位 1 = 表示接收数据缓存为空 0 = 表示接收数据缓存非空
[23]	**VARCLK_EN**	**可调时钟使能 (仅主模式)** 1 = 串行时钟输出频率可调。输出频率由 VARCLK, DIVIDER, 和 DIVIDER2 的值决定。 0 = 串行时钟输出频率固定，只由 DIVIDER 的值决定。 **注意**：当使能 VARCLK_EN 位，TX_BIT_LEN 必须设置为 0x10 (16-bit mode)
[22]	**TWOB**	**两位传输模式使能** 1 = 使能两位传输模式

		0 = 禁止两位传输模式. **注意：**当使能 TWOB，两位串行数据从 SPI_TX1/0 被发送，被接收到 SPI_RX1/0。
[21]	**FIFO**	**FIFO 模式使能位** 1 = 使能 FIFO 模式 0 = 禁用 FIFO 模式 **注意:** 1. 在使能 FIFO 模式前，其他相关的设定必须事先设定. 2. 在主机模式，如果 FIFO 模式被使能，在数据被写入 8 级深度发送 FIFO 中之后，GO_BUSY 将会自动被硬件设置为 1。当SPI控制器空闲，GO_BUSY 将会自动清0。当所有存储在发送FIFO缓存的数据全部发送出去 ，TX_EMPTY 会置1， GO_BUSY会清0。 3. 此位被清零后，如果用户需要再次置1，必须等待至少2个外设时钟周期。
[20]	**Reserved**	保留
[19]	**REORDER**	**字节重排序功能使能** 1 =使能字节重排序功能，在每两个字节之间插入一个字节休眠间隔。字节休眠间隔时间取决于SP_CYCLE 的设置。 0 =禁止字节重排序功能. **注意:** 1. 字节重排序仅在 TX_BIT_LEN 被定义为 16 位，24 位和 32 位时有效. 2. 在电平触发的从机模式下，如果字节休眠功能被使能，从机选择管脚必须在字节休眠间隔期间保持有效 。 3. 当可变时钟功能或者双 I/O 模式被使能时，不支持字节重排序功能。
[18]	**SLAVE**	**从机模式** 1 = SPI 控制器被设置为从机模式 0 = SPI控制器被设置为主机模式.
[17]	**IE**	**中断使能** 1 = 使能 SPI 中断 0 = 禁用 SPI 中断
[16]	**IF**	**中断标志位** 1 = 表示传输完成 0 =表示传输还没有完成 **注意：**该位写 1 清零。
[15:12]	**SP_CYCLE**	**休眠间隔(仅主模式)** 该四位提供在一次数据传输过程中连续两个发送/接收事务之间可配置的休眠间隔。如果CLKP =0，休眠间隔从当前事务的最后一个串行时钟的下降沿开始到接下来的事务的串行时钟的第一个上升沿结束；而如果 CLKP = 1，休眠间隔从当前事务的最后一个串行时钟的上升沿开始到接下来的事务的串行时钟的第一个下降沿结束。 缺省值是 0x3. 要求的休眠间隔根据如下公式获得： (SP_CYCLE[3:0] + 0.5) * SPICLK时钟周期 例: SP_CYCLE = 0x0 … 0.5 SPICLK 时钟周期 SP_CYCLE = 0x1 … 1.5 SPICLK 时钟周期 …… SP_CYCLE = 0xE … 14.5 SPICLK 时钟周期 SP_CYCLE = 0xF … 15.5 SPICLK 时钟周期

		如果可变时钟功能使能，在连续的两个事务之间最小休眠间隔时间 (在 FIFO 缓存中发送数据非空) 是 (6.5 + SP_CYCLE) * SPICLK clock cycle.
[11]	**CLKP**	**时钟极性** 1 = SPICLK 空闲状态的电平默认为高 0 = SPICLK 空闲状态的电平默认为低.
[10]	**LSB**	**LSB 优先** 1 = LSB, SPI_TX0/1 的位 0，首先被发送到 SPI 数据输出管脚；从 SPI 数据输入管脚上接收到的第一个数据位将会被放置到 SPI_RX 寄存器 (SPI_RX0/1) LSB 的位置. 0 = MSB, 具体是发送/接收寄存器的哪一位，首先被发送/接收，取决于 TX_BITLEN 的设定值。
[9:8]	**Reserved**	保留
[7:3]	**TX_BIT_LEN**	**传输位长** 该位域指定在一个发送/接收事务中，多少个数据位将会被传输。最小位长是 8，最多可以达到 32 位。 TX_BIT_LEN = 0x08 … 8 bits TX_BIT_LEN = 0x09 … 9 bits …… TX_BIT_LEN = 0x1F … 31 bits TX_BIT_LEN = 0x00 … 32 bits
[2]	**TX_NEG**	**在下降沿发送** 1 = 在 SPICLK 的下降沿发送数据信号改变 0 = 在 SPICLK 的上升沿发送数据信号改变
[1]	**RX_NEG**	**在下降沿接收** 1 = 在 SPICLK 的下降沿锁存数据输入信号. 0 = 在 SPICLK 的上升沿锁存数据输入信号.
[0]	**GO_BUSY**	**SPI 传输控制位和忙状态** 1 = 在主机模式下，写 1 到该位开始 SPI 数据传输。在从机模式下，写 1 到该位表示从机已经准备好与主机进行通信 0 =停止数据传输. 如果 FIFO 模式被禁止，在整个数据传输过程中，该位保持为 '1'。当传输结束，该位自动被清除。软件可以读取该位来检查 SPI 是否处于忙状态。 在 FIFO 模式下，该位由硬件控制。在软件不能修改该位的值。从机模式下，软件读该寄存器总是返回 1。在主机模式下，该位反映 SPI 是处于忙，还是空闲状态。 **注意:** 1. 当 FIFO 模式被禁止，在写 1 到 GO_BUSY 位之前，所有配置必须事先在 SPI_CTL 寄存器中被设定好。 1. 当 FIFO 被禁止，且软件使用 TX 或 RX PDMA 功能来传输数据，在 PDMA 控制器结束数据传输之后，该位将会被清除。

SPI 分频寄存器(SPI_DIVIDER)

寄存器	偏移地址	R/W	描述	复位值
SPI_DIVIDER	SPIn_BA+0x04	R/W	时钟分频寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
DIVIDER2[7:0]							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DIVIDER[7:0]							

位	描述	
[31:24]	**Reserved**	保留
[23:16]	**DIVIDER2**	**时钟分频 2 寄存器(Master Only)** 这些位是设置第二个频率分频器，用于产生可变时钟功能的第二个时钟。可根据下列公式获得所期望的频率： $f_{clock2} = \frac{f_{spi\_eclk}}{(DIVIDER2+1)*2}$ 当 VARCLK_EN 被清为零时，该设置无意义。
[15:8]	**Reserved**	保留
[7:0]	**DIVIDER**	**时钟除频器1寄存器** 该域的值是产生SPI主机的SPI引擎时钟、f_{spi_eclk},和SPI串行时钟的频率分频器，可根据下列公式获得所期望的频率： 如果BCn, SPI_CNTRL2[31], 为 0 $f_{spi\_eclk} = \frac{f_{system\_clock}}{(DIVIDER+1)*2}$ 如果BCn 为 1 $f_{spi\_eclk} = \frac{f_{spi\_clock\_src}}{(DIVIDER+1)}$ $f_{spi\_clock\_src}$ 是 SPI 引擎时钟源，在 CLK_SEL1 中定义

SPI 从机选择寄存器 (SPI_SSR)

寄存器	偏移地址	R/W	描述	复位值
SPI_SSR	SPIn_BA+0x08	R/W	从机选择寄存器	0x0000_0000

31	30	29	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		LTRIG_FLAG	SS_LTRIG	AUTOSS	SS_LVL	SSR	

位	描述	
[31:6]	**Reserved**	保留
[5]	**LTRIG_FLAG**	**电平触发完成标志** 当 SS_LTRIG 位在从机模式下被置位，读该位的值可用来表示一次传输完成后接收位的数量是否达到要求。 1 = 传输的位长度满足 TX_BIT_LEN定义的要求. 0 =一次传输的位长度不满足指定的要求 **注意:** 该位只读。当GO_BUSY 被软件置1， LTRIG_FLAG 会在4个SPI引擎时钟加一个系统时钟之后清0。在FIFO模式，该位没有意义。
[4]	**SS_LTRIG**	**从机选择电平触发 (Slave only)** 1 = 从机选择信号将是电平触发。根据 SS_LVL 来决定是高电平/低电平触发。 0 = 输入从机选择信号是边沿触发，该值为默认值。根据 SS_LVL 来决定是下降沿/上升沿触发。
[3]	**AUTOSS**	**自动从机选择 (Master only)** 1 = 该位置位，SPISSx0/1 信号自动产生。这表示当通过设置 GO_BUSY 位开始发送/接收时，SSR[1:0] 中设定的设备/从机选择信号将由 SPI 控制器设为有效状态，而当每次发送/接收结束时，设备/从机选择信号又会设为无效状态。 0 = 如果该位清零，从机选择信号将由 SSR[1:0] 相关位的设定值来决定激活或失效。
[2]	**SS_LVL**	**从机选择触发电平** 定义从机选择信号 (SPISSx0/1) 的有效状态. 1 = 从机选择信号 SPISSx0/1在高电平/上升沿有效 0 = 从机选择信号 SPISSx0/1在低电平/下降沿有效
[1:0]	**SSR**	**从机选择寄存器 (Master only)** 如果 AUTOSS 位被清零，写 1到 SSR[1:0] 任一位将会激活所对应的 SPISSx0/1 线，写 0 则相应的 SPISSx0/1 线为非激活状态。 如果 AUTOSS 位置 1，写 0 到 SSR[1:0] 任一位将会保持相应的 SPISSx0/1 线为非激活状态；写 1 到 SSR[1:0] 任一位将会选择相应的 SPISSx0/1 线在发送/接收的时间内被自动驱动到激活状态，而其他时间为非激活状态。SPISSx0/1 的激活状态类型由 SS_LVL 指

		定。 **注意：** SPISSx0 在从机模式下被定义为从机选择输入。

SPI 数据接收寄存器(SPI_RX)

寄存器	偏移地址	R/W	描述	复位值
SPI_RX0	SPIn_BA+0x10	R	数据接收寄存器 0	0x0000_0000
SPI_RX1	SPIn_BA+0x14	R	数据接收寄存器1	0x0000_0000

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

位	描述	
[31:0]	**RX**	**数据接收寄存器** 数据接收寄存器保存从SPI数据输入管脚接收到的数据。如果FIFO模式禁止，最后接收到的数据可以通过软件读该寄存器访问。如果FIFO位为1并且RX_EMPTY、SPI_CNTRL[24]或SPI_STATUS[24]没有被置1，接收FIFO缓存可以通过软件读该寄存器访问。该寄存器只读。

SPI 数据发送寄存器(SPI_TX)

寄存器	偏移地址	R/W	描述	复位值
SPI_TX0	SPIn_BA+0x20	W	数据发送寄存器 0	0x0000_0000
SPI_TX1	SPIn_BA+0x24	W	数据发送寄存器 1	0x0000_0000

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

位	描述	
[31:0]	**TX**	**数据发送寄存器** 数据发送寄存器内存储下一次被发送的数据。数据的有效长度依据 SPI_CNTRL 寄存器内定义的长度决定。 例如，如果 TX_BIT_LEN 为 0x08 ，则 TX0[7:0] 位将会被发送。如果 TX_BIT_LEN 为 0x00，则 SPI 控制器会传输32-位数据。 **注意1:**当SPI控制器配置位从机设备且FIFO模式禁止，如果SPI控制器想发送数据到主机，发送数据寄存器应该在设置GO_BUSY为1之前更新。 **注意2**:在主机模式，一旦写入数据到该寄存器后，SPI控制器在5个外设时钟周期后开始传输数据

SPI 可调时钟类型寄存器 (SPI_VARCLK)

寄存器	偏移地址	R/W	描述	复位值
SPI_VARCLK	SPIn_BA+0x34	R/W	可调时钟类型寄存器	0x007F_FF87

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

位	描述	
[31:0]	VARCLK	**可调时钟类型** 该寄存器的值表示 SPI 传输时钟的频率类型。如果可调时钟功能禁止，该设置没有意义。更多详细细节请参考可调串行时钟频率章节。

SPI DMA 控制寄存器(SPI_DMA)

寄存器	偏移地址	R/W	描述	复位值
SPI_DMA	SPIn_BA+0x38	R/W	SPI DMA 控制寄存器	0x0000_0000

31	30	29	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					PDMA _RST	RX_DMA_GO	TX_DMA_GO

位	描述	
[31:2]	Reserved	保留
[2]	PDMA_RST	**PDMA 复位** 1 = 复位SPI控制器的PDMA 控制逻辑。该位会自动清0。 0 = 无效.
[1]	RX_DMA_GO	**接收 DMA 开始** 设置该位为 1 将会开始接收 PDMA 处理过程。当SPI接收缓存不是空的，SPI 控制器将会自动向 PDMA 控制器触发请求。PWMA传输完成之后，该位会由硬件自动清0。 如果使用 RX_PDMA 去接收数据，但是没有使用PDMA发送功能，GO_BUSY 位必须由用户设定。 如果软件使用多个PDMA通道来传输数据，建议使能FIFO模式。 在从机模式，且 FIFO 位被禁止的情况下，如果软件仅使用一个PDMA 通道用来接收，其他的PDMA 通道没有使用，在边沿触发模式，在两个连续事务输入之间的最小休眠间隔需要大于 9 个 SPI 从机引擎时钟 + 4 APB 时钟；而在电平触发模式，该休眠间隔需要大于 9.5 个 SPI 从机引擎时钟 + 4 APB 时钟..
[0]	TX_DMA_GO	**发送 DMA 开始** 设置该位为 1 将会开始发送 PDMA 处理过程。SPI 控制器将会自动向 PDMA 控制器触发请求。在PDMA传输完成之后，硬件会自动清0该位。 如果使用 PDMA 模式来传输数据，不要去设定GO_BUSY 位。当需要的时候，SPI 控制器中的 DMA 控制器将会自动的设置 GO_BUSY 位。 在从机模式，且 FIFO 位被禁止的情况下，在边沿触发模式，在两个连续传输之间的最小休眠间隔需要大于 8 个 SPI 从机引擎时钟 +14 APB 时钟；而在电平触发模式，该休眠间隔需要大于 9.5 个 SPI 从机引擎时钟 + 14 APB 时钟。如果两位传输模式使能，在上述条件下，需要额外18个APB时钟周期。

SPI 控制和状态寄存器2 (SPI_CNTRL2)

寄存器	偏移地址	R/W	描述	复位值
SPI_CNTRL2	SPIn_BA+0x3C	R/W	控制和状态寄存器2	0x0000_1000

31	30	29	28	27	26	25	24
BCn	Reserved						
23	22	21	20	19	18	17	16
Reserved							SS_INT_OPT
15	14	13	12	11	10	9	8
Reserved		DUAL_ IO_EN	DUAL_ IO_DIR	SLV_START _INTSTS	SSTA_ INTEN	SLV_ABORT	NOSLVSEL
7	6	5	4	3	2	1	0
Reserved							

位	描述	
[31]	**BCn**	**SPI 引擎时钟向后兼容选项** 1 = 时钟配置不向后兼容. 0 =时钟配置向后兼容. 详情请参考SPI_DIVIDER 寄存器的描述
[30:17]	**Reserved**	保留
[16]	**SS_INT_OPT**	**从机选择无效中断选项** 该设置仅在SPI控制器配置为电平触发从机设备时有效。 1 = 当从机选择信号变为无效电平时， IF 位会被置 1。. 0 = 当从机选择信号变为无效电平时， IF 位不会被置 1。.
[15:14]	**Reserved**	保留
[13]	**DUAL_IO_EN**	**双 I/O 模式使能** 1 = 双 I/O 模式使能. 0 = 双 I/O 模式禁止.
[12]	**DUAL_IO_DIR**	**双 I/O 模式方向选择** 1 = 双输出模式 0 =双输入模式.
[11]	**SLV_START_INTSTS**	**从机 3-线模式开始中断状态** 该位用来表示在3线模式下，传输已经开始。它是SPI_STATUS[11]的相互镜像位。 1 = 在3线模式下，传输已经开始。该位在传输完成后自动清位或写 1 清位。 0 =自从SSTA_INTEN置1， 从机没有侦测到任何SPI 时钟。
[10]	**SSTA_INTEN**	**从机3-线模式开始中断使能** 当在没有从机选择的从机模式下传输开始时，该位用于使能中断。如果在传输开始后的（用户定义）时间后，没有传输完成的中断，则用户可以置位 SLV_ABORT 位强

		制完成传输。 1 =使能传输开始中断。该位在传输完成后清0或 SLV_START_INTSTS 位被清除后清0。 0 =禁用传输开始中断。
[9]	**SLV_ABORT**	**从机 3-线模式终止控制** 在正常操作中，当接收到数据符合 TX_BIT_LEN 的要求位的值时，会有中断事件。 如果接收到的位数少于要求的值而且在3线模式的从机模式下，在一次传输的时间后没有更多的串口时钟输入时，用户可以设定该位来强制完成当前的传输，然后用户就可以收到一个传输完成的中断事件。 **注意:** 软件设置该位为1后，该位会被硬件自动清0。
[8]	**NOSLVSEL**	**从机 3-线模式使能** 该位用于忽略从机模式下的从机选择信号。当 SPI 控制器被设置为从机设备时，它可以工作于3-线接口，包括 SPICLK, SPI_MISO 和 SPI_MOSI。. 1 = 3-线双向接口 0 = 4-线双向接口. **注意:** 在从机 3-线模式， SS_LTRIG, SPI_SSR[4] 会被自动置1。
[7:0]	**Reserved**	保留

SPI FIFO 控制寄存器(SPI_FIFO_CTL)

寄存器	偏移地址	R/W	描述	复位值
SPI_FIFO_CTL	SPIn_BA+0x40	R/W	SPI FIFO 控制寄存器	0x4400_0000

31	30	29	28	27	26	25	24
Reserved	TX_THRESHOLD			Reserved	RX_THRESHOLD		
23	22	21	20	19	18	17	16
Reserved		TIMEOUT_ INTEN	Reserved				
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	RXOV_ INTEN	Reserved		TX_INTEN	RX_INTEN	TX_CLR	RX_CLR

位	描述	
[31]	**Reserved**	保留
[30:28]	**TX_THRESHOLD**	**发送 FIFO 阈值** 如果发送FIFO缓存的有效数据数量小于或者等于TX_THRESHOLD， TX_INTSTS 将会被设置为 1， 否则TX_INTSTS 将会被设置为 0。
[27]	**Reserved**	保留.
[26:24]	**RX_THRESHOLD**	**接收 FIFO 阈值** 如果 接收FIFO缓存有效数据数量大于RX_THRESHOLD ，RX_INTSTS 将会被设置为 1， 否则RX_INTSTS 将会被设置为 0。
[23:22]	**Reserved**	保留
[21]	**TIMEOUT_INTEN**	**接收 FIFO 超时中断使能** 1 = 超时中断使能 0 = 超时中断禁止
[20:7]	**Reserved**	保留
[6]	**RXOV_INTEN**	**接收 FIFO Overrun中断使能** 1 = 接收 FIFO overrun中断使能 0 = 接收 FIFO overrun 中断禁止
[5:4]	**Reserved**	保留.
[3]	**TX_INTEN**	**发送阀值中断使能** 1 = TX阀值中断使能. 0 = TX阀值中断禁止.
[2]	**RX_INTEN**	**接收阀值中断使能** 1 = RX 阀值中断使能.

		0 = RX 阀值中断禁止
[1]	**TX_CLR**	**清发送FIFO 缓存** 1 = 清发送FIFO 缓存。TX_FULL标志会被清0并且TX_EMPTY 标志会被置1。通过软件写1到该位之后，硬件会自动将它清0。 0 = 无效.
[0]	**RX_CLR**	**清 FIFO 缓存** 1 = 清接收 FIFO 缓存。RX_FULL标志会被清0并且TX_EMPTY 标志会被置1。通过软件写1到该位之后，硬件会自动将它清0。. 0 = 无效

SPI 状态寄存器(SPI_STATUS)

寄存器	偏移地址	R/W	描述	复位值
SPI_STATUS	SPIn_BA+0x44	R/W	SPI 状态寄存器	0x0500_0000

31	30	29	28	27	26	25	24
TX_FIFO_COUNT				TX_FULL	TX_EMPTY	RX_FULL	RX_EMPTY
23	22	21	20	19	18	17	16
Reserved			TIMEOUT	Reserved			IF
15	14	13	12	11	10	9	8
RX_FIFO_COUNT				SLV_START_INTSTS	Reserved		
7	6	5	4	3	2	1	0
Reserved			TX_INTSTS	Reserved	RX_OVERRUN	Reserved	RX_INTSTS

位	描述	
[31:28]	**TX_FIFO_COUNT**	**发送 FIFO 数据计数 (只读)** 该位域表明发送FIFO缓存的有效数据计数。
[27]	**TX_FULL**	**发送 FIFO 缓存满标志 (只读)** SPI_CNTRL[27]的相互镜像位。. 1 = 发送 FIFO 缓存满. 0 = 发送 FIFO 缓存没满
[26]	**TX_EMPTY**	**发送 FIFO 缓存空标志 (只读)** SPI_CNTRL[26] 的相互镜像位。. 1 = 发送 FIFO 缓存空. 0 = 发送 FIFO 缓存非空.
[25]	**RX_FULL**	**接收 FIFO 缓存满标志(只读)** SPI_CNTRL[25] 的相互镜像位。. 1 = 接收 FIFO 缓存满 0 = 接收 FIFO 缓存不满.
[24]	**RX_EMPTY**	**接收 FIFO 缓存空标志 (只读)** SPI_CNTRL[24] 的相互镜像位。. 1 = 接收 FIFO 缓存为空 0 = 接收 FIFO 缓存非空
[23:21]	**Reserved**	保留
[20]	**TIMEOUT**	**超时中断标志** 1 = 接收 FIFO 缓存非空且主机模式下超过64个SPI时钟周期或从机模式下超过576个SPI引擎时钟周期没有读操作。当接收到的FIFO缓存被软件读取，超时状态会自动清0。 . 0 = 没有收到 FIFO 超时事件.

		注意: 向该位写1清0
[19:17]	**Reserved**	保留
[16]	**IF**	**SPI 单位传输中断标志** SPI_CNTRL[16]的相互镜像位. 1 = SPI 控制器已经完成一个单元传输. 0 = 一旦该位清0，没有传输完成 **注意:** 向该位写1清0.
[15:12]	**RX_FIFO_COUNT**	**接收 FIFO 数据计数 (只读)** 该域表明接收 FIFO 缓存有效数据计数..
[11]	**SLV_START_INTSTS**	**从机开始中断状态** 该位用来表明在3线模式下，是否已经开始了一次传输。是SPI_CNTRL2[11]的相互镜像位。 1 =在3线模式下，已经开始一次传输。当一次传输结束，或者写1到该位会清除该位。 0 = 自动SSTA_INTEN置1，从机没有侦测到任何SPI时钟传输。
[10:5]	**Reserved**	保留
[4]	**TX_INTSTS**	**发送 FIFO 阀值中断状态 (只读)** 1 =发送FIFO缓存的有效数据计数少于或等于TX_THRESHOLD的设定值。 0 =发送FIFO缓存的有效数据计数大于TX_THRESHOLD的设定值。 **注意:** 如果 TX_INTEN = 1 和 TX_INTSTS = 1，SPI 控制器会产生一个SPI中断请求。
[3]	**Reserved**	保留
[2]	**RX_OVERRUN**	**接收 FIFO Overrun 状态** 当接收 FIFO缓存满，上面的数据会丢掉，该位会置1。 **注意:** 向该位写1清0.
[1]	**Reserved**	保留
[0]	**RX_INTSTS**	**接收 FIFO 阀值中断状态(只读)** 1 =接收FIFO缓存的有效数据计数大于RX_THRESHOLD的设定值。 0 =接收FIFO缓存的有效数据计数小于或等于RX_THRESHOLD的设定值。 **注意:** 如果 RX_INTEN = 1 和 RX_INTSTS = 1，SPI 控制器会产生一个SPI中断请求

5.18 I²S 控制器 (I²S)

5.18.1 概述

I²S 控制器由I²S 协议与外部音频 CODEC 接口组成。两个8字节的 FIFO 分别用于读和写通道，可以处理8-, 16-, 24- 和 32- 位字节大小。PDMA 控制器处理 FIFO 和 内存之间的数据移动。

5.18.2 特征

- 支持主机和从机模式
- 能处理8-, 16-, 24- 和 32-位字节大小
- 支持单声道和立体声的音频数据
- 支持I²S 和 MSB 校正数据格式
- 提供两个8字节的 FIFO 数据缓存，一个用于发送，一个用于接收
- 支持PDMA传输

5.18.3 框图

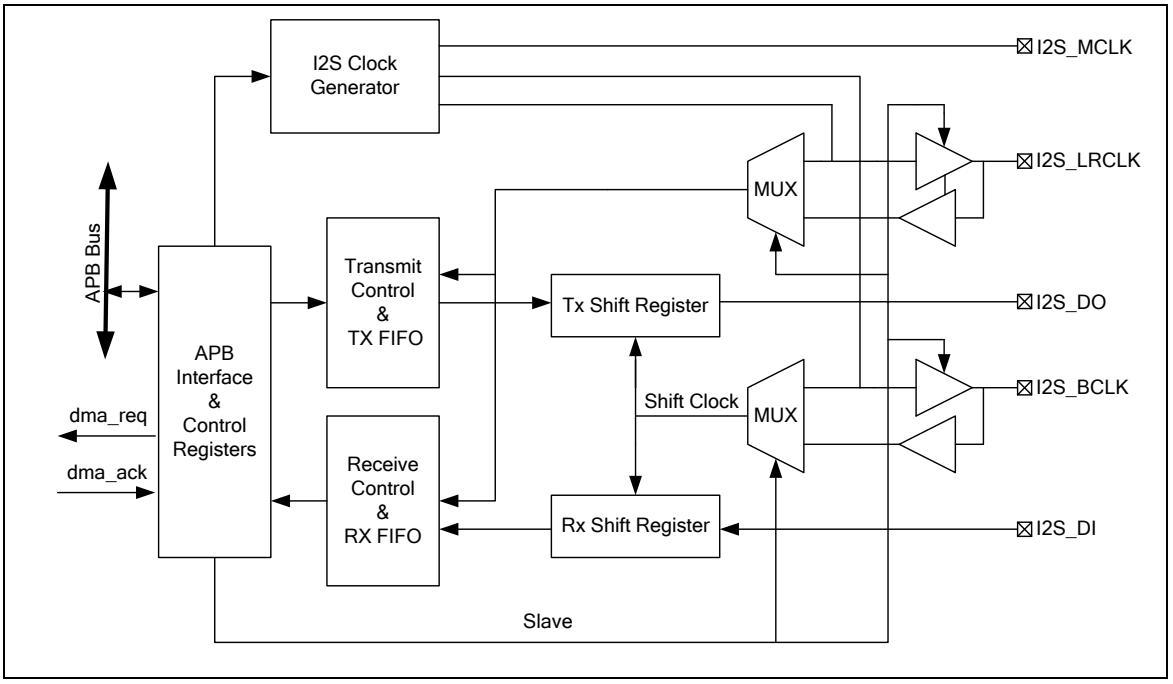


图 5-128 I²S 控制器框图

5.18.4 基本配置

I²S管脚功能由寄存器GPA_MFP, GPC_MFP, ALT_MFP 和ALT_MFP1配置。I²S外围时钟可以由I2S_EN (APBCLK[29])使能。时钟源由I2S_S (CLKSEL2[1:0])设定。

5.18.5 功能描述

5.18.5.1 *I²S 时钟*

I²S 控制器有四个时钟源，通过 I2S_S(CLKSEL2[1:0])选择。I²S 时钟频率必须低于或等于系统时钟。

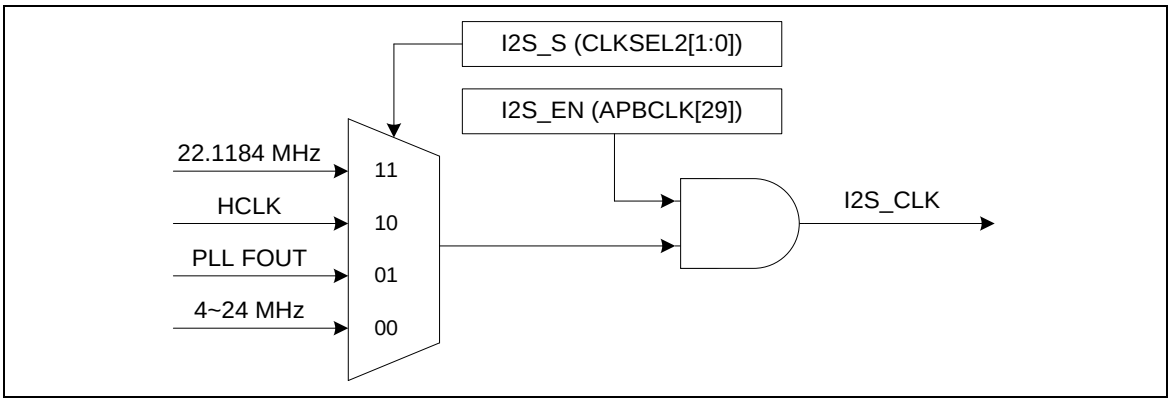


图 5-129 I²S 时钟控制框图

5.18.5.2 *I²S 操作*

I²S控制器支持MSB调整和I²S数据格式。I2SLRCLK信号表明哪一个声音通道正在传输。一个声音通道的位数计数由WORDWIDTH的设定决定。传输顺序总是从最重要的位MSB开始。读数据在上升时钟沿，数据在下降时钟沿驱动。

在 I²S 数据格式中，MSB在声音通道的第二个时钟被发送和锁存。

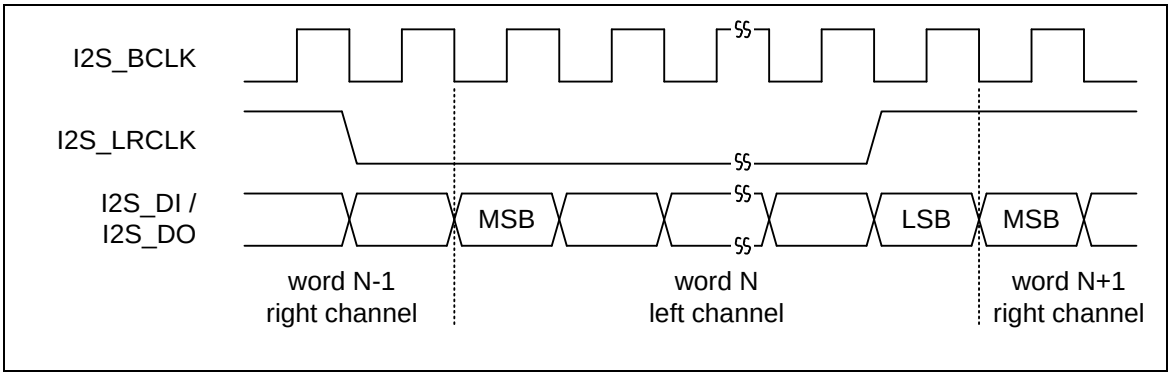


图 5-130 I²S 数据格式时序图

在 MSB调整的数据格式中，MSB在声音通道的第一个时钟被发送和锁存。

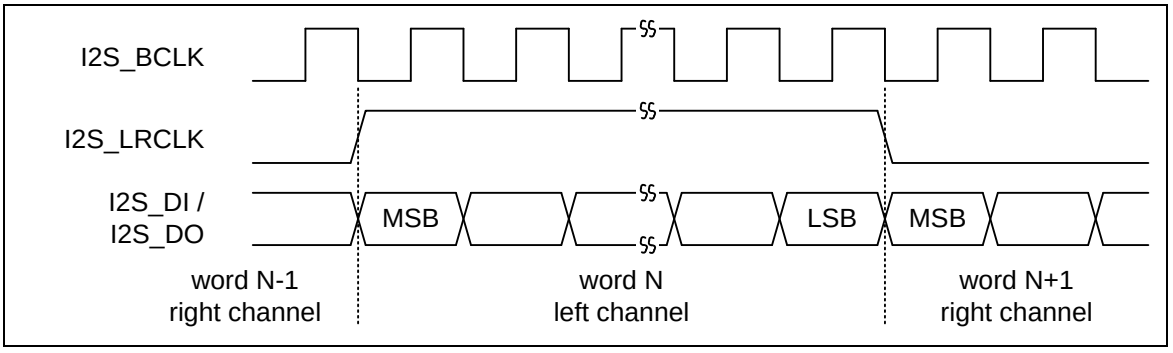


图 5-131 MSB 调整的数据格式时序图

5.18.5.3 *I²S 中断源*

在发送操作中，I²S控制器支持左通道过零中断，右通道过零中断，发送FIFO阀值中断，发送FIFO溢出中断和发送FIFO下溢中断。在接收操作中，支持接收FIFO阀值中断，接收FIFO溢出中断和接收FIFO下溢中断。当I²S中断发生，用户可以检查I2STXINT(I2SSTATUS[2]) 和 I2SRXINT (I2SSTATUS[1])标志来识别中断源。

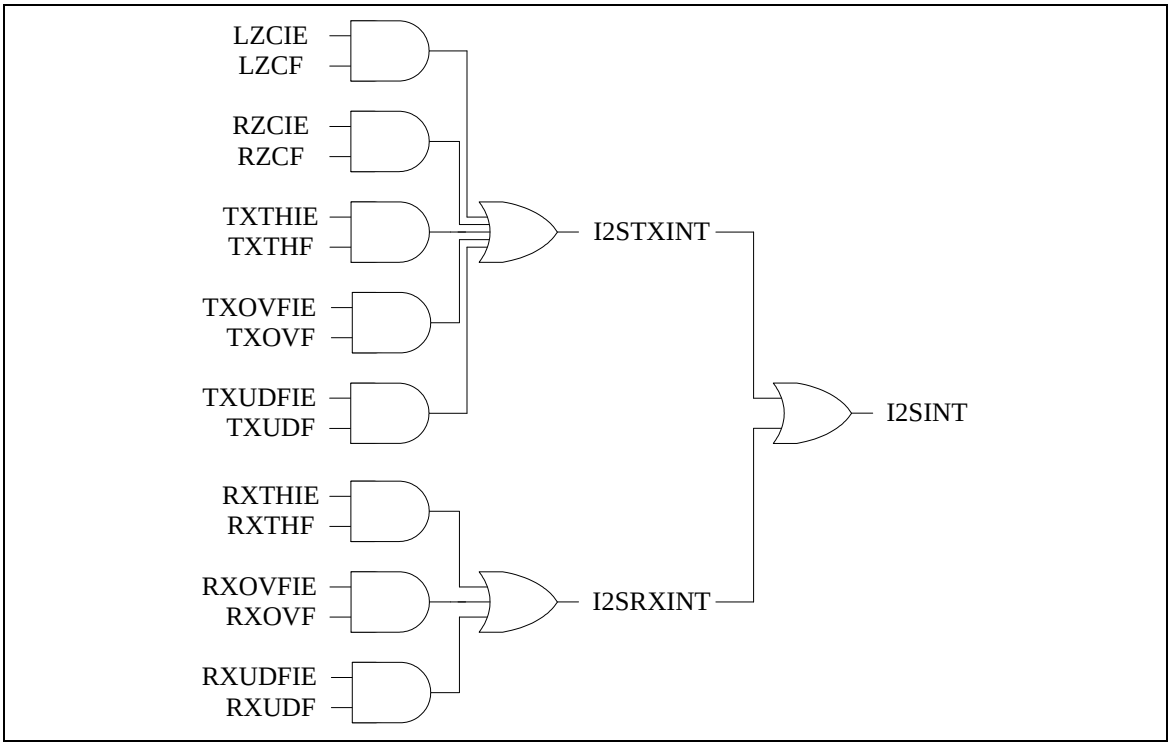


图 5-132 I²S 中断

5.18.5.4 *FIFO 操作*

一个声音通道的字长可以是 8, 16, 24 或 32 位。各种设置的存储器排列如下图所示。

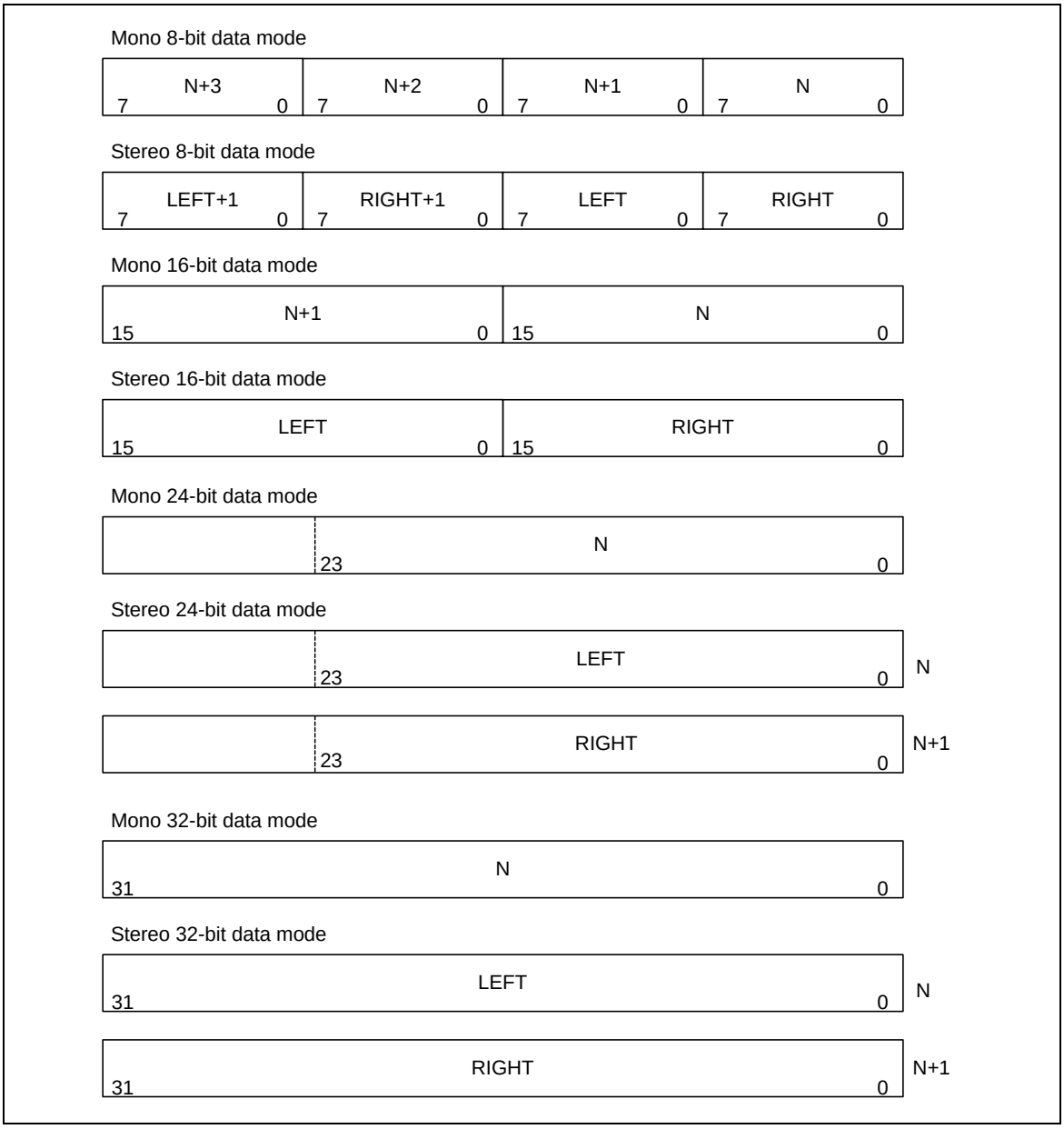


图 5-133 各种 I²S 模式的 FIFO 内容

5.18.5.5 *过零检测*

当通过I²S功能播放音频，所播放的数据通过PDMA或CPU从内存取得。但是用户在改变播放增益电平时会导致有pop声。因为输出数据不是零，输出数据经过增益调整输出后，将会产生刺耳的POP噪音。因此，过零标志将减少这种状况。如果使能过零检测功能，硬件将检测下一个传输数据是零还是正负符号改变。如果是零或者信号改变，过零检测标志将置1，并且自动静音，直到标志被软件清除。

5.18.5.6 *PDMA 模式*

I²S功能可以用PDMA功能访问数据。在传输模式中 使能PDMA功能，如果TX FIFO没满，I²S将产生请求信号和通过PDMA从存储器中自动取数据，直到RX FIFO 满。用户可以使能PDMA功能来降低CPU的负担。而在接收模式下，当PDMA功能被使能后，如果RX FIFO不是空的，I2S将会产生请求信号并通过PDMA自动搬移接收到的数据到memory，直到RX FIFO为空为止。

5.18.5.7 *主机/从机接口*

如果I²S控制器配置为主机模式，它通过I2S_MCLK, I2S_BCLK 和 I2S_LRCLK信号线驱动从设备，例如音频CODEC。

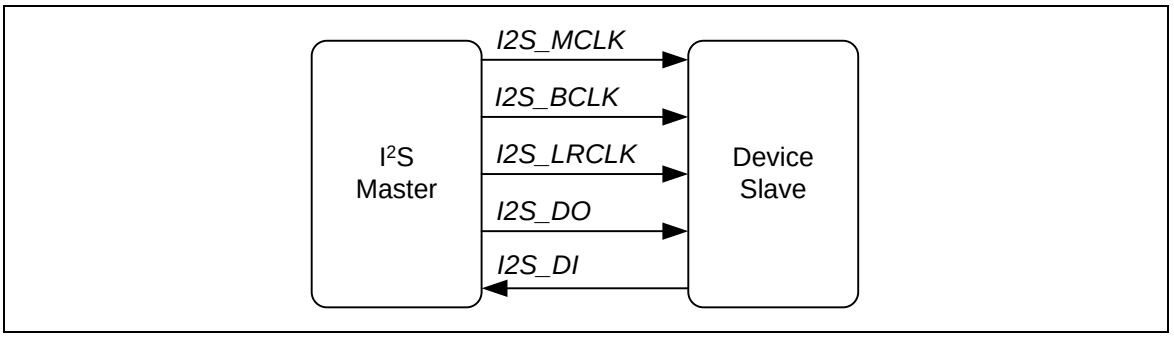


图 5-134 主模式传输

如果I²S控制器配置为从机模式， I2S_MCLK, I2S_BCLK 和 I2S_LRCLK被主机设备驱动，如同一个音频CODEC。

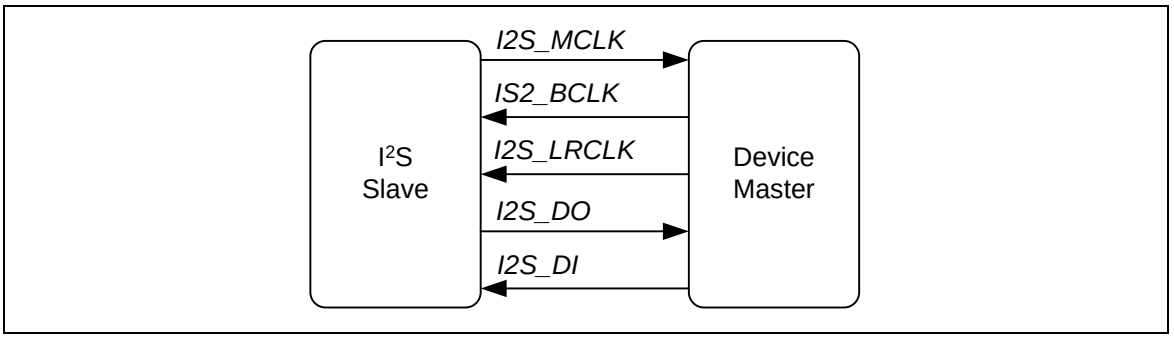


图 5-135 从机模式输入

5.18.6 寄存器映射

R: 只读, **W**:只写, **R/W**:读/写

寄存器	偏移地址	R/W	描述	复位值
I²S 基地址: **I2S_BA = 0x401A_0000**				
I2SCON	I2S_BA+0x00	R/W	I²S控制寄存器	0x0000_0000
I2SCLKDIV	I2S_BA+0x04	R/W	I²S时钟分频寄存器	0x0000_0000
I2SIE	I2S_BA+0x08	R/W	I²S中断使能寄存器	0x0000_0000
I2SSTATUS	I2S_BA+0x0C	R/W	I²S状态寄存器	0x0014_1000
I2STXFIFO	I2S_BA+0x10	W	I²S传送 FIFO 寄存器	0x0000_0000
I2SRXFIFO	I2S_BA+0x14	R	I²S接收 FIFO 寄存器r	0x0000_0000

5.18.7 寄存器描述

I2S控制寄存器(I2SCON)

寄存器	偏移地址	R/W	描述	复位值
I2SCON	I2S_BA+0x00	R/W	I^2S控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
RXLCH	Reserved	RXDMA	TXDMA	CLR_RXFIFO	CLR_TXFIFO	LCHZCEN	RCHZCEN
15	14	13	12	11	10	9	8
MCLKEN	RXTH			TXTH			SLAVE
7	6	5	4	3	2	1	0
FORMAT	MONO	WORDWIDTH		MUTE	RXEN	TXEN	I2SEN

位	描述	
[31:22]	**Reserved**	保留.
[23]	**RXLCH**	**接收左通道使能** 当选择单声道格式时(MONO = 1)，如果RXLCH置0，I^2S 控制器会接收右通道数据，如果RXLCH置1，I^2S 控制器会接收左通道数据。 1 = 单声道模式时，接收左声道数据。 0 = 单声道模式时，接收右声道数据。.
[22]	**Reserved**	保留.
[21]	**RXDMA**	**使能接收 DMA** 当 RX DMA 使能时，如果 FIFO 非空，I^2S 请求 DMA 从接收 FIFO 向 SRAM 传输数据。 1 = 使能 RX DMA 0 = 禁用 RX DMA
[20]	**TXDMA**	**使能传送 DMA** 当 TX DMA 使能时，如果 FIFO 没满，I^2S 请求 DMA 从 SRAM 向 发送 FIFO 传输数据。 1 = 使能 TX DMA 0 = 禁用 TX DMA
[19]	**CLR_RXFIFO**	**清除接收 FIFO** 写1 清接收 FIFO，内部指针复位指向 FIFO 的起始位置， RXFIFO_LEVEL[3:0] 返回 0，接收 FIFO 变为空。 该位由硬件自动清零，读时返回0。

[18]	**CLR_TXFIFO**	**清除发送 FIFO** 写 1 清发送 FIFO，内部指针复位指向 FIFO 的起始位置， TXFIFO_LEVEL[3:0] 返回 0，发送 FIFO 变为空，但在发送 FIFO 中的数据不变。 该位由硬件自动清零，读时返回0。
[17]	**LCHZCEN**	**使能左声道过零检测** 如果该位置 1，当左声道数据符号位改变或下一个移位数据位都为0，则寄存器 I2S_STATUS 的 LZCF 标志被置为 1。该功能仅在传输操作时有效。 1 = 使能左声道过零检测 0 = 禁用左声道过零检测
[16]	**RCHZCEN**	**使能右声道过零检测** 如果该位置 1，当右声道数据符号位改变或下一个移位数据位都为0，则寄存器 I2S_STATUS 的 RZCF 标志被置为 1。该功能仅在传输操作时有效。 1 = 使能右声道过零检测 0 = 禁用右声道过零检测
[15]	**MCLKEN**	**主时钟使能位** 如果 MCLKEN 置1，I^2S 控制器会在I2SMCLK管脚上产生主时钟用于外部音频设备。 1 = 主时钟使能 0 = 主时钟禁止
[14:12]	**RXTH**	**接收 FIFO 阀值水平** 当缓存中的接收字等于或大于阈值水平时，RXTHF (I2SSTATUS[10]) 标志置位。 000 = 接收 FIFO 中有 1 字数据 001 = 接收 FIFO 中有 2 字数据 010 = 接收 FIFO 中有 3 字数据 011 = 接收 FIFO 中有 4 字数据 100 = 接收 FIFO 中有 5 字数据 101 = 接收 FIFO 中有 6 字数据 110 = 接收 FIFO 中有 7 字数据 111 = 接收 FIFO 中有 8 字数据
[11:9]	**TXTH**	**发送 FIFO 阈值水平** 如果发送 FIFO 中剩下的数据字 (32 位) 等于或小于阈值水平，则 TXTHF (I2SSTATUS[18]) 标志置位。 000 = 发送 FIFO 中有 0 个字数据 001 = 发送 FIFO 中有 1 个字数据 010 = 发送 FIFO 中有 2 个字数据 011 = 发送 FIFO 中有 3 个字数据 100 = 发送 FIFO 中有 4 个字数据 101 = 发送 FIFO 中有 5 个字数据 110 = 发送 FIFO 中有 6 个字数据 111 = 发送 FIFO 中有 7 个字数据

[8]	**SLAVE**	**从机模式** I^2S 可工作于主机模式或从机模式。主机模式下，I2S_BCLK 和 I2S_LRCLK 管脚都为输出模式并作为 NuMicro™ NUC200 系列发送位时钟到音频 CODEC 芯片。从机模式下，I2S_BCLK 和 I2S_LRCLK 管脚都为输入模式，接收来自外部音频 CODEC 芯片的 I2S_BCLK 和 I2S_LRCLK 信号。 1 = 从机模式 0 = 主机模式
[7]	**FORMAT**	**数据格式** 1 = MSB 校正数据格式 0 = I^2S 数据格式
[6]	**MONO**	**单声道数据** 1 = 数据为单声道格式 0 = 数据为立体声格式
[5:4]	**WORDWIDTH**	**字宽度** 00 = 数据为 8-位 01 = 数据为 16-位 10 = 数据为 24-位 11 = 数据为 32-位
[3]	**MUTE**	**发送静音使能位** 1= 发送通道数据为 0 0 = 发送数据由缓存移动
[2]	**RXEN**	**接收使能** 1 = 使能数据接收 0 = 禁用数据接收
[1]	**TXEN**	**发送使能** 1 = 使能数据发送 0 = 禁用数据发送
[0]	**I2SEN**	**使能 I^2S 控制器** 1 = 使能 0 = 禁用

I²S时钟分频寄存器(I2SCLKDIV)

寄存器	偏移地址	R/W	描述	复位值
I2SCLKDIV	I2S_BA+0x04	R/W	I²S 时钟分频控制寄存器	0x0000_0000

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

位	描述	
[31:16]	**Reserved**	保留.
[15:8]	**BCLK_DIV**	**位时钟分频** 在主机模式下，I²S控制器会产生位时钟 。位时钟率 F_BCLK由下面表达式决定。 F_BCLK = F_I2SCLK /(2x(BCLK_DIV + 1)) ， F_I2SCLK 是 I²S 时钟的频率。.
[7:3]	**Reserved**	保留.
[2:0]	**MCLK_DIV**	**主时钟分频** 如果 MCLKEN置1， I²S控制器会位外部音频设备产生主时钟。主时钟率F_MCLK由下面表达式决定。 如果 MCLK_DIV >= 1, F_MCLK = F_I2SCLK/(2x(MCLK_DIV)) 如果 MCLK_DIV = 0, F_MCLK = F_I2SCLK F_I2SCLK 是 I²S 时钟的频率。. 通常，主时钟频率是采样时钟频率的256倍。

I^2S中断使能寄存器 (I2SIE)

寄存器	偏移地址	R/W	描述	复位值
I2SIE	I2S_BA+0x08	R/W	I^2S 中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**LZCIE**	**RZCIE**	**TXTHIE**	**TXOVFIE**	**TXUDFIE**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**RXTHIE**	**RXOVFIE**	**RXUDFIE**

位	描述	
[31:13]	**Reserved**	保留
[12]	**LZCIE**	**左声道过零检测中断使能位** 如果该位设为1 且左声道有过零信号，则中断产生。 1 = 使能中断 0 = 禁用中断
[11]	**RZCIE**	**右声道过零检测中断使能位** 如果该位设为 1 且 右声道有过零信号，则中断产生。 1 = 使能中断 0 = 禁用中断
[10]	**TXTHIE**	**发送 FIFO 阈值水平中断使能位** 如果该位为1，而且发送 FIFO 中的数据字小于TXTH （I2SCON[11:9]），则中断发生。 1 = 使能中断 0 = 禁用中断
[9]	**TXOVFIE**	**接收 FIFO 溢出中断使能位** 如果该位为1，而且发送 FIFO 的溢出标志设为 1，则中断发生。 1 = 使能中断 0 = 禁用中断
[8]	**TXUDFIE**	**发送 FIFO 下溢中断使能** 如果该位为1，而且发送 FIFO 的下溢标志设为 1，则中断发生。 1 = 使能中断 0 = 禁用中断
[7:3]	**Reserved**	保留.

[2]	**RXTHIE**	**接收 FIFO 阈值水平中断使能位** 当接收 FIFO 中的数据字等于或大于 RXTH(I2SCON[14:12])，且RXTHIE被置1，则中断发生。 1 = 使能中断 0 = 禁用中断
[1]	**RXOVFIE**	**接收 FIFO 溢出中断使能位** 1 = 使能中断 0 = 禁用中断
[0]	**RXUDFIE**	**接收 FIFO 下溢中断使能** 1 = 使能中断 0 = 禁用中断

I²S 状态寄存器 (I2SSTATUS)

寄存器	偏移地址	R/W	描述	复位值
I2SSTATUS	I2S_BA+0x0C	R/W	I²S 状态寄存器	0x0014_1000

31	30	29	28	27	26	25	24
TX_LEVEL				RX_LEVEL			
23	22	21	20	19	18	17	16
LZCF	RZCF	TXBUSY	TXEMPTY	TXFULL	TXTHF	TXOVF	TXUDF
15	14	13	12	11	10	9	8
Reserved			RXEMPTY	RXFULL	RXTHF	RXOVF	RXUDF
7	6	5	4	3	2	1	0
Reserved				RIGHT	I2STXINT	I2SRXINT	I2SINT

位	描述	
[31:28]	**TX_LEVEL**	**发送 FIFO 水平** 这些位表示发送 FIFO 中数据字的数目。 0000 = 无数据 0001 = 发送 FIFO 有 1 个字 …. 1000 = 发送 FIFO 有 8 个字
[27:24]	**RX_LEVEL**	**接收 FIFO 水平** 这些位表示接收 FIFO 中数据字的数目。 0000 = 无数据 0001 = 接收 FIFO 有 1 个字 …. 1000 = 接收 FIFO 有 8 个字
[23]	**LZCF**	**左声道过零标志** 该位表示左声道下一个采样数据符号位已经改变或所有数据位为 0。 1 = 检测到左声道过零 0 = 没有过零 **注意**：写 1 清除该位 为 0。
[22]	**RZCF**	**右声道过零标志** 该位表示左声道下一个采样数据符号位已经改变或所有数据位为 0。 1 = 检测到右声道过零 0 = 没有过零 **注意**：写 1 清除该位 为 0。

[21]	**TXBUSY**	**发送忙** 当发送 FIFO 中的所有数据和移位缓存都被移出时，该位清零。当第一个数据加载到移位缓存时，该位置 1。 1 = 发送移位缓存忙 0 = 发送移位缓存为空 **注意**：该位只读。
[20]	**TXEMPTY**	**发送 FIFO 为空** 该位反映发送 FIFO 中的数据字号为 0. 1 = 空 0 = 非空 **注意**：该位只读。
[19]	**TXFULL**	**发送 FIFO 满** 该位反映发送 FIFO 中的数据字号为8。 1 = 满 0 = 没满. **注意**：该位只读。
[18]	**TXTHF**	**发送 FIFO 阈值标志** 当发送 FIFO 中的数据字等于或低于 TXTH(I2SCON[11:9]) 中设定的阈值，则 TXTHF 位变为 1。该位将保持 1 直到TX_LEVEL (I2SSTATUS[31:28]) 高于 TXTH[1:0]。 1 = FIFO 中的数据字等于或低于阈值水平 0 = FIFO 中的数据字高于阈值水平 **注意**：该位只读。
[17]	**TXOVF**	**发送 FIFO 溢出标志** 当发送 FIFO 已满时写数据到该 FIFO，该位被置为 1。 1 = 溢出 0 = 没有溢出 **注意**：写 1 清除该位为 0。
[16]	**TXUDF**	**发送 FIFO 下溢标志** 当发送 FIFO 为空且移位逻辑硬件从数据 FIFO 读数据,会导致该位被置为1。 1 = 下溢 0 = 没有下溢 **注意**：写 1 清除该位为 0。
[15:13]	**Reserved**	保留.
[12]	**RXEMPTY**	**接收 FIFO 为空** 该位表示接收 FIFO 中的数据字号为0 1 = 空 0 = 非空 **注意**：该位只读。

[11]	**RXFULL**	**接收 FIFO 满** 该位表示接收 FIFO 中的数据字号为8 1 = 满 0 = 未满 **注意**：该位只读。
[10]	**RXTHF**	**接收 FIFO 阈值标志** 当接收 FIFO 中数据字等于或高于RXTH (I2SCON[14:12])中所设定的阈值时，RXTHF 位变为 1。该位将保持 1直到RX_LEVEL (I2SSTATUS[27:24]) 低于 RXTH[1:0]。 1 = FIFO 中数据字等于或高于阈值水平 0 = FIFO 中数据字低于阈值水平 **注意**：该位只读。
[9]	**RXOVF**	**接收 FIFO 溢出标志** 当接收 FIFO 满了时，接收硬件还尝试写数据到接收 FIFO，则该位被置为1，第一个缓存中数据被覆盖。 1 = 发生溢出 0 = 没发生溢出 **注意**：写 1 清除该位为 0。
[8]	**RXUDF**	**接收 FIFO 下溢标志** 当接收 FIFO 为空时，读接收 FIFO，则该位置 1表示发生下溢。 1 = 发生下溢 0 = 没有发生下溢 **注意**：写 1 清除该位为 0。
[7:4]	**Reserved**	保留.
[3]	**RIGHT**	**右声道** 该位表示当前的发送数据属于右声道。 1 = 右声道 0 = 左声道 **注意**：该位只读。
[2]	**I2STXINT**	**I^2S 发送中断** 1 = 发送中断 0 = 无发送中断 **注意**：该位只读。
[1]	**I2SRXINT**	**I^2S 接收中断** 1 = 接收中断 0 = 无接收中断 **注意**：该位只读。
[0]	**I2SINT**	**I^2S 中断标志** 该位为 I2STXINT 和 I2SRXINT 位的“或”。 1 = I^2S 中断 0 = 无 I^2S 中断 **注意**：该位只读。

I²S发送FIFO寄存器 (I2STXFIFO)

寄存器	偏移地址	R/W	描述	复位值
I2STXFIFO	I2S_BA+0x10	W	I²S发送FIFO寄存器	0x0000_0000

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

位	描述	
[31:0]	**TXFIFO**	**发送 FIFO 寄存器** I²S 包含 8个字 (8x32 bit) 的数据缓存用于数据传送。写数据到该寄存器准备用于传送的数据。剩余的字数目由TX_LEVEL (I2SSTATUS[31:28]) 指示。

I²S接收FIFO寄存器 (I2SRXFIFO)

寄存器	偏移地址	R/W	描述	复位值
I2SRXFIFO	I2S_BA+0x14	R	I²S 接收FIFO寄存器	0x0000_0000

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

位	描述	
[31:0]	**RXFIFO**	**接收 FIFO 寄存器** I²S 包含 8 个字 (8x32 bit) 的数据缓存用于数据接收。读该寄存器获取 FIFO 中的数据。剩下的数据字数目由 I2S_STATUS 寄存器的RX_LEVEL (I2SSTATUS[27:24])指示。

5.19 USB 器件控制器(USBD)

5.19.1 预览

本器件带一套USB 2.0全速设备控制器发送器。其与USB 2.0全速设备规范兼容，并支持控制/批量/中断/同步四种传输类型。

在此设备控制器中，有两个主要接口，APB总线和USB总线。USB总线来自于USB 硬件收发器。APB总线，CPU可以通过该总线来控制相应控制寄存器。控制器中还有个512字节的内部SRAM作为数据缓冲区。CPU通过APB或SIE对SRAM读写进行数据传输。使用过程中，用户需要先通过寄存器(USB_BUFSEGx)对每一个端点在SRAM中设置相应的有效起始地址。

该控制器共有8个端点。每个端点可独立配置成输入或输出模式。所有传输模式包括控制/批量/中断/同步四种模式都通过这一模块传输。端点控制模块也用于管理数据流同步，端点状态，端点起始地址，处理状态和每个端点数据缓冲状态。

控制器中有四个不同的中断事件，它们是唤醒事件，器件插拔事件，USB事件和BUS事件。以上任何事件都会导致一个中断产生，用户只需要在中断事件状态寄存器(USB_INTSTS)查找相关事件标志就可以知道发生了哪种中断事件，然后查找相关的USB端点状态寄存器(USB_EPSTS)就可以知道在这个端点中发生了何种中断事件。

在这个USB控制器中也支持软件断开连接功能。这个功能用于仿真从设备从主设备断开连接的过程。如果DRVSE0位 (USB_DRVSE0[0])被置位，USB控制器将强迫把USB_D+ 和 USB_D-拉倒低电平。DRVSE0位被清零后，主设备将再次枚举所插入的USB设备。

详细内容请参考*Universal Serial Bus Specification Revision 1.1*

5.19.2 特性

- 兼容USB 2.0全速规范
- 提供一个包括4种不同中断事件（包括唤醒、插拔、USB、总线）在内的中断向量
- 支持控制、批量、中断、同步四种传输类型
- 支持当总线闲置3ms以上切换到总线挂起功能
- 提供可配置为控制/批量/中断/同步四种传输模式的8个通讯端点，以及一个最大512字节的数据缓冲区
- 提供远程唤醒功能

5.19.3 框图

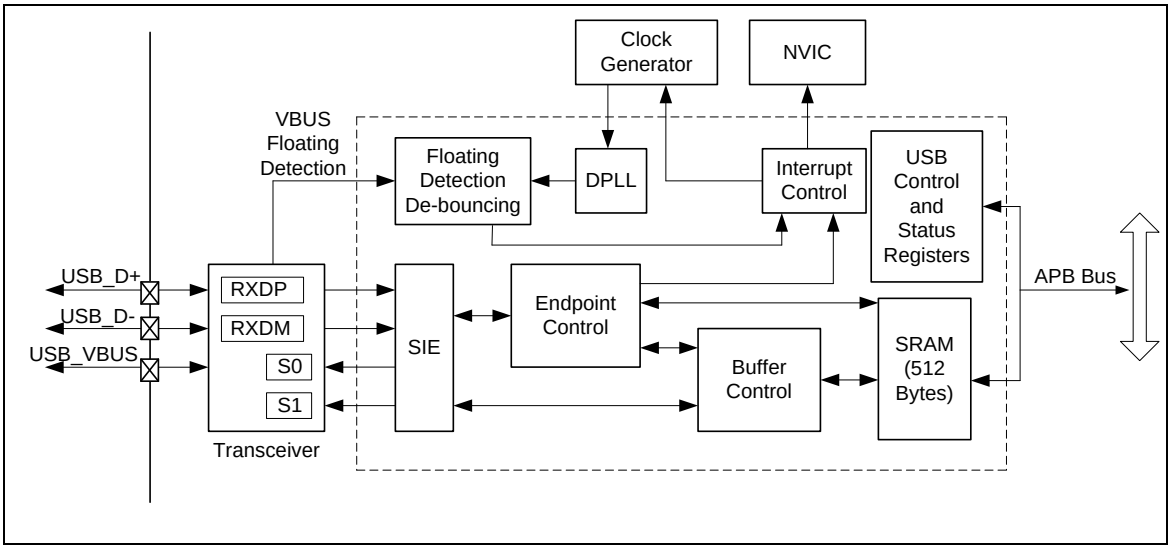


图 5-136USB 设备框图

5.19.4 基本配置

USB设备的时钟源由PLL时钟分频所得。用户必须在USB设备控制器使能前，先对PLL时钟进行相关配置。设置USBD_EN 位(APBCLK[27])可使能USB时钟，设置USB_N (CLKDIV[7:4])对USB时钟源预分频，产生合适的时钟频率。

5.19.5 功能描述

5.19.5.1 *SIE (串行接口引擎)*

SIE是设备控制器的前端，用于处理多数的USB包协议。SIE主要的主要工作是把数据传输到传出层。它所处理的工作包括：

- 包识别，处理传输序列
- SOF,EOP,复位和恢复 信号的检测与产生
- 时钟/数据分离
- NRZI数据编解码和位填充
- CRC校验的生成和检测（仅对Token和data）
- 包ID (PID)产生和检测/解码
- 串-并/并-串转换

5.19.5.2 *端点控制*

控制器中总共有8个端点。所有的端点都可以配置成控制，批量，中断或同步传输模式。这四种模式所对应的传输过程都是通过这个模块来完成。该控制器也用于管理数据流，同步，端点状态控制，当前端点起始地址，当前传输状态，以及每个端点的数据缓冲区状态。

5.19.5.3 *数字锁相环(DPLL)*

USB数据的传输速率是12MHz，DPLL使用来自时钟控制器的48MHz时钟源来锁定RXDP和RXDM上的输入数据。12M时钟源也同样来自DPLL。

5.19.5.4 *插拔去抖检测*

USB设备有可能经常被插拔。当USB设备被拔下后，为了监测到它的状态，设备控制器提供了一个USB检测中断硬件去抖，来避免USB插拔时的抖动问题。USB设备插10ms后会产生一个去抖检测中断。用户可以通过读USB_FLDET寄存器的内容知道USB设备的插拔状态。FLDET标志表示USB BUS 没有de-bouncing 情况下的当前状态。如果FLDET为1，表示USB线被插入。如果用户轮询该标志来检测USB的状态，需要通过软件来做de-bouncing.

5.19.5.5 *中断*

USB控制器带有一个USB中断向量，该中断包含了四个中断事件（唤醒事件，插拔事件，USB事件，BUS事件）。其中唤醒中断事件当系统进入到Power-down低功耗模式后用于唤醒系统时钟，（低功耗模式在Power-down控制寄存器PWRCON做了定义）。插拔中断事件用于USB设备插拔检测。USB中断事件用于告知MCU产生了一些USB请求，如IN ACK, OUT ACK等。BUS中断事件用来告知MCU产生了UBS事件，如挂起，恢复等。当需要用到这些中断时，必须在USB设备控制器的中断使能控制寄存器(USB_INTEN)中打开相应位。

唤醒中断是系统在Power- down模式下，把MCU唤醒的中断。系统进入Power-down后，如果唤醒使能，USB_VBUS,USB_D+ 和 USB_D-上的任何电平变化可以唤醒MCU。如果这个变化不是有意而为的，那么只有唤醒中断会发生。若 USB 唤醒超过 20 ms 后，没有其他 USB 中断事件发生，唤醒中断将发生。下图为唤醒中断的控制流程。

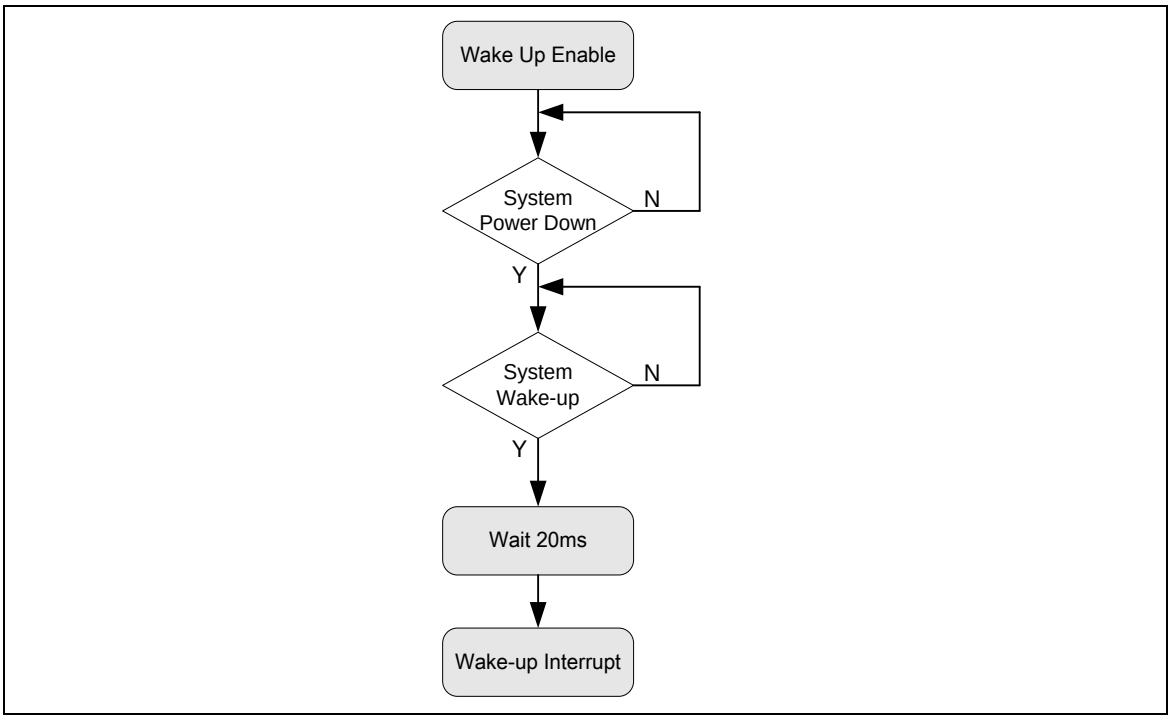


图 5-137 唤醒中断操作流程

USB 中断事件用于告知 MCU USB 总线上所发生的事件， MCU 可以读取 EPSTS 位(USB_EPSTS[31:8])和EPEVT7~0 位(USB_INTSTS[23:16])的内容，可以知道USB当前的状态以便进行下一步操作。

和USB中断事件一样，BUS中断事件是用于告知MCU一些BUS事件。比如USB复位，挂起，超时溢出以及总线恢复等。可以读USB_ATTR寄存器内容从而知道USB的总线的状态。

5.19.5.6 *省电*

当MCU进入Power-down 模式，USB模块会自动关闭PHY收发器以节省功耗。在挂起等特殊情况下，也可以写0到USB_ATTR[4]位来手动禁止PHY来省电。

5.19.5.7 *缓冲控制*

USB控制器中总共有512字节的SRAM和共享这些RAM缓冲区的8个端点。在USB模块功能使能前，应先在缓冲区内给每个端点分配一个有效起始地址。”缓冲区控制”模块就是用于控制每个端点的有效起始地址和它所分配的SRAM缓冲区间大小，缓冲区的大小在各个端点相应的USB_MXPLDx寄存器

中设置。.

錯誤! 找不到參照來源。描述了，根据USB_BUFSEGx 和 USB_MXPLDx寄存器中所定义的，各个端点缓冲区起始地址和大小。比如USB_BUFSEG0被设置位0x08h, USB_MXPLD0被设为0x40h,那么端点0所分配的缓冲区的大小就是从USBD_BA+0x108h开始，到USBD_BA+0x148h结束。(注意：USB 的SRAM起始地址是从USBD_BA+0x100h开始).

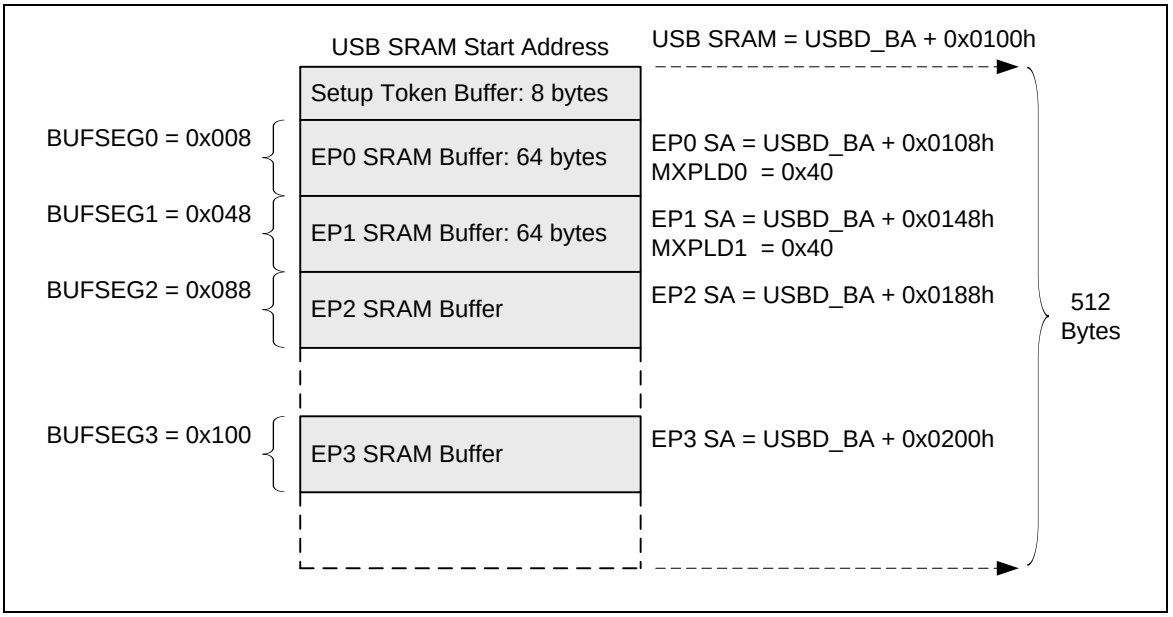


图 5-138 端点 SRAM 结构图

5.19.5.8 *使用USB设备外设处理收发*

用户可以使用中断或轮询USB_INTSTS寄存器来监视USB的数据收发。当传输开始后，USB_INTSTS寄存器被硬件置位，并产生一个中断请求给CPU（相关中断必须打开），或者也可以不使用中断方式，用轮询USB_INTSTS寄存器的相应位的方法来获取事件信息。以下是使用中断方式的控制流程。

当USB主对从设备发出一个读数据请求后，从设备需要把相关数据放到指定的端点缓冲区。填充完数据后，从设备需要写实际数据长度到USB_MXPLDx这个寄存器当中。一旦数据被写入，立即会产生一个“In_Rdy”信号，当接收到主设备发来的IN token信号后，缓冲区数据会被立即传送出去。需要注意的是指定数据被发送完成后，“In_Rdy”信号会自动取消。

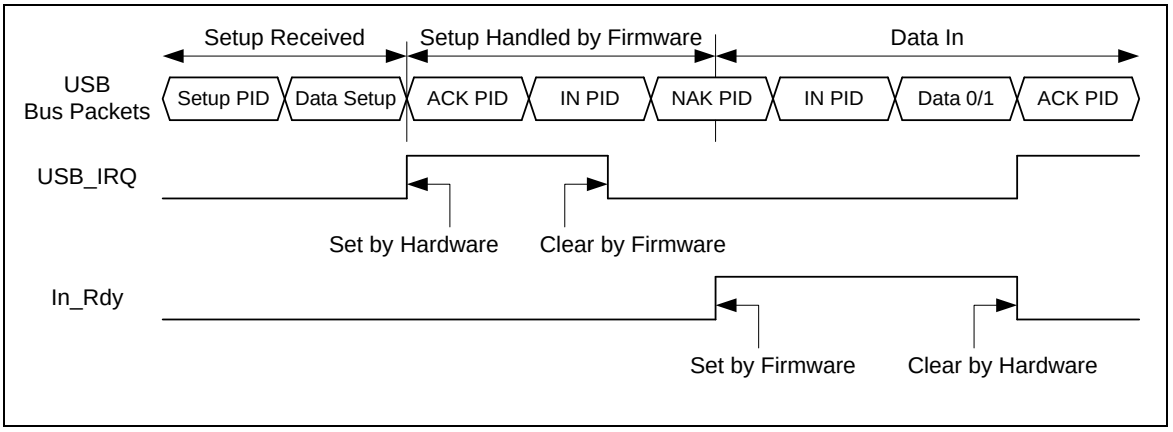


图 5-139 主机读从机数据发送流程图

相应的，当USB主机需要传送数据到从设备控制器的输出端点，硬件会把数据填充到指定的端点缓冲区。填充完成后，硬件会在端点对应的USB_MXPLDx寄存器中自动记录数据长度，并取消“Out_Rdy”信号。这种做法可以避免当前端点的数据被搬移完以前，硬件又接收到新数据。一旦数据被搬移完毕，用户需要手动对USB_MXPLDx寄存器软件清零，以便再次产生“Out_Rdy”信号来接收下一个数据。

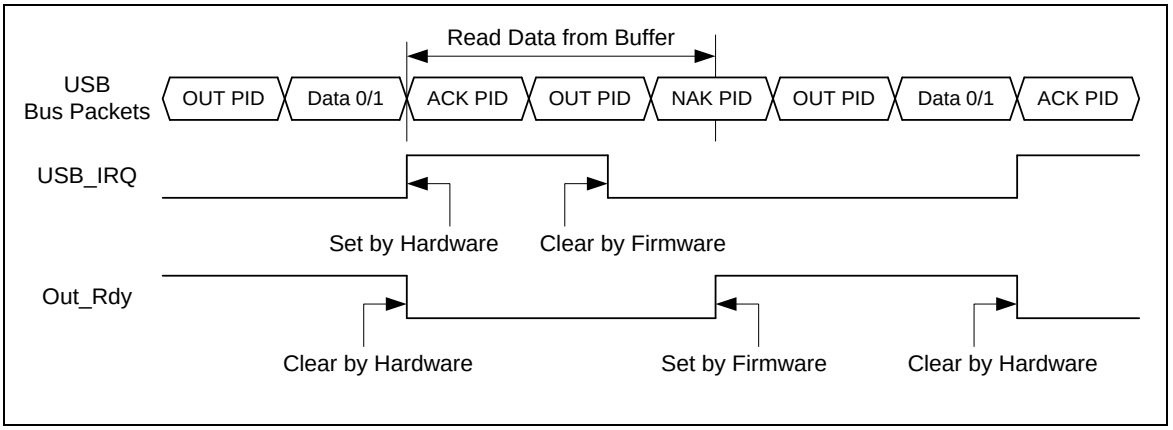


图 5-140 主机数据输出发送

5.19.6 寄存器映射

R: 只读, **W**: 只写, **R/W**: 可读写

寄存器	偏移地址	R/W	寄存器描述	复位值
USB 基地址 USBD_BA = 0x4006_0000				
USB_INTEN	USBD_BA+0x000	R/W	USB 中断使能寄存器	0x0000_0000
USB_INTSTS	USBD_BA+0x004	R/W	USB 中断事件状态寄存器	0x0000_0000
USB_FADDR	USBD_BA+0x008	R/W	USB 从设备功能地址寄存器	0x0000_0000
USB_EPSTS	USBD_BA+0x00C	R	USB 端点状态寄存器	0x0000_0000
USB_ATTR	USBD_BA+0x010	R/W	USB总线状态和属性寄存器	0x0000_0040
USB_FLDET	USBD_BA+0x014	R	USB 插拔监测寄存器	0x0000_0000
USB_STBUFSEG	USBD_BA+0x018	R/W	Setup Token 包缓冲区设置寄存器	0x0000_0000
USB_DRVSE0	USBD_BA+0x090	R/W	USB Drive SE0 控制寄存器	0x0000_0001
USB_BUFSEG0	USBD_BA+0x500	R/W	端点0缓冲区地址设置寄存器	0x0000_0000
USB_MXPLD0	USBD_BA+0x504	R/W	端点0 最大载荷寄存器	0x0000_0000
USB_CFG0	USBD_BA+0x508	R/W	端点0 配置寄存器	0x0000_0000
USB_CFGP0	USBD_BA+0x50C	R/W	端点0设置Stall和清除In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG1	USBD_BA+0x510	R/W	端点1缓冲区地址设置寄存器	0x0000_0000
USB_MXPLD1	USBD_BA+0x514	R/W	端点1 最大载荷寄存器	0x0000_0000
USB_CFG1	USBD_BA+0x518	R/W	端点1 配置寄存器	0x0000_0000
USB_CFGP1	USBD_BA+0x51C	R/W	端点1设置Stall和清除In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG2	USBD_BA+0x520	R/W	端点2缓冲区地址设置寄存器	0x0000_0000
USB_MXPLD2	USBD_BA+0x524	R/W	端点2 最大载荷寄存器	0x0000_0000
USB_CFG2	USBD_BA+0x528	R/W	端点2 配置寄存器	0x0000_0000
USB_CFGP2	USBD_BA+0x52C	R/W	端点2设置Stall和清除In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG3	USBD_BA+0x530	R/W	端点3缓冲区地址设置寄存器	0x0000_0000
USB_MXPLD3	USBD_BA+0x534	R/W	端点3 最大载荷寄存器	0x0000_0000
USB_CFG3	USBD_BA+0x538	R/W	端点3 配置寄存器	0x0000_0000
USB_CFGP3	USBD_BA+0x53C	R/W	端点3设置Stall和清除In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG4	USBD_BA+0x540	R/W	端点4缓冲区地址设置寄存器	0x0000_0000
USB_MXPLD4	USBD_BA+0x544	R/W	端点4 最大载荷寄存器	0x0000_0000
USB_CFG4	USBD_BA+0x548	R/W	端点4 配置寄存器	0x0000_0000
USB_CFGP4	USBD_BA+0x54C	R/W	端点4设置Stall和清除In/Out 准备控制寄存器	0x0000_0000

USB_BUFSEG5	USBD_BA+0x550	R/W	端点5缓冲区地址设置寄存器	0x0000_0000
USB_MXPLD5	USBD_BA+0x554	R/W	端点5 最大载荷寄存器	0x0000_0000
USB_CFG5	USBD_BA+0x558	R/W	端点5 配置寄存器	0x0000_0000
USB_CFGP5	USBD_BA+0x55C	R/W	端点5设置Stall和清除In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG6	USBD_BA+0x560	R/W	端点6缓冲区地址设置寄存器	0x0000_0000
USB_MXPLD6	USBD_BA+0x564	R/W	端点6 最大载荷寄存器	0x0000_0000
USB_CFG6	USBD_BA+0x568	R/W	端点6 配置寄存器	0x0000_0000
USB_CFGP6	USBD_BA+0x56C	R/W	端点6设置Stall和清除In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG7	USBD_BA+0x570	R/W	端点7缓冲区地址设置寄存器	0x0000_0000
USB_MXPLD7	USBD_BA+0x574	R/W	端点7 最大载荷寄存器	0x0000_0000
USB_CFG7	USBD_BA+0x578	R/W	端点7 配置寄存器	0x0000_0000
USB_CFGP7	USBD_BA+0x57C	R/W	端点7设置Stall和清除In/Out 准备控制寄存器	0x0000_0000

存储器类型	**地址**	**容量**	**描述**
USBD_BA = 0x4006_0000			
SRAM	USBD_BA+0x100 ~ USBD_BA+0x2FF	512 字节	SRAM用于整个端点缓冲区。详细内容请参考段5.4.4.7，端点SRAM结构和描述

5.19.7 寄存器描述

USB 中断使能寄存器(USB_INTEN)

寄存器	偏移地址	读/写	描述	复位值
USB_INTEN	USBD_BA+0x000	R/W	USB中断使能寄存器	0x0000_0000

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

位	描述	
[31:16]	保留	保留
[15]	**INNAK_EN**	收到IN Token 时，激活的 NAK功能及状态 0 =当从设备收到 IN token 应答 NAK 时，NAK 状态不会被更新到端点状态寄存器 USB_EPSTS，所以不会产生 INNAK 中断事件。 1 = 当从设备收到 IN token 时应答 NAK，NAK 的状态被更新到端点状态寄存器 USB_EPSTS，并发生 INNAK 中断事件。
[14:9]	保留	保留
[8]	**WAKEUP_EN**	唤醒功能使能位 0 = USB.唤醒功能禁止 1 = USB唤醒功能使能
[7:4]	保留	保留
[3]	**WAKEUP_IE**	USB唤醒中断使能位 0 =唤醒中断禁止 1 =唤醒中断使能
[2]	**FLDET_IE**	插拔检测中断使能位 0 =插拔检测中断禁止 1 =插拔检测中断使能
[1]	**USB_IE**	USB事件中断使能位 0 = USB事件中断禁止 1 = USB事件中断使能
[0]	**BUS_IE**	Bus事件中断使能位 0 = BUS事件中断禁止

		1 = BUS事件中断使能

USB中断事件状态寄存器(USB_INTSTS)

寄存器	偏移地址	R/W	描述	复位值
USB_INTSTS	USBD_BA+0x004	R/W	USB中断事件状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
SETUP	保留						
23	22	21	20	19	18	17	16
EPEVT7	EPEVT6	EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留				WAKEUP_STS	FLDET_STS	USB_STS	BUS_STS

位	描述	
[31]	SETUP	**Setup 事件状态** 0 = 没有SETUP事件 1 = 发生SETUP事件，写1 到 USB_INTSTS[31]位清零.
[30:24]	保留	保留
[23]	EPEVT7	**端点 7的USB 事件状态** 0 = 端点7没有事件发生 1 = 端点7有USB 事件发生，检查 USB_EPSTS[31:29] 位可以知道发生了哪种USB事件，可以对USB_INTSTS[23] 位或USB_INTSTS[1]位写1来清零.
[22]	EPEVT6	**端点 6的USB 事件状态** 0 =端点6没有事件发生. 1 =端点6有USB 事件发生，检查USB_EPSTS[28:26] t位可以知道发生了哪种USB事件，可以对 USB_INTSTS[22] 位或 USB_INTSTS[1]位写1来清零
[21]	EPEVT5	**端点 5的USB 事件状态** 0 =端点5没有事件发生 1 =端点5有USB 事件发生，检查USB_EPSTS[25:23] 位可以知道发生了哪种USB事件，可以对 USB_INTSTS[21]位或 USB_INTSTS[1]位写1来清零
[20]	EPEVT4	**端点4的USB 事件状态** 0 =端点4没有事件发生 1 =端点4有USB 事件发生，检查 USB_EPSTS[22:20] 位可以知道发生了哪种USB事件，可以对USB_INTSTS[20] 位或 USB_INTSTS[1]位写1来清零
[19]	EPEVT3	**端点 3的USB 事件状态** 0 =端点3没有事件发生 1 =端点3有USB 事件发生，检查 USB_EPSTS[19:17] 位可以知道发生了哪种USB事件,，可以对USB_INTSTS[19] 位或USB_INTSTS[1]位写1来清零

[18]	**EPEVT2**	**端点2的USB 事件状态** 0 =端点2没有事件发生 1 =端点2有USB 事件发生，检查 USB_EPSTS[16:14] 位可以知道发生了哪种USB事件，可以对USB_INTSTS[18] 位或USB_INTSTS[1]位写1来清零
[17]	**EPEVT1**	**端点 1的USB 事件状态** 0 =端点1没有事件发生 1 =端点1有USB 事件发生，检查 USB_EPSTS[13:11] 位可以知道发生了哪种USB事件，可以对USB_INTSTS[17]位或USB_INTSTS[1]位写1来清零
[16]	**EPEVT0**	**端点 0的USB 事件状态** 0 =端点0没有事件发生. 1 =端点0有USB 事件发生，检查 USB_EPSTS[10:8] 位可以知道发生了哪种USB事件,可以对USB_INTSTS[16] 位或USB_INTSTS[1]位写1来清零
[15:4]	保留	保留
[3]	**WAKEUP_STS**	**唤醒中断状态** 0 = 没有唤醒事件发生 1 = 有唤醒事件发生，对USB_INTSTS[3]位写1来清零
[2]	**FLDET_STS**	**插拔检测中断状态** 0 = 没有USB设备插拔事件发生 1 = USB总线上有插拔事件发生，对USB_INTSTS[2]位写1清零
[1]	**USB_STS**	**USB 中断事件状态** USB 事件包括在总线上的SETUP Token, IN Token, OUT ACK, ISO IN, 或 ISO OUT 事件 0 = 没有USB事件发生 1 = 有USB事件发生，检查 EPSTS0~7 位可以知道发生了哪种USB事件。对USB_INTSTS[1] 位或 对EPEVT0~7 和SETUP (USB_INTSTS[31])位写1清零
[0]	**BUS_STS**	**BUS中断事件状态** BUS中断事件说明总线上发生了挂起或恢复功能. 0 = 没有BUS中断事件发生 1 = 发生了Bus中断事件; 检查 USB_ATTR[3:0] 位可以知道发生了哪种BUS中断事件，对USB_INTSTS[0]位写1清零

USB 设备功能地址寄存器 (USB_FADDR)

7位有效数据用于设置USB总线上从设备地址.

寄存器	偏移地址	R/W	描述	复位值
USB_FADDR	USBD_BA+0x008	R/W	USB 从设备功能地址寄存器	0x0000_0000

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

位	描述	
[31:7]	保留	保留
[6:0]	**FADDR**	USB从设备功能地址

USB 端点状态寄存器(USB_EPSTS)

寄存器	偏移地址	R/W	描述	复位值
USB_EPSTS	USBD_BA+0x00C	R	USB 端点状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
EPSTS7			EPSTS6			EPSTS5[2:1]	
23	22	21	20	19	18	17	16
EPSTS5[0]	EPSTS4			EPSTS3			EPSTS2[2]
15	14	13	12	11	10	9	8
EPSTS2[1:0]		EPSTS1			EPSTS0		
7	6	5	4	3	2	1	0
OVERRUN	保留						

位	描述	
[31:29]	**EPSTS7**	**端点7总线状态** 这些位用于指示该端点的当前状态 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 110 = Out Packet Data1 ACK. 011 = Setup ACK. 111 = 同步传输结束.
[28:26]	**EPSTS6**	**端点6总线状态** 这些位用于指示该端点的当前状态 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 110 = Out Packet Data1 ACK. 011 = Setup ACK. 111 =同步传输结束.
[25:23]	**EPSTS5**	**端点5总线状态** 这些位用于指示该端点的当前状态 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 110 = Out Packet Data1 ACK. 011 = Setup ACK. 111 =同步传输结束.
[22:20]	**EPSTS4**	**端点4总线状态** 这些位用于指示该端点的当前状态

		000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 110 = Out Packet Data1 ACK. 011 = Setup ACK. 111 =同步传输结束.
[19:17]	**EPSTS3**	**端点3总线状态** 这些位用于指示该端点的当前状态 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 110 = Out Packet Data1 ACK. 011 = Setup ACK. 111 =同步传输结束..
[16:14]	**EPSTS2**	**端点2总线状态** 这些位用于指示该端点的当前状态 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 110 = Out Packet Data1 ACK. 011 = Setup ACK. 111 =同步传输结束..
[13:11]	**EPSTS1**	**端点1总线状态** 这些位用于指示该端点的当前状态 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 110 = Out Packet Data1 ACK. 011 = Setup ACK. 111 =同步传输结束.
[10:8]	**EPSTS0**	**端点0总线状态** 这些位用于指示该端点的当前状态 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 110 = Out Packet Data1 ACK. 011 = Setup ACK. 111 =同步传输结束..
[7]	**OVERRUN**	**Overrun** 用于指示所接收数据长度是否超过最大负荷范围 0 = 未超出. 1 =主机发送数据超出MXPLD 寄存器设置范围，或Setup 数据超过8 个字节
[6:0]	保留	保留

USB 总线状态和属性寄存器 (USB_ATTR)

寄存器	偏移地址	R/W	描述	复位值
USB_ATTR	USBD_BA+0x010	R/W	USB 总线状态和属性寄存器	0x0000_0040

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留					BYTEM	PWRDN	DPPU_EN
7	6	5	4	3	2	1	0
USB_EN	保留	RWAKEUP	PHY_EN	TIMEOUT	RESUME	SUSPEND	USBRST

位	描述	
[31:11]	保留	保留
[10]	BYTEM	CPU 存取 USB SRAM 大小模式选择 0 = 字模式: CPU到 USB SRAM 的数据传输必须以字为单位 1 =字节模式: CPU到 USB SRAM 的数据传输必须以字节为单位
[9]	PWRDN	PHY 收发器开关 0 = 关闭PHY收发器相关电路. 1 = 打开 PHY 收发器相关电路.
[8]	DPPU_EN	USB_D+ 上拉电阻使能位 0 = USB_D+ 脚上的上拉电阻禁止. 1 = USB_D+脚上的上拉电阻打开.
[7]	USB_EN	USB 控制器使能位 0 = USB 控制器禁止. 1 = USB 控制器使能
[6]	保留	保留
[5]	RWAKEUP	远程唤醒 0 = 把USB总线从 K 状态释放 1 = 强迫USB 总线到 K (USB_D+ low, USB_D- high) 状态, 用于远程唤醒
[4]	PHY_EN	PHY 收发器使能位 0 = PHY 收发器功能禁止. 1 = PHY收发器功能使能.
[3]	TIMEOUT	时间溢出状态 0 =没有时间溢出情况发生 1 =超过18个位长度计数器时间，总线上仍没有数据回复.

		注意:该位只读
[2]	**RESUME**	**总线恢复状态** 0 =没有总线恢复. 1 = 总线从挂起状态恢复. **注意:** 该位只读.
[1]	**SUSPEND**	**挂起状态** 0 =总线没有挂起事件发生. 1 = 总线闲置状态超过3ms, 有可能是从设备被拔下或主机进入了睡眠模式 **注意:** 该位只读.
[0]	**USBRST**	**USB 总线复位状态** 0 =没有总线复位 1 = 有总线复位，此时SE0 (single-ended 0)已超过 2.5us. **注意:** 该位只读.

插拔检测寄存器(USB_FLDET)

寄存器	偏移地址	R/W	描述	复位值
USB_FLDET	USBD_BA+0x014	R	USB 插拔检测寄存器	0x0000_0000

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

位	描述	
[31:1]	保留	保留
[0]	**FLDET**	**设备插拔检测状态** 0 = 设备没有插入到USB主机 1 =设备被插入到USB主机

缓冲区设置寄存器(USB_STBUFSEG)

仅对Setup token包

寄存器	偏移地址	R/W	描述	复位值
USB_STBUFSEG	USBD_BA+0x018	R/W	Setup Token 缓冲区设置寄存器	0x0000_0000

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

位	描述	
[31:9]	保留	保留
[8:3]	**STBUFSEG**	**Setup Token包缓冲区设置寄存器** 用于指示 SETUP token 包在USB设备SRAM中的起始地址，它的有效地址是USB_SRAM address + {STBUFSEG[8:3], 3'b000} 此处USB_SRAM address = USBD_BA+0x100h. **注意:**该内容只适用于SETUP toke.
[2:0]	保留	保留

Buffer Segmentation 寄存器 (USB_BUFSEGx)

寄存器	偏移地址	R/W	描述	复位值
USB_BUFSEG0	USBD_BA+0x500	R/W	端点0缓冲区设置寄存器	0x0000_0000
USB_BUFSEG1	USBD_BA+0x510	R/W	端点1缓冲区设置寄存器	0x0000_0000
USB_BUFSEG2	USBD_BA+0x520	R/W	端点2缓冲区设置寄存器	0x0000_0000
USB_BUFSEG3	USBD_BA+0x530	R/W	端点3缓冲区设置寄存器	0x0000_0000
USB_BUFSEG4	USBD_BA+0x540	R/W	端点4缓冲区设置寄存器	0x0000_0000
USB_BUFSEG5	USBD_BA+0x550	R/W	端点5缓冲区设置寄存器	0x0000_0000
USB_BUFSEG6	USBD_BA+0x560	R/W	端点6缓冲区设置寄存器	0x0000_0000
USB_BUFSEG7	USBD_BA+0x570	R/W	端点7缓冲区设置寄存器	0x0000_0000

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

位	描述	
[31:9]	保留	保留
[8:3]	**BUFSEG**	**端点缓冲区设置** 用于指示每个端点在USB SRAM 的偏移地址，端点的有效起始地址是 USB_SRAM 地址 + { BUFSEG[8:3], 3'b000} 此处USB_SRAM 地址 = USBD_BA+0x100h. 请参考段5.4.4.7 查阅端点SRAM的结构和相关描述
[2:0]	**保留**	保留

最大负荷设置寄存器 (USB_MXPLDx)

寄存器	偏移地址	R/W	描述	复位值
USB_MXPLD0	USBD_BA+0x504	R/W	端点0最大负荷设置寄存器	0x0000_0000
USB_MXPLD1	USBD_BA+0x514	R/W	端点1最大负荷设置寄存器	0x0000_0000
USB_MXPLD2	USBD_BA+0x524	R/W	端点2最大负荷设置寄存器	0x0000_0000
USB_MXPLD3	USBD_BA+0x534	R/W	端点3最大负荷设置寄存器	0x0000_0000
USB_MXPLD4	USBD_BA+0x544	R/W	端点4最大负荷设置寄存器	0x0000_0000
USB_MXPLD5	USBD_BA+0x554	R/W	端点5最大负荷设置寄存器	0x0000_0000
USB_MXPLD6	USBD_BA+0x564	R/W	端点6最大负荷设置寄存器	0x0000_0000
USB_MXPLD7	USBD_BA+0x574	R/W	端点7最大负荷设置寄存器	0x0000_0000

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

位	描述	
[31:9]	保留	保留
[8:0]	**MXPLD**	**最大负荷** 定义了发送到主机(IN token)的实际数据长度，或 从主机接收到数据 (OUT token)的实际长度. 也用于指示IN token时做了发送数据准备，或OUT token时做好了接收数据准备. (1) 当CPU写入值后, 对 IN token, MXPLD寄存器的值用于指示要发送的数据长度，切已做好发送准备。 对OUT token,设备已经做好了从主机接收数据的准备。MXPLD的值就表示从主机所接收数据的实际长度。 (2) 当CPU读该寄存器时, 对IN token, MXPLD 的值表示要发送到主机的数据长度 对OUT token, MXPLD 的值表示从主机接收到的实际数据长度 **注意:** 一旦 MXPLD 的值被写入，收到IN/OUT token后.数据包可以立即收发

配置寄存器(USB_CFGx)

寄存器	偏移地址	R/W	描述	复位值
USB_CFG0	USBD_BA+0x508	R/W	端点0配置寄存器	0x0000_0000
USB_CFG1	USBD_BA+0x518	R/W	端点1配置寄存器	0x0000_0000
USB_CFG2	USBD_BA+0x528	R/W	端点2配置寄存器	0x0000_0000
USB_CFG3	USBD_BA+0x538	R/W	端点3配置寄存器	0x0000_0000
USB_CFG4	USBD_BA+0x548	R/W	端点4配置寄存器	0x0000_0000
USB_CFG5	USBD_BA+0x558	R/W	端点5配置寄存器	0x0000_0000
USB_CFG6	USBD_BA+0x568	R/W	端点6配置寄存器	0x0000_0000
USB_CFG7	USBD_BA+0x578	R/W	端点7配置寄存器	0x0000_0000

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

位	描述	
[31:10]	保留	保留
[9]	**CSTALL**	**清STALL Response** 0 =在setup阶段允许设备清除STALL 1 =在setup阶段禁止设备清除STALL
[8]	**保留**	保留
[7]	**DSQ_SYNC**	**数据时序同步** 0 = DATA0 PID. 1 = DATA1 PID. **注意:** 该位用于指定在接下来的IN token 传输过程是DATA0还是DATA1 PID. 在IN token过程，硬件会基于该位硬件自动触发选择
[6:5]	**STATE**	**端点状况** 00 =端点被禁止 01 =输出端点 10 =输入端点 11 =未定义.

[4]	**ISOCH**	**同步端点设置** 该位用于设置该端点为同步端点。无握手信号 0 = 非同步端点. 1 = 同步端点
[3:0]	**EP_NUM**	**端点号** 用于定义当前端点的端点号

扩展配置寄存器(USB_CFGPx)

寄存器	偏移地址	R/W	描述	复位值
USB_CFGP0	USBD_BA+0x50C	R/W	端点0设置Stall 和清In/Out 准备控制寄存器	0x0000_0000
USB_CFGP1	USBD_BA+0x51C	R/W	端点1设置Stall 和清In/Out 准备控制寄存器	0x0000_0000
USB_CFGP2	USBD_BA+0x52C	R/W	端点2设置Stall 和清In/Out 准备控制寄存器	0x0000_0000
USB_CFGP3	USBD_BA+0x53C	R/W	端点3设置Stall 和清In/Out 准备控制寄存器	0x0000_0000
USB_CFGP4	USBD_BA+0x54C	R/W	端点4设置Stall 和清In/Out 准备控制寄存器	0x0000_0000
USB_CFGP5	USBD_BA+0x55C	R/W	端点5设置Stall 和清In/Out 准备控制寄存器	0x0000_0000
USB_CFGP6	USBD_BA+0x56C	R/W	端点6设置Stall 和清In/Out 准备控制寄存器	0x0000_0000
USB_CFGP7	USBD_BA+0x57C	R/W	端点7设置Stall 和清In/Out 准备控制寄存器	0x0000_0000

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

位	描述	
[31:2]	保留	保留
[1]	**SSTALL**	**设置 STALL** 0 = 禁止设备响应STALL. 1 = 设置设备自动响应STALL
[0]	**CLRRDY**	**清除准备** 当USB_MXPLD寄存器被设置后，表示该端点准备好可以发送或接收数据。如果用户想在传输开始前关闭传输，需要设置该位为1来进行关闭，该位会自动清零。 对IN token，写 ‘1’ 清除 IN token 时发送数据到 USB 的准备信号 对OUT token，写 ‘1’ 清除 OUT token 时从 USB 接收数据的准备信号 该位只能写 1，读数据返回值总是 0。

USB 驱动SE0 寄存器 (USB_DRVSE0)

寄存器	偏移地址	R/W	描述	复位值
USB_DRVSE0	USBD_BA+0x090	R/W	USB 驱动SE0控制寄存器	0x0000_0001

31	30	29	28	27	26	25	24
保留							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
保留							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
保留							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
保留							**DRVSE0**

位	描述	
[31:1]	保留	保留
[0]	**DRVSE0**	**在USB总线上驱动SE0(Single Ended Zero)** Single Ended Zero (SE0)是把 (USB_D+ and USB_D-) 两根信号线都拉倒低 0 = 无. 1 = 迫使USB PHY 收发器驱动到SE0.状态

5.20 控制器局域网(CAN)

5.20.1 概述

C_CAN由CAN内核，报文RAM，报文处理器，控制寄存器和模块接口（参看图6-22）组成。CAN内核按照CAN协议2.0版本A部分和B部分规范执行通信。位速率最高可达 1Mbit/s。为与物理层相连，还需另外外接硬件收发器。

在CAN网络中，各个报文对象是可以独立配置的。报文对象和用于在接收时进行报文过滤的标识符掩码都存储在报文RAM中。所有与报文处理相关的功能都在报文处理器中执行。这些功能包括接收过滤、CAN内核与报文RAM之间的报文传输、处理传送请求以及模块中断的产生。

C-CAN的寄存器组可以通过模块接口被软件直接访问。这些寄存器用来控制/配置CAN内核和报文处理器，以及访问报文RAM。

5.20.2 特性

- 支持CAN协议 2.0版本 A和 B部分
- 位速率最高可达1 Mbit/s
- 32个报文对象
- 每个报文对象都有自己的标示符掩码
- 可编程FIFO模式（链接报文对象）
- 中断可屏蔽
- 禁用时间触发CAN应用下的自动重传模式
- 支持用于自检测的可编程环回模式
- 连接到AMBA APB总线上的16-位模块接口
- 支持唤醒功能

5.20.3 模块图

C_CAN与AMBA APB总线相接。图6-22为C_CAN的模块图。

- **CAN内核**

包括CAN协议控制器和用于报文串/并数据转换的RX/TX移位寄存器

- **报文RAM**

用于保存报文对象和标识符掩码 。

- **寄存器组**

包含所有控制和配置C_CAN的寄存器

- **报文处理器**

用于控制CAN内核的RX/TX移位寄存器和报文RAM间的数据传输的状态机，以及按照控制和配置寄存器中设定产生中断 。

- **模块接口**

C_CAN与来自ARM的AMBA APB16-位总线相接。

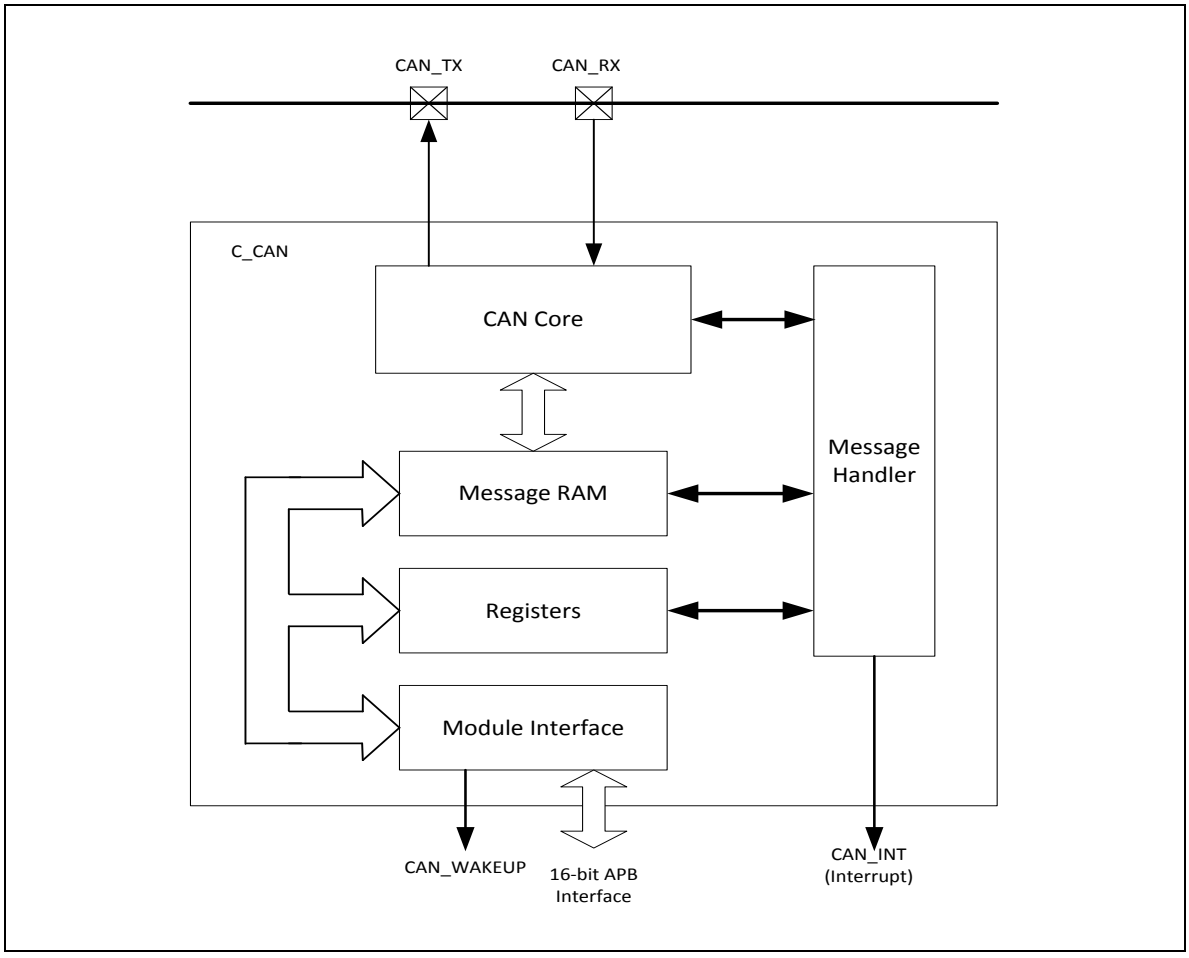


图 5-141 CAN 外设模块图

5.20.4 基本配置

CAN 的基本配置如下：

- 通过GPA_MFP 和 GPC_MFP 寄存器配置CAN信号管脚
- 使能CAN模块时钟（相关寄存器控制位为CAN0_EN (APBCLK[24]) 和 CAN1_EN (APBCLK[25])).
- 重启CAN控制器(相关寄存器控制位为CAN0_RST (IPRSTC2[24]) 和 CAN1_RST (IPRSTC2[25])).

5.20.5 功能描述

5.20.5.1 *软件初始化*

软件初始化可以通过软件或硬件复位，或者通过进入Bus_off状态置位CAN控制寄存器的Init位(CAN_CON[0])来实现。

当Init位被置位后，所有CAN总线上的报文传输都会停止，CAN_TX输出管脚上的状态为隐性（高电平）。错误管理逻辑（EML）计数器将保持不变。设置Init位不会改变任何配置寄存器。

为了初始化CAN控制器，软件必须设置位定时寄存器和每个报文对象。如果不需要用到某个报文对

象，那么必须清除相应的MsgVal 位 (CAN_IFn_ARB2[15])，否则，整个报文对象都需要进行初始化。

当CAN控制寄存器的Init和CCE位(CAN_CON[6])都被置位时，用于配置位定时的位定时寄存器和波特率预分频扩展寄存器的访问才会被使能

复位Init位（只能通过软件方式）完成软件初始化。接着位流处理器（BSP）（参看6.6.7.15：配置位定时）在等待 11个连续隐性位（总线空闲）的位序列后先将自身与CAN总线上数据实现同步，然后再参与总线活动和开始报文传输。

报文对象的初始化是独立于Init， 可以在忙碌时完成。但在BSP开始传输报文之前，所有报文对象的标识符必须先配置成指定值或设成无效。

在正常工作期间改变一个报文对象的配置时，软件必须先复位相应的MsgVal 位，然后再改变配置。当配置完成后，需再次设置MsgVal 位。

5.20.5.2 *CAN报文传输*

一旦C_CAN被初始化以及Init bit (CAN_CON[0])位被重置为0，C_CAN内核就会将自身与CAN总线同步，并开始报文传输。

如果接收到的报文通过了报文处理器的接收过滤，将会被存储在相应的报文对象中。保存在报文对象中的报文包括所有的仲裁位，DLC(CAN_IFn_MCON[3:0])和8个字节数据(CAN_IFn_DAT_A1/2；CAN_IFn_DAT_B1/2)。如果使用了标识符掩码,那些被掩码为”无关”位的仲裁位会在报文对象中被覆盖。

软件可以通过接口寄存器在任何时间去读或写每条报文。如果同时访问，报文处理器将保证数据的一致性。

发送的报文由应用软件进行更新。如果一个报文永久性的存在某个报文对象中（仲裁位和控制位在配置时被设定），那么仅有数据字节会被更新，在置位TxRqst位(CAN_IFn_MCON[8]) 和NewDat(CAN_IFn_MCON[15])位会后就会开始传送。如果几条传输报文被指派给同一个报文对象(当报文对象的数量不够用时)，那么在发送这些报文前必须对整个报文对象进行配置。

任何数量的报文对象都可以在同一时间请求传送。报文对象会按照它们的内部优先级进行依次传送。报文可以随时更新或设置为无效，甚至在发送请求还在等待阶段也可进行。如果一条报文在其未发送前被更新，则旧的数据将会被丢弃。

根据报文对象的配置，在接收到匹配标识符的遥控帧后，会自动产生报文发送请求。

5.20.5.3 *禁用自动重传*

依照CAN协议规范（见ISO11898，6.3.3恢复管理），C_CAN 为那些在传送过程中丢失仲裁或被错误干扰的帧，提供自动重传机制。在传送完全成功之前，帧传送服务是不能证实给用户。这也就意味着，默认情况下，自动重传是使能的。在C_CAN工作在时间触发CAN（TTCAN，见ISO11898-1）环境时，可以禁用自动重传功能。

通过设置CAN控制器中的禁用自动重传（DAR(CAN_CON[5])）位为1来禁用自动重传模式。在这种操作模式下，用户必须考虑报文缓存控制寄存器中TxRqst(CAN_IFn_MCON[8]) 和NewDat(CAN_IFn_MCON[15])位不同设置时的状况：

- 当传送开始时，各个报文缓存的TxRqst位被清除，而NewDat位一直置位
- 当传送完全成功时，NewDat位被清除
- 当传送失败（丢失仲裁或发生错误），NewDat位保持置位

- 为了重新开始传送，软件必须再将TxRqst位置位

5.20.6 测试模式

设定CAN控制寄存器的Test(CAN_CON[7])位进入测试模式。在测试模式下，测试寄存器的Tx1 (CAN_TEST[6])，Tx0 (CAN_TEST[5])， LBack (CAN_TEST[4]),，Silent (CAN_TEST[3]) 和 Basic (CAN_TEST[2])位可写。Rx位监视CAN_RX管脚的状态，因此该位只读。当Test位被清除时，所有的测试寄存器功能被禁用。

5.20.6.1 *静默模式*

通过设置测试寄存器中的Silent位(CAN_TEST[3])为1，CAN内核可被设为Silent模式。在Silent模式中，C_CAN能够接收有效的数据帧和有效的遥控帧，但是它在CAN总线上仅发送隐性位，而且它不能启动发送。如果CAN内核被要求发送一个显性位（ACK位，错误帧），那么该显性位将在内部自动改道以便CAN内核检测到该显性位，而CAN总线仍然保持在隐性状态。因为可以在传送显性位时不会影响到CAN总线，所以Silent模式可以用来分析CAN总线的运输状况。图6-23为静默模式下，CAN_TX和CAN_RX与CAN内核的连接信号。

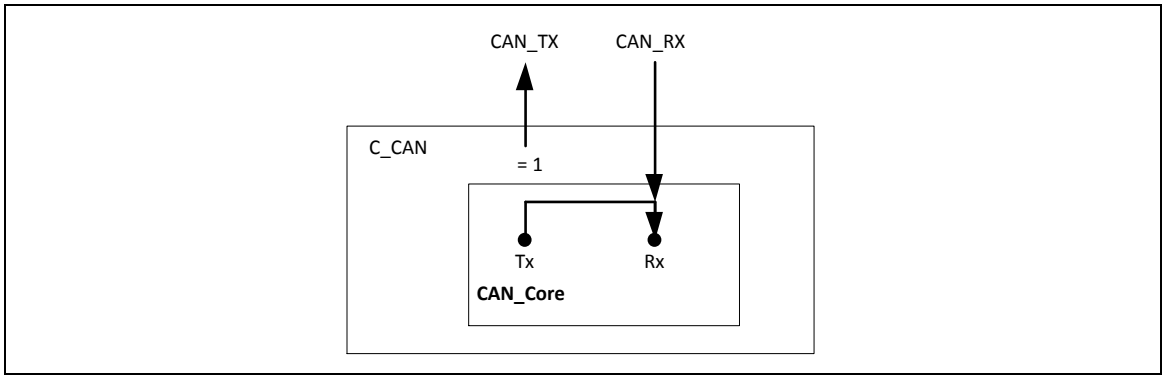


图 5-142 CAN 内核静默模式

5.20.6.2 *环回模式*

通过设定测试寄存器的LBack位(CAN_TEST[4])为1，CAN内核可被设为Loop Back模式。在Loop Back模式下，CAN内核把自己发送的报文作为接收到的报文对待，并将它们保存在接收缓存中（如果它们通过了接收过滤）。图6-24为Loop Back模式下，CAN_TX和CAN_RX 与CAN内核的连接信号。

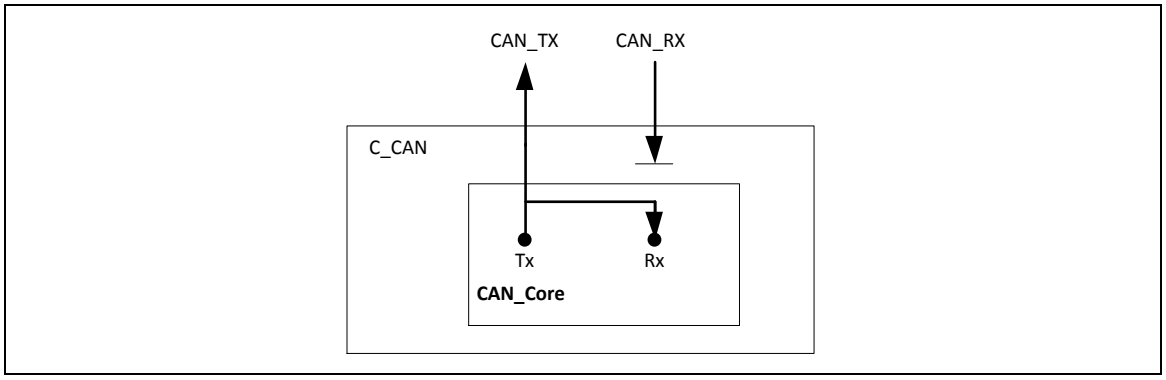


图 5-143 CAN 内核环回模式

该模式用于自检。为了不受外部的干扰，在Loop Back模式下，CAN内核忽略应答错误（数据/远程

帧应答槽内采样到隐性位）。在该模式下，CAN内核的Tx输出通过内部直接反馈到它的Rx输入上。CAN_RX输入管脚的真实值则被CAN内核忽略。发送的报文信号仍可通过 CAN_TX管脚进行监测。

5.20.6.3 *环回模式和静默默式的组合*

通过设定LBack(CAN_TEST[4])和Silent(CAN_TEST[3])位同时为1，还可以将Loop Back模式和Silent模式结合一起使用。该模式可用于“热自测 （Hot Selftest）”，即C_CAN可以在不影响已与CAN_TX 和CAN_RX 管脚相连的CAN系统的情况下进行测试。在该模式下，CAN_RX管脚与CAN内核断开，CAN_TX管脚保持隐性电平。图6-25为环回模式和静默模式整合下，CAN_TX 和CAN_RX与CAN内核的连接信号。

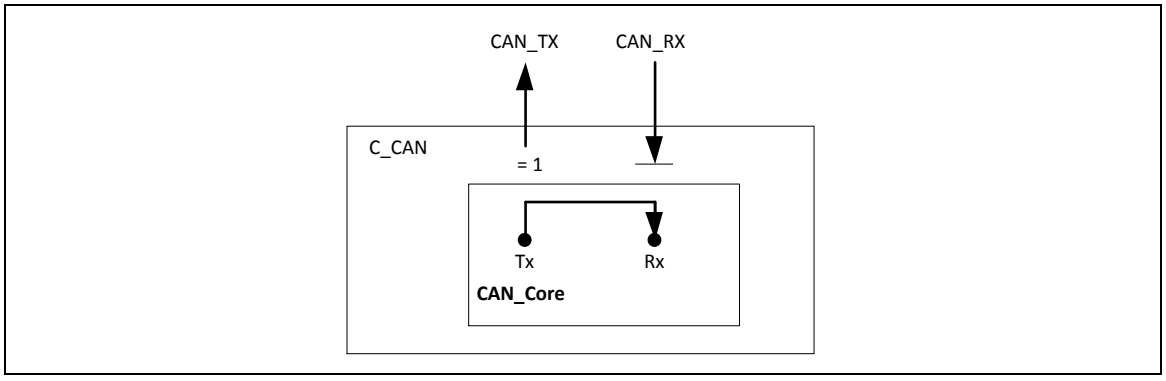


图 5-144 CAN 内核环回模式和静默模式的组合

5.20.6.4 *基本模式*

通过设定测试寄存器的Basic位(CAN_TEST[2])为1可以设定CAN内核工作在Basic模式。该模式下C_CAN工作时没有报文RAM。

IF1 寄存器组用作发送缓存。IF1 寄存器组中内容通过设定IF1命令请求寄存器中的Busy位(CAN_IFn_CREQ[15])为1来请求发送。当Busy位置位时，IF1 寄存器组锁定。Busy位指示传送正在挂起。

一旦CAN总线空闲，IF1 寄存器组会被载入到CAN内核的移位寄存器，然后传送开始。当传输完成时，Busy位复位，并释放锁定的IF1寄存器组。

当IF1寄存器组锁定时，挂起的发送可以被随时中止，方法为复位IF1的命令请求寄存器的Busy位。如果软件已经复位了Busy位，那么为防止仲裁丢失或错误出现的重传是被禁用的。

IF2寄存器组用作接收缓存。收到报文后，移位寄存器的内容不经过任何接受过滤就存储到IF2寄存器组。

此外，在报文传输的过程中，移位寄存器的真实内容能够被监视。每次写IF2 命令请求寄存器中的Busy位为1，读报文对象就会被初始化，移位寄存器的内容则被保存在IF2寄存器组。

在Basic模式下，所有与报文对象相关的控制和状态位以及IFn掩码寄存器的控制位都不能进行赋值。命令请求寄存器的报文编号也不能赋值。IF2 报文控制寄存器的NewDat(CAN_IFn_MCON[15])和MsgLst(CAN_IFn_MCON[14])位保留其功能。DLC3-0指示接收到的DLC(CAN_IFn_MCON[[3:0])，其他控制位读取值皆为‘0’。

5.20.6.5 *软件控制CAN_TX管脚*

CAN发送管脚CAN_TX上有四种输出功能可用。除了缺省功能（串行数据输出），CAN发送管脚还

能驱动输出CAN采样点信号，用于监视CAN内核的位定时，而且CAN发送管脚还能持续驱动输出显性或隐性电平。后两种功能，结合可读的CAN接收管脚CAN_RX，能够用于检测CAN总线的物理层。

CAN_TX管脚的输出模式通过设定CAN测试寄存器的Tx1和Tx0位进行选择。

三种CAN_TX管脚测试功能参与了所有的CAN协议功能。当CAN进行报文传输或者选择任一测试模式（Loop Back Mode，Silent Mode,或Basic Mode）时，CAN_TX 必须使用它的默认功能。

5.20.7 CAN 通信

5.20.7.1 *管理报文对象*

报文RAM中的报文对象配置（除CAN控制寄存器的MsgVal, NewDat, IntPnd, 和TxRqst位外)不会受芯片复位影响。在应用软件清除Init位之前，所有的报文对象必须被应用软件初始化或是设置为 “无效”（MsgVal=0），而且位时序必须在应用软件清Init位(CAN_CON[0])之前被配置好。

通过配置两个接口寄存器中一个接口寄存器的屏蔽、仲裁、控制和数据域为期望值后，报文对象的配置完成。通过写相应的IFn命令请求寄存器，IFn报文缓存寄存器组会被载入到报文RAM中指定地址的报文对象中。

CAN控制寄存器的Init位被清除后，CAN内核的CAN协议控制状态机和报文处理器状态机控制C_CAN的内部数据流。收到的报文通过接收过滤后被保存在报文RAM中。挂起发送请求的报文被载入到CAN内核的移位寄存器，并通过CAN总线进行发送。

应用软件通过IFn接口寄存器组读取收到的报文以及更新发送的报文。依照配置，应用软件会被特定的CAN报文和CAN错误事件打断。

5.20.7.2 *报文处理器状态机*

报文处理器控制CAN内核的Rx/Tx移位寄存器、报文RAM和IFn寄存器组之间的数据传输。

报文处理器 FSM控制如下功能：

- 从IFn寄存器传送数据到报文RAM
- 从报文RAM传送数据到IFn寄存器
- 从移位寄存器传送数据到报文RAM
- 从报文RAM传送数据到移位寄存器
- 从移位寄存器传送数据到接收过滤单元
- 扫描报文RAM寻找匹配报文对象
- 处理TxRqst标志位
- 处理中断

5.20.7.3 *报文RAM的数据传输*

当应用软件对IFn寄存器组和报文RAM间的数据传输初始化后，报文处理器设置各自命令请求寄存器的Busy位（CAN_IFn_CREQ[15]）为‘1’。在传输完成后，Busy位会被再次清除（见图6-26）。

命令掩码寄存器可以指定是一个完整的报文对象还是报文对象的部分内容将被传输。因为报文RAM的结构，写操作不能仅仅针对报文对象的单独某位/字节，而是必须写入完整的报文对象到报文RAM。因此，从IFn寄存器到报文RAM的数据传输需要一个读-修改-写的周期。首先，将报文对象中不变的

部分从报文RAM读出，然后将报文缓存寄存器的完整报文内容写入到报文对象。

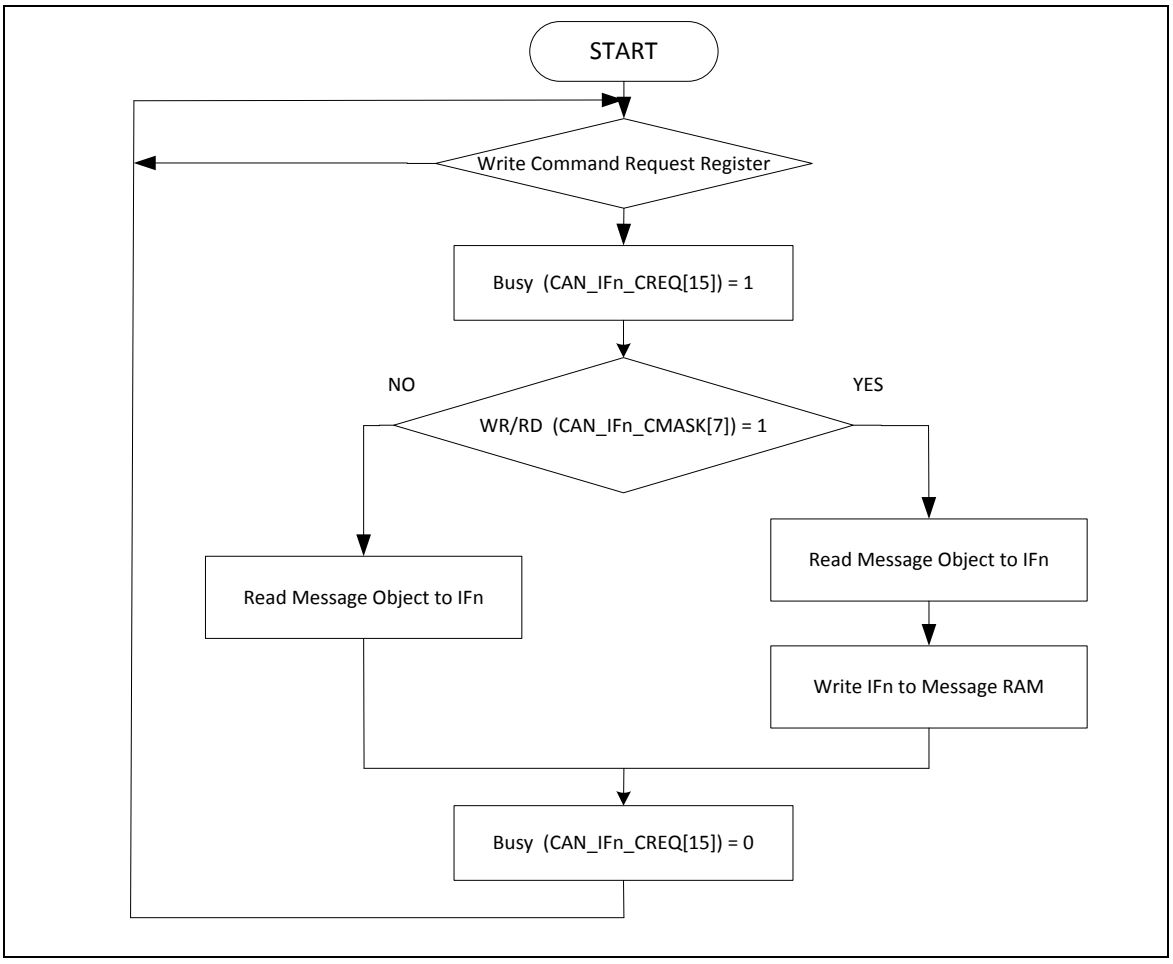


图 5-145 IFn寄存器和报文RAM间的数据传输

在写入一个报文对象部分内容后，没被命令掩码寄存器选中的报文缓存寄存器将设置选中报文对象的实际内容。

在读取一个报文对象部分内容后，没被命令掩码寄存器选中的报文缓存寄存器将保持不变

5.20.7.4 *报文传送*

如果CAN内核单元的移位寄存器准备载入，但在IFn寄存器组和报文RAM之间没有数据传输，内核将重新评估MsgVal 位(CAN_IFn_ARB2[15]) 和TxRqst位(CAN_TXREQ1/2)。具有最高优先级发送请求的有效报文对象将被报文处理器载入到移位寄存器，开始发送。报文对象的NewDat (CAN_IFn_MCON[15])位被复位。

成功传送后或者自传送开始后没有新的数据写入报文对象(NewDat = ‘0’)，报文控制寄存器TxRqst位 (CAN_IFn_MCON[8])将被复位。如果报文控制寄存器TxIE位(CAN_IFn_MCON[11])被置位，中断标识符寄存器的IntPnd 位(CAN_IFn_MCON[13])在传送成功后将被置位。如果C_CAN仲裁失败或者在传送过程中有错误发生，报文将会在CAN总线空闲的时候进行重传。同时，如果有更高优先级的报文传送请求，则报文将会按照报文的优先级顺序进行传送。

5.20.7.5 *收到报文的接收过滤*

当一个输入报文的仲裁和控制域(Identifier + IDE + RTR + DLC)完全被移位到CAN内核的Rx/Tx移位寄存器时，报文处理器FSM开始扫描报文RAM，寻找匹配的有效报文对象。

扫描报文RAM寻找匹配的报文对象，需要将CAN内核移位寄存器的仲裁位载入到接收过滤单元。步骤为：先载入报文对象1的仲裁和掩码域（包括MsgVal (CAN_IFn_ARB2[15]), UMask (CAN_IFn_MCON[12]), NewDat (CAN_IFn_MCON[15]), 和EoB (CAN_IFn_MCON[7])）到接收过滤单元，再和移位寄存器的仲裁位进行比较。该步骤对接下来的每一个报文对象重复进行，直到匹配的报文对象出现或者到达报文RAM的末端。

如果匹配发生，扫描停止，报文处理器FSM将根据接收帧的类型（数据帧或远程帧）进行处理。

接收数据帧

报文处理器FSM将CAN内核移位寄存器中报文保存到报文RAM中的各个报文对象。不仅数据字节，所有仲裁位和数据长度码都被保存到相应的报文对象中。这是为了将数据字节与标识符继续保持关联性。

置位NewDat 位(CAN_IFn_MCON[15])用于指示新数据（还没有被软件看到的）已经收到了。当报文对象被读取后，应用软件必须复位NewDat 位(CAN_IFn_MCON[15])。如果在接收的时候，NewDat位已经被置位，MsgLst (CAN_IFn_MCON[14])用来指示之前的数据（假定还没有被软件看到）已丢失。如果RxIE位 (CAN_IFn_MCON[10])被置位，IntPnd 位(CAN_IFn_MCON[13])也置位将使中断寄存器指向该报文对象。

当请求的数据帧刚刚收到时，报文对象的TxRqst(CAN_IFn_MCON[8])位会被复位用来阻止传送远程帧。

接收远程帧

当收到一个远程帧时，必须考虑匹配的报文对象的三种配置：

1) Dir (CAN_IFn_ARB2[13]) = ‘1’ (direction = transmit), RmtEn (CAN_IFn_MCON[9]) = ‘1’, UMask (CAN_IFn_MCON[12]) = ‘1’或’0’

当收到匹配的远程帧时，报文对象的TxRqst位被置位。报文对象的其他部分保持不变。

2) Dir = ‘1’ (direction = transmit), RmtEn = ‘0’, UMask =’0’

当收到匹配的远程帧时，报文对象的TxRqst位保持不变，远程帧被忽略。

3) Dir = ‘1’ (direction = transmit), RmtEn = ‘0’, UMask =’1’

当收到匹配的远程帧时，报文对象的TxRqst位被复位。移位寄存器中的仲裁和控制域(Identifier + IDE + RTR + DLC)被保存到报文RAM中的报文对象，报文对象的NewDat (CAN_IFn_MCON[15])位被置位。报文对象的数据域保持不变。收到的远程帧处理方式与数据帧类似。

5.20.7.6 *接收/发送优先级*

报文对象接收/发送的优先级与报文号相关。报文对象1拥有最高优先级，报文对象32拥有最低优先级。如果有一个以上的发送请求挂起时，它们将根据相应报文对象的优先级来进行服务。

5.20.7.7 *配置传送对象*

表6-6列出如何初始化一个发送对象。

Ms	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

表 5-24 初始化发送对象

注意： appl.=应用软件

仲裁寄存器(ID28-0 (CAN_IFn_ARB1/2) 和Xtd 位(CAN_IFn_ARB2[14]))由应用软件设置。它们定义标识符和发出的报文类型。如果11-位标识符（标准帧）被使用，它将被编程到ID28 - ID18。ID17 - ID0可以被忽略。

如果TxIE(CAN_IFn_MCON[11])位被置位，在报文对象成功发送之后，IntPnd(CAN_IFn_MCON[13])位将被置位。

如果RmtEn(CAN_IFn_MCON[9])位被置位，收到匹配的远程帧将置TxRqst(CAN_IFn_MCON[8])位，而且该远程帧会自动被一个数据帧应答。

数据寄存器的值(DLC3-0 (CAN_IFn_MCON[3:0]) ，Data0-7)由应用软件提供。在数据有效之前，TxRqst和RmtEn可能不会被置位。

(UMask (CAN_IFn_MCON[12]) = ’1’)时,掩码寄存器(Msk28-0, UMask, MXtd, 和 MDir bits)可以用于允许有相似标识符的远程帧组设置TxRqst位。但Dir (CAN_IFn_ARB2[13])位不能被屏蔽。

5.20.7.8 *更新传送对象*

不必复位MsgVal (CAN_IFn_ARB2[15])和TxRqst(CAN_IFn_MCON[8])，软件就可以通过IFn接口寄存器随时更新发送对象的数据字节。

即使只更新部分数据字节，在内容传输给报文对象之前，IFn数据A寄存器或B寄存器中的所有4个字节数据都必须有效。应用软件必须写所有4个字节到IFn数据寄存器或者在软件写新的数据字节前报文对象已被传输到IFn数据寄存器。

仅当（8个）数据字节更新时，先写0x0087到命令掩码寄存器，然后再写报文对象的编号到命令请求寄存器，同时更新数据字节及设置TxRqst.

当数据更新时，为防止在传输的最后阶段TxRqst复位，必须同时置位NewDat (CAN_IFn_MCON[15])和TxRqst。

当NewDat和TxRqst一起被置位时，NewDat将会在新的传送开始时立即被复位。

5.20.7.9 *配置接收对象*

表6-7为如何初始化接收对象。

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

表 5-25 初始化接收对象

仲裁寄存器的值(ID28-0 (CAN_IFn_ARB1/2) 和 Xtd 位 (CAN_IFn_ARB2[14]))由应用软件设置。它们定义标识符和接收报文的类型。如果11-位标识符(“标准帧”)被使用，它将会编程到ID28 - ID18。ID17 - ID0可以被忽略。当带有11-位标识符的数据帧被收到时，ID17 - ID0将被置0。

如果RxIE(CAN_IFn_MCON[10])位置位，当接收到数据帧并被保存到报文对象时，IntPnd (CAN_IFn_MCON[13])位将被置位。

数据长度码（(DLC3-0 (CAN_IFn_MCON[3:0])由应用软件设置。当报文处理器保存一个数据帧到报文对象时，它将会保存收到的数据长度码和8数据字节。如果数据长度码少于8，则报文对象余下的字节将被未指定的值覆盖。

(UMask (CAN_IFn_MCON[12]) = ’1’)时,掩码寄存器(Msk28-0, UMask, MXtd, 和MDir)可以被用于允许有相似标识符的数据帧组被接收。在典型的应用中，Dir (CAN_IFn_ARB2[13])位不能被屏蔽。

5.20.7.10 *处理接收报文*

应用软件可以通过IFn接口寄存器随时读一条收到报文。报文处理器的状态机将保证数据的一致性。

通常的，软件先写0x007F到命令掩码寄存器，然后写报文对象编号到命令请求寄存器。这就会将收到的报文整个从报文RAM传输到报文缓存寄存器。并且，在报文RAM(不是报文缓存)中NewDat (CAN_IFn_MCON[15])和 IntPnd (CAN_IFn_MCON[13])位将被清除。

如果报文对象使用掩码进行接收过滤，则仲裁位表示哪条匹配的报文已经收到了。NewDat的真实值（当前值）表示自上次该报文对象被读取后，是否收到新的报文。MsgLst (CAN_IFn_MCON[14])的真实值表示自上次该报文对象被读取后，收到的报文是否超过1条。MsgLst不会被自动复位。

利用远程帧，软件可以请求另一个CAN节点为接收报文对象提供新的数据。设定接收报文对象的TxRqst(CAN_IFn_MCON[8])位将引发带有接收报文对象的标识符的远程帧进行传送。该远程帧触发另一个CAN节点开始发送与之匹配的数据帧。如果在发送远程帧之前收到匹配的数据帧，TxRqst位将被自动复位。

5.20.7.11 *配置FIFO缓存*

除EoB(CAN_IFn_MCON[7])位外，FIFO缓存的接收报文对象群的配置和单个接收报文对象的配置是一样的。参看章节5.13.6.5：配置接收对象

为在FIFO缓存中关联2个或更多报文对象，这些报文对象的标识符和掩码（如使用）必须设置为要匹配的值。因为报文对象的绝对优先级，最低编号的报文对象将会是FIFO缓存的第一个报文对象。FIFO缓存中（除了最后的报文对象外）所有报文对象的EoB位都必须设置为0。FIFO缓存中的最后一个报文对象的EoB位需设置为1，表示模块的结束。

5.20.7.12 *接收带FIFO缓存的报文*

收到的报文如果匹配FIFO缓存中的标识符会被保存到该FIFO缓存中，报文存储的顺序为从最低报文编号的报文对象开始。

当一条报文保存到FIFO缓存的报文对象中时，该报文对象的NewDat(CAN_IFn_MCON[15])位被置位。当EoB(CAN_IFn_MCON[7])为0时通过置位NewDat，报文对象将被锁定用于之后 的报文处理器的写访问，直到应用软件将NewDat位写回0。

报文群将一直保存在FIFO缓存中直到FIFO缓存的最后一个报文对象。如果之前没有一个报文对象通过写NewDat位为0释放，则FIFO缓存之后收到的报文都将被写入到该FIFO缓存的最后一个报文对象中，当然之前的报文会被覆盖。

5.20.7.13 *从FIFO缓存中读取数据*

当应用软件通过写报文对象编号到IFn命令请求寄存器将报文对象的内容传输到IFn报文缓存寄存器时，相应的命令掩码寄存器设置为：(TxRqst/NewDat (CAN_IFn_CMASK[2]) = ‘1’ 和 ClrIntPnd (CAN_IFn_CMASK[3]) = ‘1’)时，NewDat (CAN_IFn_MCON[15]) 和 IntPnd (CAN_IFn_MCON[13]) 位复位为零。报文控制寄存器以上各个的寄存器位一直反映其状态。

为保证FIFO缓存的正确使用，应用软件须从FIFO缓存中最低报文编号的报文对象读起。

图6-27表示应用软件如何处理一组和FIFO缓存相连的报文对象。

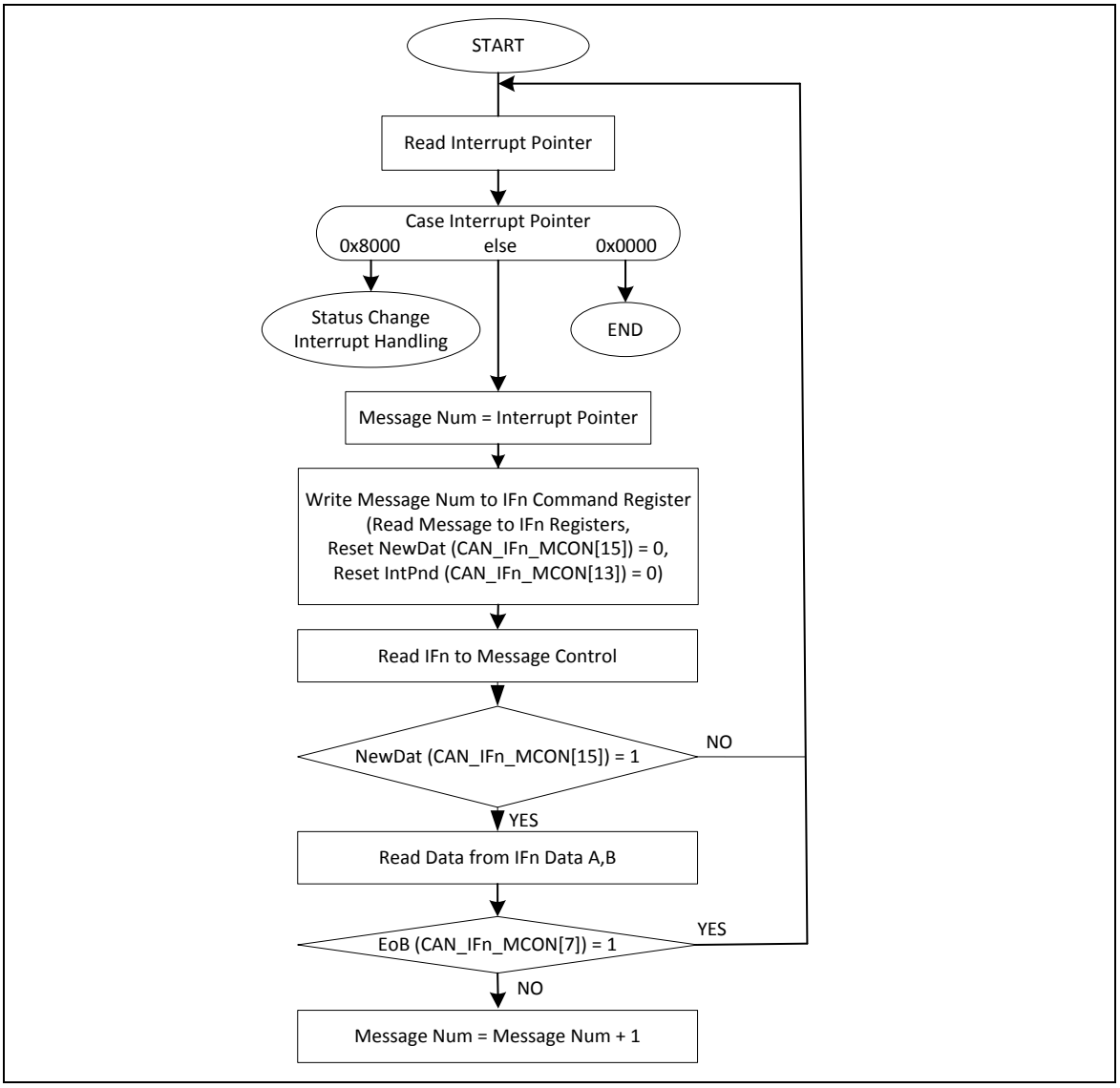


图 5-146 应用软件处理FIFO缓存

5.20.7.14 中断处理

如果几个中断同时挂起,则CAN的中断寄存器将指向具有最高优先级的挂起中断，而无需理会他们的时序。一个中断会保持挂起状态直到应用软件清除它。

状态中断拥有最高优先级。在报文中断中，报文对象的中断优先级随着报文编号的增加而降低。

报文中断通过清除报文对象的IntPnd(CAN_IFn_MCON[13])位进行清除。状态中断通过读状态寄存器清除。

中断寄存器的中断标识符Intld指示引发中断的原因。当没有中断挂起时，寄存器将保持为0。如果中断寄存器的值不为0，则有中断挂起。如果IE(CAN_IFn_CON[1])被置位，则CAN_INT中断信号被激活。中断保持激活直到中断寄存器归零（中断源被复位）或直到IE被复位。

值0x8000指示有中断正被挂起，原因是CAN内核更新了（不是必须改变）状态寄存器（错误中断或状态中断）。.该中断具有最高优先级，应用软件可以更新（复位）状态位RxOk (CAN_STATUS[4]), TxOk (CAN_STATUS[3]) 和 LEC (CAN_STATUS[2:0])，但是软件对状态寄存器的写访问不会产生或复位中断。

其他的值表示中断源是其中一个报文对象。Intld指向挂起中断中的具有最高中断优先级的报文中断。

应用软件决定改变状态寄存器是否可能导致产生中断（EIE (CAN_IFn_MCON[3]) 和 SIE (CAN_IFn_MCON[2])），以及当中断寄存器的值不等于零时中断线是否需要激活（CAN控制寄存器的IE位）。即使IE复位，中断寄存器仍将会更新。

应用软件跟踪报文中断源有2种可能方式。一种是通过查看中断寄存器的Intld位，第二种是轮询中断挂起寄存器。

中断处理服务惯例为：读触发中断源的报文并在读报文的同时复位报文对象的IntPnd（ClrIntPnd(CAN_IFn_CMASK[3])) 位。当IntPnd被清除，中断寄存器将指向下一个带有挂起中断的报文对象。

5.20.7.15 *配置位定时*

CAN位定时配置的小错误不会导致通讯立即失败，但是会显著降低CAN网络的整体性能。

很多情况下，CAN位同步可以修补CAN位定时的错误配置，而这种错误有可能仅仅是偶发性发生一个错误帧。但是，在仲裁时，当两个或更多CAN节点同时试着发送一帧，那么一个错位的采样点可能会导致其中一个传送器变成被动错误状态。

如果要分析此类分散性错误， 需要对CAN节点内CAN位同步以及CAN总线上CAN节点间相互作用有详细的了解。

5.20.7.16 **位时间和位速率**

CAN支持的位速率低至1 kb/s，高至1000 kb/s。CAN网络中的每个成员都有自己的时间发生器，通常是石英振荡器。位时间的定时参数（例如：位速率的倒数）可以单独配置给每个CAN节点，这样尽管CAN节点的振荡器周期(f_{osc})可能不同，仍可创建一个公共的位速率。

这些振荡器的频率不是绝对的稳定，温度或电压的变化或者组件恶化会导致一些小的变化。只要这种变化保持在指定的振荡器容差范围（df）内，CAN节点就能够通过位流重同步对不同的位速率进行补偿。

依据CAN规范，位时间分为四段：同步段，传播时间段，相位缓存段1和相位缓存段2（见图6-28）。每段由一个特定的、可编程的时间片组成（见表6-8）。时间片的长度(t_q),是位时间的基本时间单元，由CAN控制器的APB时钟f_{APB}和位定位寄存器（CAN_BTR）的BRP位定义：(CAN_BTIME[5:0]) : t_q =

BRP / f_{APB}.

同步段Sync_Seg，是位时间的一部分，是CAN总线电平跳变边沿发生的地方。Sync_Seg之外发生跳变边沿和Sync_Seg之间的距离称为边沿的相位误差。传播时间段Prop_Seg用于补偿CAN网络内的物理延时时间。相位缓存段Phase_Seg1和Phase_Seg2围绕着采样点。(重)同步跳转宽度(SJW)定义了重同步将采样点在相位缓存段定义的范围内可能移动的距离，此用于补偿边沿相位误差。

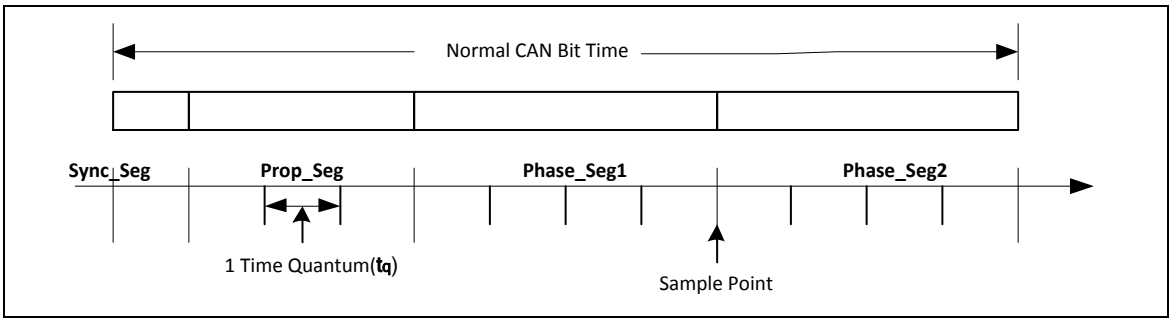


图 5-147位时间

参数	范围	备注
BRP	[1 .. 32]	定义时间片的长度t_q
Sync_Seg	1 t_q	固定长度，同步输入总线到APB时钟
Prop_Seg	[1.. 8] t_q	补偿物理延时时间
Phase_Seg1	[1..8] t_q	允许同步暂时延长
Phase_Seg2	[1.. 8] t_q	允许同步暂时缩短
SJW	[1 .. 4] t_q	不能比任一个相位缓存段长
该表格描述了CAN协议要求的最小可编程范围		

表 5-26 CAN 位时间参数

一个给定的位速率可以有不同的位时间配置，但是对于功能正常的CAN网络，物理延时时间和振荡器容错范围必须考虑。

5.20.7.17 传播时间段

该部分位时间用于补偿网络的物理延时时间。这些延时是由总线上的信号传播时间和CAN节点的内部延时时间组成。

CAN总线上，CAN节点位流将和传送端位流不同步，这是由两个节点间的传播时间造成的。CAN协议的非破坏性的逐位仲裁和CAN 报文接收器发出的显性响应位，要求负责发送位流的CAN节点必须同时能够接收其他同步到该位流的其它CAN 节点发送的显性位。图6-29的示例表示连两个CAN节点间的相位移动和传播时间。

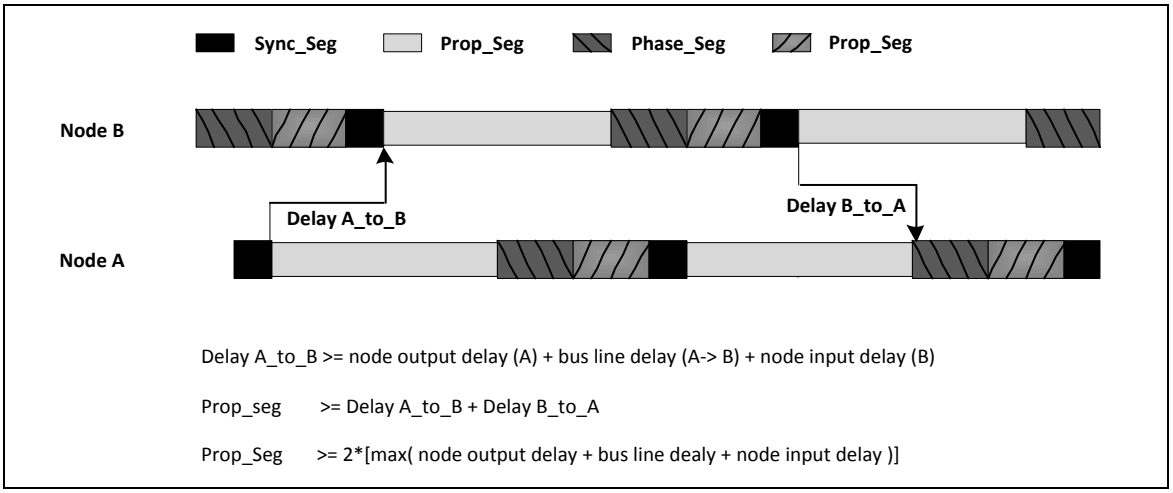


图 5-148 传播时间段

该示例中，节点A和B都是发送器，CAN 总线执行仲裁。节点A早于节点B发送它的起始帧位，时差小于1个位时间，因此节点B同步自身到从隐性到显性的接收边沿。因为B节点是在发送后才收到这个边沿延时(A_to_B)，所以B的位时间段须参照A移位。B节点发送的是一个高优先级的标识符，所以在B传送一个显性位，而A发送一个隐性位时，B将在这个标识位获得仲裁。节点B发送的显性位将延时(B_to_A)后到达在A。

由于振荡器的偏差，节点A采样点的实际位置可以在节点A相位缓存段标称的任何位置。所以节点B的位传送必须在Phase_Seg1开始之前到达节点A。该条件决定Prop_Seg的长度。

如果节点B传输信号的隐性到显性边沿在Phase_Seg1开始后才到达节点A，那么可能会出现节点A采样结果为隐性位而不是显性位的情况，这将导致一个位错误和错误标志并破坏当前帧。

这种错误只在两个节点都要求CAN总线进行仲裁，而两个节点的振荡器的偏差方向相反，处于长总线两端的情况下才会发生。这是一个位定时配置错误(Prop_Seg 定义时间太短)导致偶发性总线错误的例子。

一些CAN硬件提供可选择的3次采样模式，但是C_CAN没有。在该模式下，CAN总线的输入信号通过一个低通滤波器，使用3个样本并根据大数逻辑来决定有效位值。这会导致额外的1个输入延时t_q，因而需要更长的Prop_Seg。

5.20.7.18 相位缓存段和同步

相位缓存段(Phase_Seg1 和 Phase_Seg2)和同步跳转宽度（SJW）用于振荡器误差补偿。通过同步，可以延长或缩短相位缓存段。

同步发生在从隐性到显性的边沿，目的是用来控制边沿和采样点之间的距离。

边沿检测通过采样每个时间片的总线电平，并与前一个采样点的总线电平进行比较。同步仅发生在前一个采样点为隐性，而当前时间片的总线电平为显性时。

如果边沿发生在Sync_Seg里面，那么它是处于同步状态，否则边沿和Sync_Seg结束处之间的距离即为边沿相位误差，时间片为时间误差单位。如果边沿在Sync_Seg之前发生，相位误差为负，其他情况则为正。

存在两种同步类型：硬同步和重同步。

硬同步在帧一开始时就会进行，而在帧内进行的是重同步。

- **硬同步**

硬同步之后，位时间将在Sync_Seg的结束处重新开始，而不用管边沿相位误差。因此硬同步强迫导致硬同步的边沿处于重新开始的位时间同步段内。

- **位重同步**

位重同步导致位时间的缩短或延长，以致采样点位置随边沿移动 。

当引发重同步的边沿相位误差是正的，Phase_Seg1延长。如果相位误差的值少于SJW，Phase_Seg1由相位误差的值延长，否则SJW延长。

当造成重同步的边沿相位误差是负的，Phase_Seg2缩短。如果相位误差的值少于SJW，Phase_Seg2由相位误差的值缩短，否则SJW缩短。

当边沿相位误差的值小于或者等于SJW的设定值，硬同步和重同步作用相同。如果相位误差大于SJW，则重同步无法完全补偿相位误差，仍有误差（相位误差-SJW）存在。

在连个采样点间只能完成一个同步，同步维持一个在边沿和采样点间最小的距离，给总线电平稳定和 过滤出小于(Prop_Seg + Phase_Seg1)的峰值的时间。

除了噪声脉冲，多数同步是由仲裁导致的。所有节点会“硬”同步在“领先”收送器（最先发送数据的收发器）的传送边沿，但是由于传播延时，这些节点不能达到理想的同步。“领头”发送器不必获得仲裁，因此各接收器必须将自身同步到随后的“夺得领先”发送器(不同于之前的“领头”发生器)。同样的情况也发生在响应域，发送器和一些接收器必须要与在传送显性响应位时“夺得领先”的接收器取得同步。

当发送器和接收振荡器时钟周期的差异点在同步时间内（最多10位）累加时，在仲裁最后的同步将由振荡器误差造成。这些累加的差异不能比SJW长，限定了振荡器误差的范围。

图6-30为相位缓存段是如何用于相位误差补偿的示例，其内有三张有两个连续的位定时的图。最上面一张表示同步在“late”边沿，最下面一张表示同步在“early”边沿，中间一张为无同步的参照图。

n.

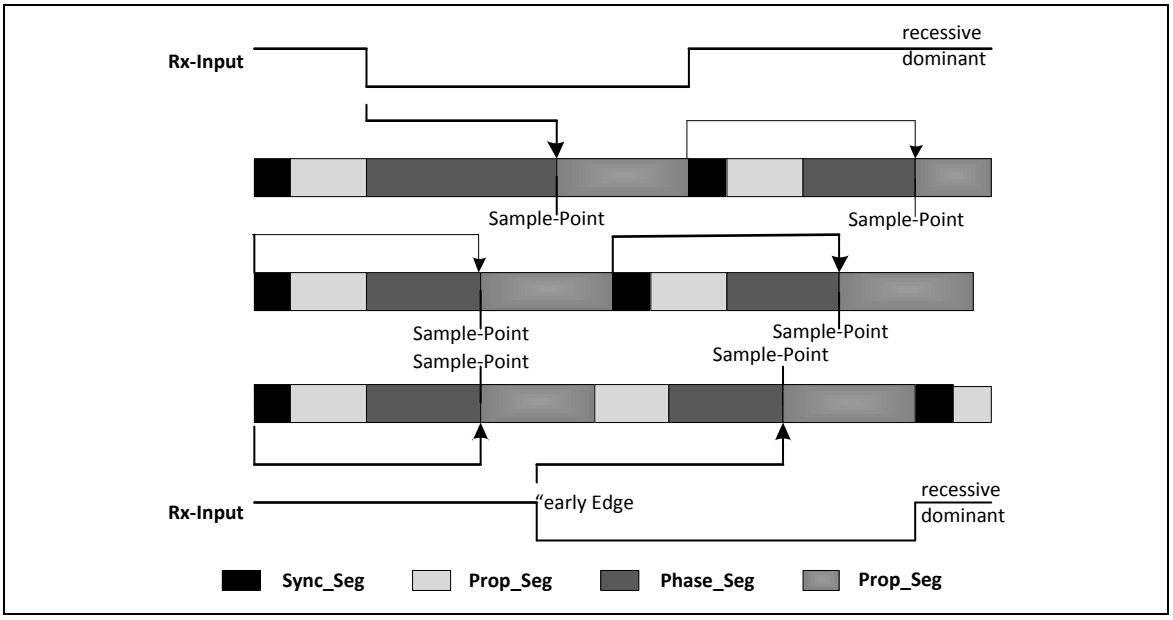


图 5-149 同步在“late” 和 “early” 边沿

在第一个示例中隐性到显性的跳变边沿发生在Prop_Seg尾端。该边沿称之为“late”，因为它发生在Sync_Seg之后。与“late”边沿的相对应，Phase_Seg1被延长，以使边沿到采样点的距离与在没有边沿发生情况下Sync_Seg到采样点的距离保持一致。“late”边沿的相位误差小于SJW，所以能被完全补偿，位（一个标称的位时间长度）结束点（从显性到隐性的边沿）发生在Sync_Seg.内。

在第二个示例中为隐性到显性的跳变边沿发生在Phase_Seg2中。该边沿称之为“early”，因为它发生在Sync_Seg之前。与“early”边沿的相对应，Phase_Seg2被缩短，Sync_Seg被忽略，以使边沿到采样点的距离与在没有边沿发生情况下Sync_Seg到采样点的距离保持一致。和上例相同，

该“early”边沿的相位误差小于SJW，所以能被完全补偿。

相位缓存段被延长或缩短只是暂时的，在下一个位时间，它们又将恢复成程序 设置的值。

在以上例子中，是从CAN硬件状态机构去观察位时序，包括位开始的位置和采样点结束的位置。当同步“early”边沿时，硬体状态机构会忽略Sync_Seg，因为它不能将在后续发生在Phase_Seg2内跳变边沿的时间片重新定义为Sync_Seg。

图6-31的示例为如何通过同步过滤短显性噪声脉冲。这些例子中毛刺都在Prop_Seg尾端开始，长度都为（Prop_Seg+ Phase_Seg1）。

在第一个例子中，同步跳转宽度大于或等于从隐性跳转到显性的毛刺边沿的相位误差。因此采样点移动到尖峰脉冲尾部后，一个隐性总线电平被采样。

在第二个例子中，SJW小于相位误差，所以采样点移动不够，显性尖峰脉冲作为当前总线电平被采样。

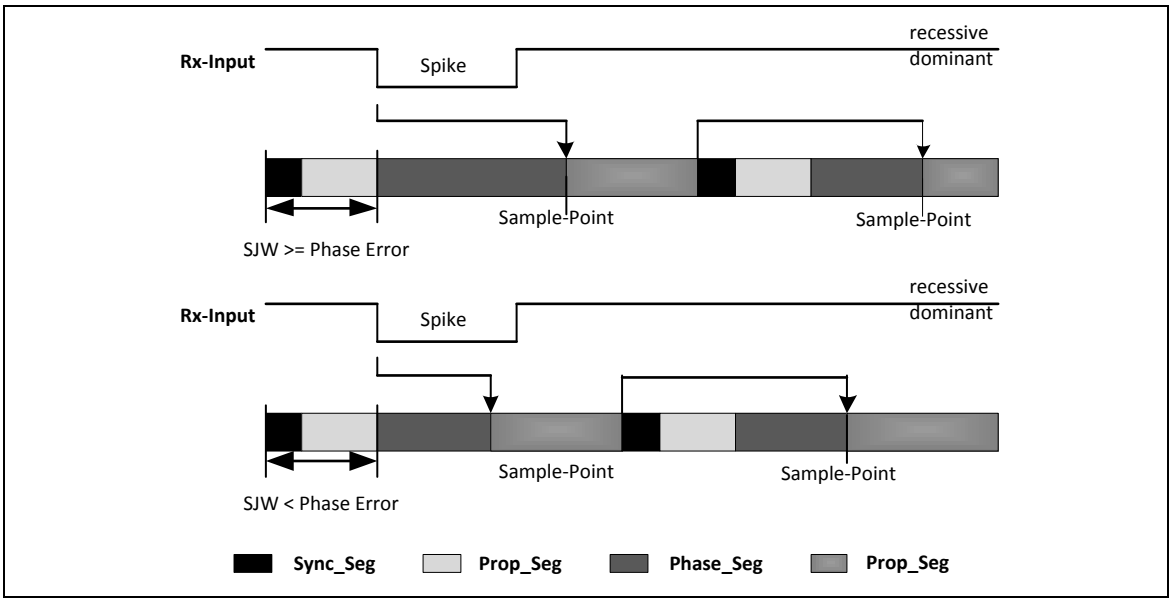


图 5-150 过滤短显性毛刺

5.20.7.19 振荡器容差范围

CAN协议从版本1.1发展到版本1.2时，扩展了振荡器偏差范围。（版本1.0未在芯片应用中实现过）选择在信号显性到隐性边沿进行同步的方式被丢弃，仅有从隐性到显性的边沿被考虑用于同步。协议更新到版本2.0（A和B），振荡器偏差范围没有更新。

振荡器频率f_{osc}基于标称频率f_{nom}的偏差范围df的公式为：

$$(1 - d_f) \cdot f_{nom} <= f_{osc} <= (1 + d_f) \cdot f_{nom}$$

该值取决于Phase_Seg1, Phase_Seg2, SJW和位时间。最大偏差值df由两个条件决定（两个都必须满足）：

$$\text{I: } d_f <= \frac{\min(\text{Phase\_Seg1}, \text{Phase\_Seg2})}{2 * (13* \text{bit\_time} - \text{Phase\_Seg2})}$$

$$\text{II: } d_f <= \frac{\text{SJW}}{20 * \text{bit\_time}}$$

注意：这些条件基于 APB时钟= f_{osc}

需要考虑的是SJW可能不会大于较小相位缓存段部分，传播时间段则会限制可被用于相位缓存段的位时间。

当Prop_Seg = 1，Phase_Seg1 = Phase_Seg2 = SJW = 4时，允许最大可能的振荡器偏差为1.58%。本组合中传播时间段仅占位时间10%不适用于短位时间，可被用于在总线40m长度上位速率达到125k Bit/s（位时间=8us）的情况。

5.20.7.20 配置CAN协议控制器

在大多数CAN编译器，也包括C_CAN，位定时配置设置在两个寄存器字节中。Prop_Seg 和 Phase_Seg1 （为TSEG1 （CAN_BTIME[11:8]））的总和和Phase_Seg2 （TSEG2（CAN_BTIME[14:12]））组合在一个字节中。SJW和BRP组合在另一个字节中。

在这些定时寄存器中，程序必须为TSEG1, TSEG2, SJW, 和 BRP这四个成员赋值，赋的值比它们的实际功能值小1。因此程序赋值范围为[0..n-1]，而不是[1..n]。举例：SJW（功能范围为[1..4]）仅用两个位就可以表示。

因此位时间的长度（程序设定值）为[TSEG1 + TSEG2 + 3] t_q或（功能值）为[Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q

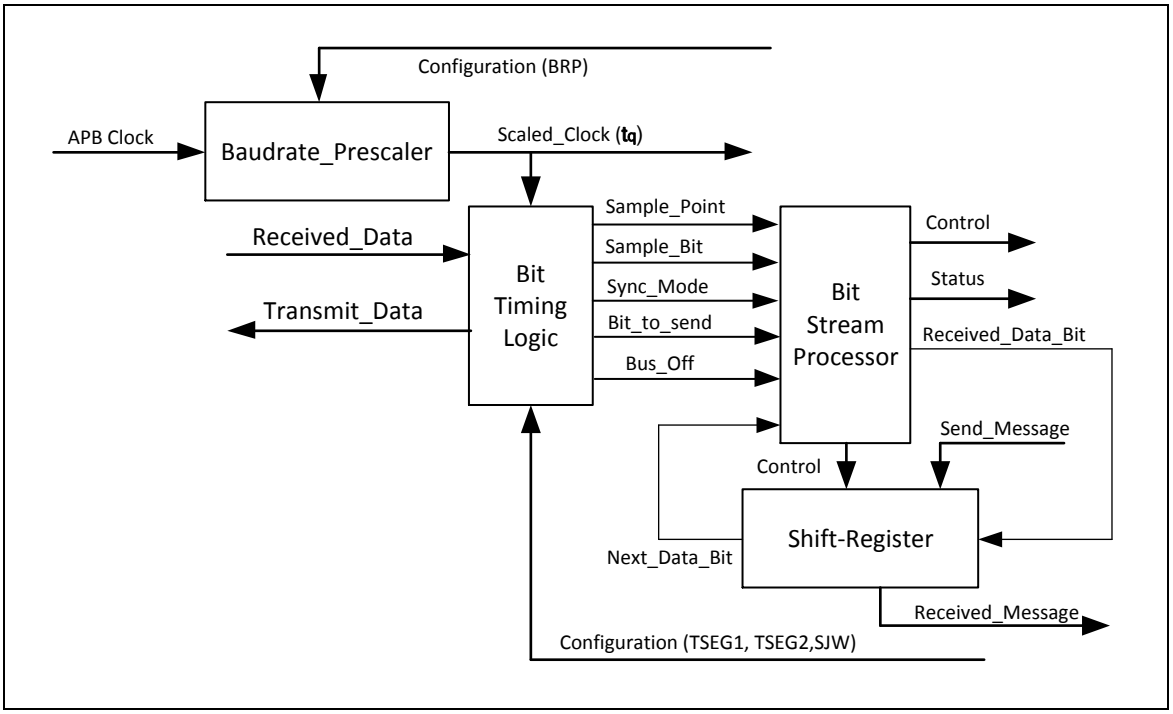


图 5-151 CAN内核的协议控制器结构

位定时寄存器中的数据是CAN协议控制器的配置输入。波特率分频器（用BRP配置）定义了时间片的长度，位时间的基本时间单元；位定时逻辑（由TSEG1, TSEG2, 和 SJW设置）定义了位时间中的时间片数目。

位定时过程，采样点位置的计算和偶发性同步都是由位定时逻辑BTL(Bit Timing Logic) 控制，BTL每时间片就要计算一次。CAN协议控制器的其他部分，位流处理器BSP(Bit Stream Processor)每个位时间在采样点计算一次。

移位寄存器以串行方式发送报文，以并行方式接收报文。该寄存器的载入和移位由BSP控制。

BSP把报文变成帧，反之亦然。它产生和丢弃封包固定格式位，插入和提取填充位，计算和检查CRC码，执行错误管理，以及决定使用哪种同步类型。它在采样点进行计算并处理采样总线输入位。计算在采样点之后发送的下一位（如数据位，CRC位，填充位，错误标记或空闲）所需的时间被称为信息处理时间（IPT）。

IPT和应用有关，但是不会长于2 t_q；对C_CAN来说IPT为0 t_q。它的长度是Phase_Seg2程序设定长度的下限。在同步的情况下，Phase_Seg2可能被缩短到一个小于IPT的值，该值不会影响总线定时。

5.20.7.21 计算位定时参数

通常，位定时配置的计算从一个期望的位速率或位时间开始。位时间（1/位 速率）的结果必须是APB时钟周期的整数倍。

位时间可能包括4到25个时间片，时间片的长度t_q由波特率分频器定义 t_q=(Baud Rate Prescaler)/f_{apb_clk}。通过下面的步骤，几种组合都可达到期望的位时间。

首先，位时间中需要定义的部分是Prop_Seg。它的长度取决于延时时间（用APB时钟测量）。对于可扩展的CAN 总线系统，最大的总线长度和最大的节点延时一样必须定义。Prop_Seg的时间也要转化成时间片（向上取最接近的t_q的整数倍的值）。

Sync_Seg是1 t_q（固定）长度，留下（位时间- – Prop_Seg – 1) t_q给两个相位缓存段。如果剩下的t_q个数是偶数，相位缓存段有相同长度，Phase_Seg2 = Phase_Seg1，如果是奇数情况，则Phase_Seg2 = Phase_Seg1 + 1.

Phase_Seg2最小的标称值也必须要考虑一下。Phase_Seg2不能比CAN控制器的IPT短，而IPT的范围是[0.2] t_q.取决于控制器的实际硬件。

同步跳转宽度的长度要设为它的最大值，即4和Phase_Seg1中的较小值。

最终配置得到的振荡器偏差差范围由章节6.6.7.19振荡器偏差范围中的方程式计算

如果有多种配置，那么应该选允许最高振荡器偏差范围的配置。

带有多个不同系统时钟的CAN节点要求不同的配置来达到相同的位速率。CAN网络中的传播时间计算是基于最长延时时间的节点，整个网络只做一次传播时间计算。

最低偏差范围的节点限制CAN系统振荡器偏差范围。

根据计算结果，可能会要求减少总线长度或位速率，或者提高振荡器频率的稳定性，目的是寻找满足协议的CAN位定时配置。写入位定时寄存器的配置结果为：

(Phase_Seg2-1) & (Phase_Seg1+Prop_Seg-1) & (SynchronisationJumpWidth-1) & (Prescaler-1)

高波特率的位定时示例：

该例中，APB_CLK的时钟为10MHz，BRP为0，位速率为1 Mbit/s。

T_q	100 ns	= t_{APB_CLK}	
delay of bus driver	50 ns		
delay of receiver circuit	30 ns		
delay of bus line (40m)	220 ns		
t_{Prop} = 6 • t_q	600 ns		
t_{SJW}	100 ns	= 1 • t_q	
t_{TSeg1} + t_{SJW}	700 ns	=t_{Prop}	

t_{TSeg2}	200	ns	= Information Processing Time + 1 • t_q
$t_{Sync-Seg}$	100	ns	= 1 • t_q
bit time	1000	ns	= $t_{Sync-Seg}$ + t_{TSeg1} + t_{TSeg2}
APB_CLK偏差	0.39	%	$= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ \ time - PB2))}$

$$= \frac{0.1us}{2 \times 13 \times (1us - 0.2us))}$$

该例中，位时间参数CAN_BTIME=0x1600

低波特率的位定时示例

该例中，APB_CLK的时钟为2MHz，BRP为1，位速率为100 Kbit/s。

t_q	1us	$= 2 \cdot t_{APB\_CLK}$
delay of bus driver	200ns	
delay of receiver circuit	80ns	
delay of bus line (40m)	220ns	
t_{Prop}	1us	$= 1 \cdot t_q$
t_{SJW}	4us	$= 4 \cdot t_q$
$t_{TSeg1} + t_{SJW}$	5us =	t_{Prop}
t_{TSeg2}	4us = Information Processing Time + $3 \cdot t_q$	
$t_{Sync-Seg}$	1us	$= 1 \cdot t_q$
bit time	10us=	$t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$

APB_CLK偏差 1.58%

$$= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ \ time - PB2))}$$

$$= \frac{4us}{2 \times 13 \times (10us - 4us))}$$

该例中，位时间参数CAN_BTIME=0x34C1

5.20.8 CAN 接口复位状态

在硬件复位后，C_CAN寄存器中值为“CAN寄存器映射寄存器描述”中的复位值。

此外，复位后为Bus-off态，CAN_TX的输出被设置为隐性（HIGH）。CAN控制寄存器的值为0x0001（Init=‘1’）使能软件初始化。直到应用软件设置Init (CAN_CON[0])位为‘0’ C_CAN才会影响CAN总线。保存在报文RAM中的数据不受硬件复位影响。在上电后，报文RAM中的内容是不确定的。

CAN寄存器表中各个Bit的功能

偏移地址	寄存器	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	CAN_CON	Reserved								Test	CCE	DAR	Res	EIE	SIE	IE	Init
04h	CAN_STATUS	Reserved								BOff	EWarn	EPass	RxOk	TxOk	LEC		
08h	CAN_ERR	RP	REC6-0							TEC7-0							
0Ch	CAN_BTIME	Res	TSeg2			TSeg1				SJW		BRP					
10h	CAN_IIDR	IntId15-8								IntId7-0							
14h	CAN_TEST	Reserved								Rx	Tx1	Tx0	LBack	Silent	Basic	Reserved	
18h	CAN_BRPE	Reserved												BRPE			
20h	CAN_IF1_CREQ	Busy	Reserved							Message Number							
24h	CAN_IF1_CMASK	Reserved								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B
28h	CAN_IF1_MASK1	Msk15-0															
2Ch	CAN_IF1_MASK2	MXtd	MDir	Res	Msk28-16												
30h	CAN_IF1_ARB1	ID15-0															
34h	CAN_IF1_ARB2	MsgVal	Xtd	Dir	ID28-16												

偏移地址	寄存器	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
38h	CAN_IF1_MCON	NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB	Reserved			DLC3-0			
3Ch	CAN_IF1_DAT_A1	Data(1)								Data(0)							
40h	CAN_IF1_DAT_A2	Data(3)								Data(2)							
44h	CAN_IF1_DAT_B1	Data(5)								Data(4)							
48h	CAN_IF1_DAT_B2	Data(7)								Data(6)							
80h	CAN_IF2_CREQ	Busy	Reserved									Message Number					
84h	CAN_IF2_CMASK	Reserved								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B
88h	CAN_IF2_MASK1	Msk15-0															
8Ch	CAN_IF2_MASK2	MXtd	MDir	Res.	Msk28-16												
90h	CAN_IF2_ARB1	ID15-0															
94h	CAN_IF2_ARB2	MsgVal	Xtd	Dir	ID28-16												
98h	CAN_IF2_MCON	NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB	Reserved			DLC3-0			
9Ch	CAN_IF2_DAT_A1	Data(1)								Data(0)							

偏移地址	寄存器	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A0h	CAN_IF2_DAT_A2	Data(3)								Data(2)							
A4h	CAN_IF2_DAT_B1	Data(5)								Data(4)							
A8h	CAN_IF2_DAT_B2	Data(7)								Data(6)							
100h	CAN_TXREQ1	TxRqst16-1															
104h	CAN_TXREQ2	TxRqst32-17															
120h	CAN_NDAT1	NewDat16-1															
124h	CAN_NDAT2	NewDat32-17															
140h	CAN_IPND1	IntPnd16-1															
144h	CAN_IPND2	IntPnd32-17															
160h	CAN_MVLD1	MsgVal16-1															
164h	CAN_MVLD2	MsgVal32-17															
168h	CAN_WU_EN	Reserved															WAKUP_EN
16Ch	CAN_WU_STATUS	Reserved															WAKUP_STS
170h	CAN_RAM_CEN	Reserved															RAM_CEN
Others	Reserved	Reserved															

表 5-27 CAN寄存器表对应各个Bit功能

注意: 读保留位都为‘0’，除了IFn Mask 2寄存器，读IFn Mask 2中保留位的值为‘1’

Res. = 保留

5.20.9 寄存器描述

C_CAN 占用256字节的地址空间。这些寄存器按照16-位寄存器安排。

两组接口寄存器(IF1 和 IF2)控制软件对报文RAM的访问。它们为往返RAM的传输数据提供缓存，避免软件访问和报文接收/发送之间发生冲突。

5.20.10寄存器表

R: 只读, **W**: 只写, **R/W**: 可读可写

寄存器	偏移地址	R/W	描述	复位值
CAN Base Address: **CAN0_BA = 0x4018_0000** **CAN1_BA = 0x4018_4000**				
CAN_CON **x=0,1**	CANx_BA+0x00	R/W	控制寄存器	0x0000_0001
CAN_STATUS **x=0,1**	CANx_BA+0x04	R/W	状态寄存器	0x0000_0000
CAN_ERR **x=0,1**	CANx_BA+0x08	R	错误计数寄存器	0x0000_0000
CAN_BTIME **x=0,1**	CANx_BA+0x0C	R/W	位定时寄存器	0x0000_2301
CAN_IIDR **x=0,1**	CANx_BA+0x10	R	中断标识符寄存器	0x0000_0000
CAN_TEST **x=0,1**	CANx_BA+0x14	R/W	测试寄存器（寄存器表备注1）	0x0000_0080
CAN_BRPE **x=0,1**	CANx_BA+0x18	R/W	BRP扩充寄存器	0x0000_0000
CAN_IF1_CREQ **x=0,1**	CANx_BA+0x20	R/W	IF1命令请求寄存器（寄存器表备注2）	0x0000_0001
CAN_IF2_CREQ **x=0,1**	CANx_BA+0x80	R/W	IF2命令请求寄存器（寄存器表备注2）	0x0000_0001
CAN_IF1_CMASK **x=0,1**	CANx_BA+0x24	R/W	IF1 命令掩码寄存器	0x0000_0000
CAN_IF2_CMASK **x=0,1**	CANx_BA+0x84	R/W	IF2 命令掩码寄存器	0x0000_0000
CAN_IF1_MASK1 **x=0,1**	CANx_BA+0x28	R/W	IF1 掩码1寄存器	0x0000_FFFF
CAN_IF2_MASK1 **x=0,1**	CANx_BA+0x88	R/W	IF2 掩码1寄存器	0x0000_FFFF
CAN_IF1_MASK2 **x=0,1**	CANx_BA+0x2C	R/W	IF1 掩码2寄存器	0x0000_FFFF

CAN_IF2_MASK2 x=0,1	CANx_BA+0x8C	R/W	IF2 掩码2寄存器	0x0000_FFFF
CAN_IF1_ARB1 x=0,1	CANx_BA+0x30	R/W	IF1 仲裁1寄存器	0x0000_0000
CAN_IF2_ARB1 x=0,1	CANx_BA+0x90	R/W	IF2 仲裁1寄存器	0x0000_0000
CAN_IF1_ARB2 x=0,1	CANx_BA+0x34	R/W	IF1 仲裁2寄存器	0x0000_0000
CAN_IF2_ARB2 x=0,1	CANx_BA+0x94	R/W	IF2 仲裁2寄存器	0x0000_0000
CAN_IF1_MCON x=0,1	CANx_BA+0x38	R/W	IF1 报文控制寄存器	0x0000_0000
CAN_IF2_MCON x=0,1	CANx_BA+0x98	R/W	IF2 报文控制寄存器	0x0000_0000
CAN_IF1_DAT_A1 x=0,1	CANx_BA+0x3C	R/W	IF1 数据A1 寄存器（寄存器表备注3）	0x0000_0000
CAN_IF1_DAT_A2 x=0,1	CANx_BA+0x40	R/W	IF1 数据A2 寄存器（寄存器表备注3）	0x0000_0000
CAN_IF1_DAT_B1 x=0,1	CANx_BA+0x44	R/W	IF1 数据B1 寄存器（寄存器表备注3）	0x0000_0000
CAN_IF1_DAT_B2 x=0,1	CANx_BA+0x48	R/W	IF1 数据B2 寄存器（寄存器表备注3）	0x0000_0000
CAN_IF2_DAT_A1 x=0,1	CANx_BA+0x9C	R/W	IF2 数据A1 寄存器（寄存器表备注3）	0x0000_0000
CAN_IF2_DAT_A2 x=0,1	CANx_BA+0xA0	R/W	IF2 数据A2 寄存器（寄存器表备注3）	0x0000_0000
CAN_IF2_DAT_B1 x=0,1	CANx_BA+0xA4	R/W	IF2 数据B1 寄存器（寄存器表备注3）	0x0000_0000
CAN_IF2_DAT_B2 x=0,1	CANx_BA+0xA8	R/W	IF2 数据B2 寄存器（寄存器表备注3）	0x0000_0000
CAN_TXREQ1 x=0,1	CANx_BA+0x100	R	发送请求寄存器1	0x0000_0000
CAN_TXREQ2 x=0,1	CANx_BA+0x104	R	发送请求寄存器2	0x0000_0000
CAN_NDAT1 x=0,1	CANx_BA+0x120	R	新数据寄存器1	0x0000_0000
CAN_NDAT2 x=0,1	CANx_BA+0x124	R	新数据寄存器2	0x0000_0000
CAN_IPND1 x=0,1	CANx_BA+0x140	R	中断挂起寄存器1	0x0000_0000

CAN_IPND2 x=0,1	CANx_BA+0x144	R	中断挂起寄存器2	0x0000_0000
CAN_MVLD1 x=0,1	CANx_BA+0x160	R	报文有效寄存器1	0x0000_0000
CAN_MVLD2 x=0,1	CANx_BA+0x164	R	报文有效寄存器2	0x0000_0000
CAN_WU_EN x=0,1	CANx_BA+0x168	R/W	唤醒使能寄存器	0x0000_0000
CAN_WU_STATUS x=0,1	CANx_BA+0x16C	R/W	唤醒状态寄存器	0x0000_0000

注意: 1. 0x00 & 0br0000000, 其中R表示CAN_RX的当前值

2. IFn: 两组报文接口寄存器– IF1 和 IF2有相同的功能

3. An/Bn: 两组数据寄存器A1, A2 和 B1, B2

4. CANx_BA,CAN寄存器基地址,x=0或x=1

CAN 控制寄存器(CAN_CON)

寄存器	偏移地址	R/W	描述	复位值
CAN_CON x=0, 1	CANx_BA+0x00	R/W	控制寄存器	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
Test	CCE	DAR	Reserved	EIE	SIE	IE	Init

位	描述	
[31:8]	**Reserved**	保留
[7]	**Test**	**测试模式使能** 0=正常操作模式 1=测试模式
[6]	**CCE**	**配置改变使能** 0=禁止写访问到 位定时寄存器 1=允许写访问到 位定时寄存器(CAN_BTIME)(当Init(CAN_CON[0]) = 1)
[5]	**DAR**	**禁止自动重传** 0=使能被干扰的报文自动重传 1=禁止自动重传
[4]	**Reserved**	保留
[3]	**EIE**	**错误中断使能** 0=禁止 -错误状态不会产生中断 1=使能 - 改变状态寄存器的BOff (CAN_STATUS[7]) 或 EWarn (CAN_STATUS[6])位将产生中断
[2]	**SIE**	**状态改变中断使能** 0=禁止 - 状态改变不会产生中断 1=使能 - 当一条报文传输成功或一个CAN总线错误被检测到时将产生中断
[1]	**IE**	**模块中断使能** 0=禁止 1=使能
[0]	**Init**	**Init初始化** 0=正常操作

		1=初始化开始

注意: 不能通过设置或复位Init(CAN_CON[0])位来缩短bus-off修复时序(参见CAN规范REV.2.0)。如果设备进入bus-off状态，它将按照自己的步调设置Init位，停止所有总线的活动。一旦Init被CPU清除，设备将在恢复正常操作之前等待129个总线空闲（129*11个连续的隐性位）。在bus-off修复时序的最后，错误管理计数器将被复位。

在复位Init之后的等待过程中，每次检测到11个连续的隐性位，一个Bit0Error码就会写入状态寄存器，使能CPU立即去检查CAN总线是否被显性或连续的干扰困在，从而监控busoff修复时序的进程。

CAN状态寄存器 (CAN_STATUS)

寄存器	偏移地址	R/W	描述	复位值
CAN_STATUS x=0, 1	CANx_BA+0x04	R/W	状态寄存器	0x0000_0000

31	30	29	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
BOFF	EWarn	EPass	RxOK	TxOK	LEC		

位	描述	
[31:8]	**Reserved**	保留
[7]	**BOff**	**Bus-Off状态（只读）** 0=CAN模块没有处于Bus-Off状态 1=CAN模块处于Bus-Off状态
[6]	**EWarn**	**错误警告状态 （只读）** 0=所有错误计数器的数值都在96次错误警告以下 1= EML中至少有一个错误计数器达到了96次错误警告
[5]	**EPass**	**被动错误（只读）** 0=Can 内核处于主动错误状态 1=CAN内核处于CAN规范定义的被动错误状态
[4]	**RxOK**	**成功接收一条报文** 0=自上次该位被CPU复位后,没有报文被成功接收到。CAN内核不能复位该位 1=自上次该位被CPU复位后, 一条报文被成功接收到.(独立于接收过滤的结果)
[3]	**TxOK**	**成功发送一条报文** 0=自上次该位被CPU复位后,没有报文被成功发送。CAN内核不能复位该位 1=自上次该位被CPU复位后, 一条报文被成功发送.
[2:0]	**LEC**	**上一次的错误码（最近一次出现在CAN总线上的错误的类型）** **LEC中的数值代表最近一次出现在CAN总线上错误的类型。当报文被成功传送，并且没有任何错误，LEC中的数值将被清零。 ‘7’为没有用到的代码， CPU 写入该数值到LEC用于检查是否有更新。表6-10详细描述错误代码的意义。**

错误码	意义
0	无错误
1	填充错误：连续超过5个相同的位出现在接收报文中，这是不允许的。
2	格式错误：收到的帧中固定格式的部分出现了错误格式。
3	应答错误：CAN内核传送的报文没有被另一个节点应答。
4	Bit1错误：在发送一条报文的过程中（除仲裁域之外），设备想发送一个隐形位（逻辑值为‘1’的位），但是监测到总线的值为显性。.
5	Bit0错误：在发送一条报文的过程中（或应答位，或主动错误标志，或超载标志），设备想发送一个显性位（逻辑值为‘0’的位），但是监测到总线的值为隐性。在busoff恢复的过程中，每检测到一个连续的11位隐性位，该状态设置一次。使能CPU监控busoff恢复序列的进程。（提示总线没有被显性或连续的干扰困住）
6	CRC错误：收到的报文CRC检查和不正确，发过来的报文中接收的CRC和计算收到数据得到的CRC不一致。
7	未使用：当LEC显示为值’7’，表示自上次CPU写该值到LEC，没有检测到CAN总线事件。

表 5-28 错误代码

状态中断

状态中断由BOff (CAN_STATUS[7]) 和EWarn(CAN_STATUS[6]) （错误中断）位或RxOK (CAN_STATUS[4])，TxOK (CAN_STATUS[3])和LEC (CAN_STATUS[2:0])（状态改变中断）位产生（假设CAN寄存器中相应的使能位被置位了）。改变EPass(CAN_STATUS[5])位或写入数据到RxOK，TxOK或LEC的不会产生状态中断。

如果该中断挂起，读状态寄存器将清除中断寄存器中的中断状态值（8000h）。

CAN错误计数寄存器(CAN_ERR)

寄存器	偏移地址	R/W	描述	复位值
CAN_ERR x=0, 1	CANx_BA+0x08	R	错误计数寄存器	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留.
[15]	RP	接收错误认可 0=接收错误计数器在被动错误标准以下 1=接收错误计数器已经达到CAN规范定义的被动错误标准
[14:8]	REC	接收错误计数器 接收错误计数器的当前状态。值到0到127之间。
[7:0]	TEC	发送错误计数器 发送错误计数器的当前状态。值到0到255之间。

位定时寄存器 (CAN_BTIME)

寄存器	偏移地址	R/W	描述	复位值
CAN_BTIME x=0, 1	CANx_BA+0x0C	R/W	位定时寄存器	0x0000_2301

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	TSeg2			TSeg1			
7	6	5	4	3	2	1	0
SJW		BRP					

位	描述	
[31:15]	**Reserved**	保留.
[14:12]	**TSeg2**	**采样点后的时间段** 0x0-0x7: TSeg2有效值为[0 … 7]。该位硬件实际解释值比程序设定值多1
[11:8]	**TSeg1**	**采样点前-Sync_Seg的时间段** 0x1-0x0f: TSeg1有效值为[1 … 15]. 该位硬件实际解释值比程序设定值多1
[7:6]	**SJW**	**(重)同步跳转宽度** 0x0-0x3:有效值为[0 … 3]. 该位硬件实际解释值比程序设定值多1
[5:0]	**BRP**	**波特率预分频器** 0x01-0x3f:该值用于振荡器频率除频产生位时间片。位时间是时间片的倍数累积。波特率分频器的有效值为[0 … 63]. 该位硬件实际解释值比程序设定值多1

注意：复位值为0x2301，对应的模块时钟APB_CLK为8 MHz，C_CAN的位速率为500kBit/s。这些寄存器只在CCE (CAN_CON[6])位和CAN (CAN_CON[0])控制寄存器中Init位被置位后才可写。

中断标识符寄存器(CAN_IIDR)

寄存器	偏移地址	R/W	描述	复位值
CAN_IIDR x=0, 1	CANx_BA+0x10	R	中断标识符寄存器	0x0000_0000

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

位	描述	
[15:0]	IntId	**中断标识符（提示中断源）** 如果几个中断同时挂起，CAN 中断寄存器将忽略中断挂起时序而指向拥有最高优先级的挂起中断。中断将保持挂起直到应用软件清除它。如果IntId不为0x0000且IE(CAN_IFn_MCON[1])置位，则到IEC的IRQ中断信号会被激活。中断保持激活状态直到IntId恢复为0x0000（引起中断的原因已复位）或直到IE复位。 状态中断拥有最高优先级。在报文中断中，报文对象的优先级随着报文编号的增加而降低。 一个报文中断通过清除该报文对象的IntPnd(CAN_IFn_MCON[13])位清除。状态中断通过状态寄存器清除。

IntId值	意义
0x0000	没有中断挂起。
0x0001-0x0020	造成中断的报文对象编号
0x0021-0x7FFF	未使用
0x8000	状态中断
0x8001-0xFFFF	未使用

表 5-29 中断源

测试寄存器 (CAN_TEST)

寄存器	偏移地址	R/W	描述	复位值
CAN_TEST x=0, 1	CANx_BA+0x14	R/W	测试寄存器（寄存器表备注1）	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
Rx	Tx[1:0]		LBack	Silent	Basic	Res	

位	描述	
[31:8]	**Reserved**	保留.
[7]	**Rx**	**监测CAN_RX管脚的当前实际值（只读）** 0 = CAN总线为显性(CAN_RX = ‘1’). 1 = CAN总线为隐性(CAN_RX = ‘0’).
[6:5]	**Tx[1:0]**	**Tx[1:0]: 控制CAN_TX 管脚** 00 = 复位值，CAN_TX由CAN内核控制 01 = 采样点监测CAN_TX管脚 10 = CAN_TX管脚驱动一个显性值(‘0’) 11 = CAN_TX管脚驱动一个 隐性值(‘1’)
[4]	**LBack**	**Loop Back模式** 0=Loop Back模式禁止 1=Loop Back模式使能
[3]	**Silent**	**Silent 模式** 0 = 正常操作 1 = 模块工作在Silent模式
[2]	**Basic**	**Basic模式** 0 = Basic模式禁止 1 = IF1 寄存器用作TX Buffer，IF2寄存器用作RX Buffer.
[1:0]	**Res**	**保留：** 该位为保留位。 该位读时总是为0，也必须写‘0’到该位

复位值：0000 0000 R000 0000 b (R:RX管脚的当前值)

注意：通过设置CAN控制寄存器的Test位来使能测试寄存器的写访问。不同的测试功能可以组合使用，但

是Tx[1-0] (CAN_TEST[6:5]) “00”会干扰报文传输。

波特率分频扩展寄存器 (CAN_BRPE)

寄存器	偏移地址	R/W	描述	复位值
CAN_BRPE x=0, 1	CANx_BA+0x18	R/W	波特率分频扩展寄存器	0x0000_0000

31	30	29	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				**BRPE**			

位	描述	
[31:4]	**Reserved**	保留.
[3:0]	**BRPE**	**波特率分频扩展** 0x00-0x0F: 通过编程BRPE，波特率分频可以被扩展到1023。该位硬件实际解释值比程序计算BRPE (MSBs) 和 BTIME (LSBs)的值要多1

报文接口寄存器组

模块共有2组接口寄存器，用于控制CPU访问报文RAM。接口寄存器避免CPU访问报文RAM时与CAN报文发送和接收（通过缓存传输）发生冲突。一个完整的报文对象或部分报文对象在报文RAM和IFn报文缓存寄存器间的传输只需单次传输就可以完成

除Basic测试模式外，2组接口寄存器的功能是相同的。它们可以这样用：一组寄存器用于传送数据到报文RAM，另一组用于接收来自报文RAM的数据，它们各自处理中断。表6-12（IF1和IF2 报文接口寄存器组）提供了2组接口寄存器的概述。

每组接口寄存器由报文缓存寄存器构成，分别由各自的命令寄存器控制。命令掩码寄存器设定数据传输的方向和报文对象的哪个部分将要被传输。命令请求寄存器用于选择报文RAM中的一个报文对象作为传输的目标或源，并开始执行命令掩码寄存器中指定的命令。

地址	IF1 寄存器组	地址	IF2 寄存器组
CANx_BA+0x20; x=0,1	IF1 命令请求	CANx_BA+0x80; x=0,1	IF2 命令请求
CANx_BA+0x24; x=0,1	IF1 命令掩码	CANx_BA+0x84; x=0,1	IF2 命令掩码
CANx_BA+0x28; x=0,1	IF1 掩码 1	CANx_BA+0x88; x=0,1	IF2 掩码1
CANx_BA+0x2C; x=0,1	IF1 掩码 2	CANx_BA+0x8C; x=0,1	IF2 掩码 2
CANx_BA+0x30; x=0,1	IF1 仲裁 1	CANx_BA+0x90; x=0,1	IF2 仲裁1
CANx_BA+0x34; x=0,1	IF1 仲裁2	CANx_BA+0x94; x=0,1	IF2 仲裁2
CANx_BA+0x38; x=0,1	IF1报文控制	CANx_BA+0x98; x=0,1	IF2 报文控制
CANx_BA+0x3C; x=0,1	IF1 数据 A 1	CANx_BA+0x9C; x=0,1	IF2 数据 A 1
CANx_BA+0x40; x=0,1	IF1 数据 A 2	CANx_BA+0xA0; x=0,1	IF2 数据 A 2
CANx_BA+0x44; x=0,1	IF1 数据 B 1	CANx_BA+0xA4; x=0,1	IF2 数据 B 1
CANx_BA+0x48; x=0,1	IF1 数据 B 2	CANx_BA+0xA8; x=0,1	IF2 数据 B 2

表 5-30 IF1 和 IF2 报文接口寄存器

IFn 命令请求寄存器 (CAN_IFn_CREQ)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_CREQ x=0,1	CANx_BA+0x20	R/W	IF1 (寄存器表备注2) 命令请求寄存器	0x0000_0001
CAN_IF2_CREQ x=0,1	CANx_BA+0x80	R/W	IF2 (寄存器表备注2) 命令请求寄存器	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
Busy	Res						
7	6	5	4	3	2	1	0
Res		Message Number					

位	描述	
[31:16]	Reserved	保留.
[15]	Busy	忙标志 0 = 读 / 写动作已经完成 1 = 写到IFn命令请求寄存器的动作正在进行。该位只能由软件读取。
[14:6]	Reserved	保留.
[5:0]	Message Number	报文编号 0x01-0x20：有效的报文号，报文RAM中被选择用于数据传输的报文对象 0x00：非有效的报文号，视同0x20 0x21-0x3f：非有效的报文号，视同0x01-0x1F

一旦应用软件写报文编号到命令请求寄存器，报文传输就开始了。伴随着这个写操作，Busy位将自动被置位来通知CPU传输正在进行中。在等待3到6个APB_CLK 周期后，接口寄存器和报文RAM之间的传输完成了。Busy位被清除。

注意：当写入到命令请求寄存器中的报文编号为非有效值时，报文号将被解释转换为一个有效的值，对应报文对象也将会被传输。

IFn 命令掩码寄存器 (CAN_IFn_CMASK)

IFn命令掩码寄存器指定传输方向和选择哪个IFn报文缓存寄存器作为数据传输的源或目标。

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_CMASK x=0,1	CANx_BA+0x24	R/W	IF1命令掩码寄存器	0x0000_0000
CAN_IF2_CMASK x=0,1	CANx_BA+0x84	R/W	IF2命令掩码寄存器	0x0000_0000

31	30	29	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
WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/ NewDat	DAT_A	DAT_B

位	描述	
[31:8]	**Reserved**	保留.
[7]	**WR/RD**	**写/读 模式** 0 = 读：将命令请求寄存器中指定地址的报文对象传输到选中的报文缓存寄存器中 1 = 写：将选中的报文缓存寄存器的数据传送到命令请求寄存器中指定地址的报文对象中
[6]	**Mask**	**访问掩码位** 写操作 0=掩码位不变 1=传输标识符 Mask + MDir + MXtd 到报文对象 读操作： 0=掩码位不变 1=传输标识符 Mask + MDir + MXtd 到IFn报文缓存寄存器
[5]	**Arb**	**访问仲裁位** 写操作： 0=仲裁位不变 1= 传输 Identifier + Dir (CAN_IFn_ARB2[13]) + Xtd (CAN_IFn_ARB2[14]) + MsgVal (CAN_IFn_ARB2[15])到报文对象 读操作： 0=仲裁位不变 1= 传输 Identifier + Dir (CAN_IFn_ARB2[13]) + Xtd (CAN_IFn_ARB2[14]) + MsgVal (CAN_IFn_ARB2[15])到IFn报文缓存寄存器

[4]	**Control**	**访问控制位** 写操作： 0=控制位不变 1=传输控制位到报文对象 读操作： 0=控制位不变 1=传输控制位到IFn报文缓存寄存器
[3]	**ClrIntPnd**	**清除中断挂起位：** 写操作： 当写报文对象时，该位忽略。 读操作： 1=清除报文对象中的IntPnd位 0=IntPnd(CAN_IFn_MCON[13])位保持不变
[2]	**TxRqst/NewDat**	**写操作时访问传送请求位** 0=TxRqst位不变 1=设置TxRqst位 **注意：**如果传送是通过设置IFn命令掩码寄存器中的TxRqst/NewDat位请求进行的，则 IFn报文控制寄存器中的TxRqst位将被忽略。 读操作时访问New Data位 0=NewDat位保持不变 1=清除报文对象中的NewDat位 **注意：**可通过复位控制位IntPnd和NewDat实现对一个报文对象的读取。传送到IFn报文控制寄存器的这些值总是反映其状态（在复位这些位之前）。
[1]	**DAT_A**	**访问数据字节[3：0]** 写： 0=数据[3:0]未变 1=写传输数据字节[3:0]到报文对象 读： 0= 数据字节[3:0]未变 1=传送数据字节[3:0]到IFn报文缓存寄存器
[0]	**DAT_B**	**访问数据字节[7：4]** 写： 0=数据[7:4]未变 1=写传输数据字节[7:4]到报文对象 读： 0= 数据字节[7:4]未变 1=传送数据字节[7:4]到IFn报文缓存寄存器

IFn 掩码1 寄存器 (CAN_IFn_MASK1)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_MASK1 x=0,1	CANx_BA+0x28	R/W	IF1 掩码1 寄存器	0x0000_FFFF
CAN_IF2_MASK1 x=0,1	CANx_BA+0x88	R/W	IF2 掩码1 寄存器	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**
Msk[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Msk[7:0]							

位	描述	
[31:16]	**Reserved**	保留 .
[15:0]	**Msk[15:0]**	**标识符掩码15-0** 0=对应的报文对象标识符位不用于接收过滤 1=对应的报文对象标识符位用于接收过滤

IFn 掩码2 寄存器 (CAN_IFn_MASK2)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_MASK2 x=0,1	CANx_BA+0x2C	R/W	IF1 掩码2 寄存器	0x0000_FFFF
CAN_IF2_MASK2 x=0,1	CANx_BA+0x8C	R/W	IF2 掩码2 寄存器	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**
MXtd	**MDir**	**Reserved**	**Msk[28:24]**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Msk[23:16]							

位	描述	
[31:16]	**Reserved**	保留.
[15]	**MXtd**	**掩码扩展标识符** 0=扩展标识符位（IDE）对于接收过滤没有影响 1=扩展标识符位（IDE）用于接收过滤 **注意**：报文对象采用11-位（标准）标识符时， 接收到的数据帧的标识符将会写入到ID28 - ID18 (CAN_IFn_ARB2[12:2])。所有这些位以及掩码位Msk28 - Msk18 (CAN_IFn_MASK2[12:2]) ，都要列入到接收过滤的考虑中。
[14]	**MDir**	**掩码报文方向** 1=报文方向(Dir (CAN_IFn_ARB2[13]))位用于接收过滤 0=报文方向位对于接收过滤没有影响
[13]	**Reserved**	保留.
[12:0]	**Msk[28:16]**	**标识符掩码28-16** 0=报文对象标识符的相应位不用于接收过滤的匹配操作 1=相应的标识符用于接收过滤

IFn仲裁1寄存器 (CAN_IFn_ARB1)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_ARB1 x=0,1	CANx_BA+0x30	R/W	IF1仲裁1寄存器	0x0000_0000
CAN_IF2_ARB1 x=0,1	CANx_BA+0x90	R/W	IF2仲裁1寄存器	0x0000_0000

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

位	描述	
[31:16]	**Reserved**	保留.
[15:0]	**ID[15:0]**	**报文标识符15-0** ID28 - ID0, 29-位标识符（“扩展帧”） ID28 - ID18, 11-位标识符（“标准帧”）

IFn仲裁2 寄存器 (CAN_IFn_ARB2)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_ARB2 x=0,1	CANx_BA+0x34	R/W	IF1仲裁2寄存器	0x0000_0000
CAN_IF2_ARB2 x=0,1	CANx_BA+0x94	R/W	IF2仲裁2寄存器	0x0000_0000

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

位	描述	
[31:16]	**Reserved**	保留.
[15]	**MsgVal**	**报文有效** 0=报文对象无效，报文处理器将忽略此报文对象 1=报文对象有效，报文对象已配置，而且报文处理器将处理此报文对象 **注意**：在初始化过程中，复位CAN控制寄存器的Init位之前，应用软件必须复位所有未使用的报文对象的MsgVal位。如果报文对象需要修改ID28-0，控制位Xtd，Dir，或数据长度码DLC23-0，那么在ID28-0，控制位Xtd，Dir，或数据长度码DLC23-0修改之前需将此位复位。
[14]	**Xtd**	**扩展标识符：** 0 = 11-位（标准）标识符用于报文对象 1 = 29-位（扩展）标识符用于报文对象
[13]	**Dir**	**报文方向：** 1=方向为发送 TxRqst 端，各自的报文对象作为数据帧被传输。在收到带有匹配标识符的远程帧时，该报文对象的TxRqst位被置位。（如果RmtEn=1） 0=方向为接收 TxRqst端，带有该报文对象标识符的远程帧被传输。在收到带有匹配标识符的数据帧时，报文保存在该报文对象内
[12:0]	**ID[28:16]**	**报文标识符28-16** ID28 - ID0, 29位标识符（“扩展帧”） ID28 - ID18, 11-位标识符（“标准帧”）

IFn报文控制寄存器 (CAN_IFn_MCON)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_MCON x=0,1	CANx_BA+0x38	R/W	IF1 报文控制寄存器	0x0000_0000
CAN_IF2_MCON x=0,1	CANx_BA+0x98	R/W	IF2 报文控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
7	6	5	4	3	2	1	0
EoB	Reserved			DLC[3:0]			

位	描述	
[31:16]	**Reserved**	保留.
[15]	**NewDat**	**新数据** 0=自上次该位被应用软件清除后，该报文对象的数据部分还没有由报文处理器写入新数据。 1=报文处理器或应用软件已经写了新数据到该报文对象的数据部分。
[14]	**MsgLst**	**报文丢失（只对报文对象方向 =接收 有效）** 0=自上次该位被CPU复位后无报文丢失 1=当报文处理器保存新的一个报文到报文对象，NewDat虽然被置位，但CPU已经丢失了一条报文。
[13]	**IntPnd**	**中断挂起** 0=该报文对象不是中断源 1=该报文对象是中断源。如果没有更高优先级的中断源存在时，中断寄存器中的中断标识符将指向该报文对象。
[12]	**UMask**	**使用验收掩码** 0=忽略掩码 1=使用掩码(Msk28-0, MXtd, and MDir)用于接收过滤 **注意：**如果UMask位被置1，则在报文对象初始化的过程中，MsgVal(CAN_IFn_ARB2[15])位被置为1之前，报文对象的掩码位必须先被设定。
[11]	**TxIE**	**发送中断使能** **0=** IntPnd (CAN_IFn_MCON[13])在成功传送一帧后将保持不变 1= IntPnd (CAN_IFn_MCON[13])在成功传送一帧后将被置位
[10]	**RxIE**	**接收中断使能** **0=** IntPnd (CAN_IFn_MCON[13])在成功接收一帧后将保持不变

		1= IntPnd (CAN_IFn_MCON[13])在成功接收一帧后将被置位
[9]	**RmtEn**	**远程使能** 0=在收到一远程帧后，TxRqst保持不变 1=在收到一远程帧后，TxRqst被置位
[8]	**TxRqst**	**传送请求** 0=该报文对象不在等待传送 1=该报文对象已请求传送但还没有完成
[7]	**EoB**	**缓存结束** 0=报文对象属于某FIFO缓存，但不是该FIFO缓存的最后一个报文对象。 1=单报文对象或FIFO缓存的最后一个报文对象。 **注意：**该位用于关联两个或多个 报文对象（最多32 个）组成一个FIFO 缓存。 对于单个报文对象（不属于任何FIFO 缓存），该位必须始终设置为 1。
[6:4]	**Reserved**	保留.
[3:0]	**DLC**	**数据长度码** 0-8：数据帧有0-8个数据字节 9-15：数据帧有8个数据字节 **注意：**报文对象的数据长度码必须和其他节点中所有带有相同标识符的报文对象的数据长度码相同。当报文处理器保存一个数据帧时，它将把收到报文提供的数值写入DLC。 Data 0：CAN数据帧的1st数据字节 Data 1：CAN数据帧的2nd数据字节 Data 2：CAN数据帧的3rd数据字节 Data 3：CAN数据帧的4th数据字节 Data 4：CAN数据帧的5th数据字节 Data 5：CAN数据帧的6th数据字节 Data 5：CAN数据帧的7th数据字节 Data 7：CAN数据帧的8th数据字节 **注意：**Data 0字节是接收过程中，移入CAN内核的移位寄存器的第一个数据字节，Data 7字节是最后一个。当报文处理器保存一个数据帧，他将会写所有8个数据字节到报文对象中。如果数据长度码小于8，则报文对象剩下的字节将被非指定值覆盖。

IFn 数据A1 寄存器 (CAN_IFn_DAT_A1)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_DAT_A1 x=0,1	CANx_BA+0x3C	R/W	IF1 数据A1 寄存器(寄存器表备注 3)	0x0000_0000
CAN_IF2_DAT_A1 x=0,1	CANx_BA+0x9C	R/W	IF2 数据A1 寄存器(寄存器表备注 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**
Data[1]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data[0]							

位	描述	
[31:16]	**Reserved**	保留
[15:8]	**Data [1]**	**数据字节1** CAN 数据帧的第2个数据字节
[7:0]	**Data [0]**	**数据字节0** CAN 数据帧的第1个数据字节

IFn 数据A2 寄存器 (CAN_IFn_DAT_A2)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_DAT_A2 x=0,1	CANx_BA+0x40	R/W	IF1 数据A2 寄存器(寄存器表备注 3)	0x0000_0000
CAN_IF2_DAT_A2 x=0,1	CANx_BA+0xA0	R/W	IF2 数据A2 寄存器(寄存器表备注 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**
Data[3]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data[2]							

位	描述	
[31:16]	**Reserved**	保留
[15:8]	**Data [3]**	**数据字节3** CAN 数据帧的第4个数据字节
[7:0]	**Data [2]**	**数据字节2** CAN 数据帧的第3个数据字节

IFn 数据B1 寄存器 (CAN_IFn_DAT_B1)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_DAT_B1 x=0,1	CANx_BA+0x44	R/W	IF1 数据B1 寄存器(寄存器表备注 3)	0x0000_0000
CAN_IF2_DAT_B1 x=0,1	CANx_BA+0xA4	R/W	IF2 数据B1 寄存器(寄存器表备注 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**
Data(5)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(4)							

位	描述	
[31:16]	**Reserved**	保留.
[15:8]	**Data [5]**	**数据字节5** CAN 数据帧的第6个数据字节
[7:0]	**Data [4]**	**数据字节4** CAN 数据帧的第5个数据字节

IFn 数据B2 寄存器 (CAN_IFn_DAT_B2)

寄存器	偏移地址	R/W	描述	复位值
CAN_IF1_DAT_B2 x=0,1	CANx_BA+0x48	R/W	IF1 数据B2 寄存器(寄存器表备注 3)	0x0000_0000
CAN_IF2_DAT_B2 x=0,1	CANx_BA+0xA8	R/W	IF2 数据B2 寄存器(寄存器表备注 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**
Data[7]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data[6]							

位	描述	
[31:16]	**Reserved**	保留.
[15:8]	**Data [7]**	**数据字节7** CAN 数据帧的第8个数据字节
[7:0]	**Data [6]**	**数据字节6** CAN 数据帧的第7个数据字节

在一个CAN数据帧中，数据[0]是第一个，数据[7]是传送或接收的最后一个字节数据。在CAN的串行位流，每个字节的MSB将被先传送。

报文内存中的报文对象

在报文RAM中有32个报文对象。为了避免应用软件在访问报文RAM时与CAN报文接收发送发生冲突，CPU不能直接访问报文对象，所有访问读取都要通过IFn接口寄存器。表6-13详细列出了一个报文对象的结构。

报文对象												
UMask	Msk [28:0]		MXtd	MDir	EoB	NewDat		MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID [28:0]		Xtd	Dir	DLC [3:0]	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7

表 5-31 报文内存中报文对象的结构

仲裁寄存器ID28-0(CAN_IFn_ARB1/2)，Xtd (CAN_IFn_ARB2[14])和Dir(CAN_IFn_ARB2[13])用于定义标识符，传出报文的传送类型以及传入报文的接收过滤（接收过滤还要用到屏蔽寄存器Msk28-0 (CAN_IFn_MASK1/2)，MXtd (CAN_IFn_MASK2[15])，和 MDir (CAN_IFn_MASK2[14]) ）。接收到的报文存储在对应标识符匹配，方向=接收（数据帧）或方向=发送（远程帧）的有效报文对象中。扩展帧只能保存在Xtd=1的报文对象中，标准帧保存在Xtd=0的报文对象中。如果一条接收到的报文（数据帧或远程帧）和1个以上的有效报文对象匹配，那么它将被保存在报文编号最低的报文对象中。

报文处理寄存器

所有的报文处理寄存器都是只读。报文处理寄存器内容（每个报文对象的TxRqst(CAN_IFn_MCON[8]), NewDat (CAN_IFn_MCON[15]), IntPnd(CAN_IFn_MCON[13]), 和 MsgVal (CAN_IFn_ARB2[15])，以及中断标示符）为报文处理器FSM提供的状态信息。

传送请求寄存器 1 (CAN_TXREQ1)

这些寄存器保存32个报文对象的TxRqst位。通过读取TxRqst位，软件可以检查哪个报文对象的传送请求正在挂起。在收到一帧或者成功传送一帧后，应用软件可以通过IFn报文接口寄存器或者报文处理器，可以置位/复位指定报文对象的TxRqst位

寄存器	偏移地址	R/W	描述	复位值
CAN_TXREQ1 x=0, 1	CANx_BA+0x100	R	传送请求寄存器 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**
TxRqst 16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TxRqst 8-1							

位	描述	
[31:16]	**Reserved**	保留.
[15:0]	**TxRqst 16-1**	**传送请求位16-1（所有报文对象）** 0=该报文对象没有等待传送 1=该报文对象已有传送请求但还未完成。 该位只读

传送请求寄存器 2 (CAN_TXREQ2)

寄存器	偏移地址	R/W	描述	复位值
CAN_TXREQ2 x=0, 1	CANx_BA+0x104	R	传送请求寄存器 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TxRqst32-25							
7	6	5	4	3	2	1	0
TxRqst24-17							

位	描述	
[31:16]	**Reserved**	保留
[15:0]	**TxRqst 32-17**	**传送请求位32-17（所有报文对象）** 0=该报文对象没有等待传送 1=该报文对象已有传送请求但还未完成。 该位只读

新数据寄存器 1 (CAN_NDAT1)

这些寄存器保存32个报文对象的NewDat位。通过读取NewDat位，软件可以检查哪个报文对象的数据部分被更新了。在收到一帧或者成功传送一帧后，应用软件可以通过IFn报文接口寄存器或者报文处理器，置位/复位指定报文对象的NewDat位。

寄存器	偏移地址	R/W	描述	复位值
CAN_NDAT1 **x=0, 1**	CANx_BA+0x120	R	新数据寄存器 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**
NewData16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NewData 8-1							

位	描述	
[31:16]	**Reserved**	保留.
[15:0]	**NewData16-1**	**新数据位 16-1（所有 报文对象）** 0=自上次该标志被软件清除后，报文处理器没有写入新数据到该报文对象的数据部分 1=报文处理器或应用软件已经写入了新数据到该报文对象的数据部分

新数据寄存器 2 (CAN_NDAT2)

寄存器	偏移地址	R/W	描述	复位值
CAN_NDAT2 **x=0, 1**	CANx_BA+0x124	R	新数据寄存器2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NewData 32-25							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NewData 24-17							

位	描述	
[31:16]	**Reserved**	备注
[15:0]	**NewData 32-17**	**新数据位 32-17（所有 报文对象）** 0=自上次该标志被软件清除后，报文处理器没有写入新数据到该报文对象的数据部分 1=报文处理器或应用软件已经写入了新数据到该报文对象的数据部分

中断挂起寄存器 1 (CAN_IPND1)

这些寄存器保存32个报文对象的IntPnd位。通过读取IntPnd位，软件可以检查哪个报文对象的中断挂起。在收到一帧或者成功传送一帧后，应用软件可以通过IFn报文接口寄存器或者报文处理器，置位/复位指定报文对象的IntPnd位。这也将会影响中断寄存器的Intld的值

寄存器	偏移地址	R/W	描述	复位值
CAN_IPND1 **x=0, 1**	CANx_BA+0x140	R	中断挂起寄存器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**
IntPnd16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntPnd 8-1							

位	描述	
[31:16]	**Reserved**	保留.
[15:0]	**IntPnd16-1**	**中断挂起位16-1(所有报文对象)** 0=表示该报文对象不是中断源 1=表示该报文对象是中断源

中断挂起寄存器 2 (CAN_IPND2)

寄存器	偏移地址	R/W	描述	复位值
CAN_IPND2 **x=0, 1**	CANx_BA+0x144	R	中断挂起寄存器2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IntPnd 32-25							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntPnd 24-17							

位	描述	
[31:16]	**Reserved**	保留.
[15:0]	**IntPnd 32-17**	**中断挂起位32-17(所有报文对象)** 0=表示该报文对象不是中断源 1=表示该报文对象是中断源

报文有效寄存器 1 (CAN_MVLD1)

这些寄存器保存32个报文对象的MsgVal位。通过读取MsgVal位，应用软件可以检查哪个报文对象是有效的。应用软件（通过IFn报文接口寄存器）可以来置位/复位指定报文对象的MsgVal位。

寄存器	偏移地址	R/W	描述	复位值
CAN_MVLD1 **x=0, 1**	CANx_BA+0x160	R	报文有效寄存器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**
MsgVal 16- 9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MsgVal 8-1							

位	描述	
[31:16]	**Reserved**	保留 .
[15:0]	**MsgVal 16-1**	**报文有效位16-1 （所有报文对象）（只读）** 0= 该报文对象无效，被报文处理器忽略 1= 该报文对象有效，可以被报文处理器考虑 例如：.CAN_MVLD1[0]表示No.1对象是否有效，如果CAN_MVLD1[0]被置位，则报文对象No.1有效。

报文有效寄存器 2 (CAN_MVLD2)

寄存器	偏移地址	R/W	描述	复位值
CAN_MVLD2 x=0, 1	CANx_BA+0x164	R	报文有效寄存器2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
MsgVal 32-25							
7	6	5	4	3	2	1	0
MsgVal 24-17							

位	描述	
[31:16]	**Reserved**	保留 .
[15:0]	**MsgVal 32-17**	**报文有效位32-17（所有报文对象）（只读）** 0= 该报文对象无效，被报文处理器忽略 1= 该报文对象有效，可以被报文处理器考虑 例如：.CAN_MVLD2[15]表示No.32对象是否有效，如果CAN_MVLD2[15]被置位，则报文对象No.32为有效

唤醒使能寄存器 (CAN_WU_EN)

寄存器	偏移地址	R/W	描述	复位值
CAN_WU_EN x=0, 1	CANx_BA+0x168	R/W	唤醒使能寄存器	0x0000_0000

31	30	29	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							WAKUP_EN

位	描述	
[31:1]	**Reserved**	保留.
[0]	**WAKUP_EN**	**唤醒使能位** 0 = 禁止唤醒功能 1 = 使能唤醒功能 **注意**: 当CAN_Rx管脚上有下降沿信号时,用户可唤醒系统

唤醒状态寄存器 (CAN_WU_STATUS)

寄存器	偏移地址	R/W	描述	复位值
CAN_WU_STATUS x=0, 1	CANx_BA+0x16C	R/W	唤醒状态寄存器	0x0000_0000

31	30	29	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							WAKUP_STS

位	描述	
[31:1]	**Reserved**	保留.
[0]	**WAKUP_STS**	**唤醒状态** 0= 没有唤醒事件发生 1= 有唤醒事件发生了 **注意**：该位可写'0'清除

5.21 模拟数字转换（ADC）

5.21.1 概述

NuMicro™ NUC200系列包含一个12-位8通道逐次逼近式的模拟-数字转换器（SAR A/D converter）。A/D转换器支持三种操作模式：单一(single)，单周期扫描（single-cycle scan）和连续扫描模式(continuous scan mode)。A/D转换器可由软件、PWM 中心对齐触发器和外部STADC管脚启动转换。

5.21.2 特性

- 模拟输入电压范围：0~VREF
- 12-bit分辨率和10-bit精确度保证
- 多达8路单端模拟输入通道或4路差分模拟输入通道
- 高达1M SPS的转换速率（芯片工作电压为5V）
- 三种操作模式：
  - 单一模式：对指定的一个通道只进行一次A/D转换。
  - 单周期扫描模式：对所有指定通道完成一次A/D转换，转换顺序从最小号通道到最大号通道。
  - 连续扫描模式：A/D转换器持续执行单周期扫描直到软件停止A/D转换。
- A/D转换开始条件：
  - 软件向ADST(ADCR[11])位写1
  - PWM 中央对齐触发
  - 外部STADC管脚
- 每个通道的转换结果都存储在对应的数据寄存器中，并且带有valid/overrun提示标志。
- 支持2路数字比较器。 转换结果可与指定的值进行比较。当转换值和指定比较寄存器中的设定值相同时，用户还可以选择是否产生一个中断请求。
- 通道7支持3 路输入源：外部模拟电压，内部带隙电压和内部温度传感器输出。

5.21.3 模块图

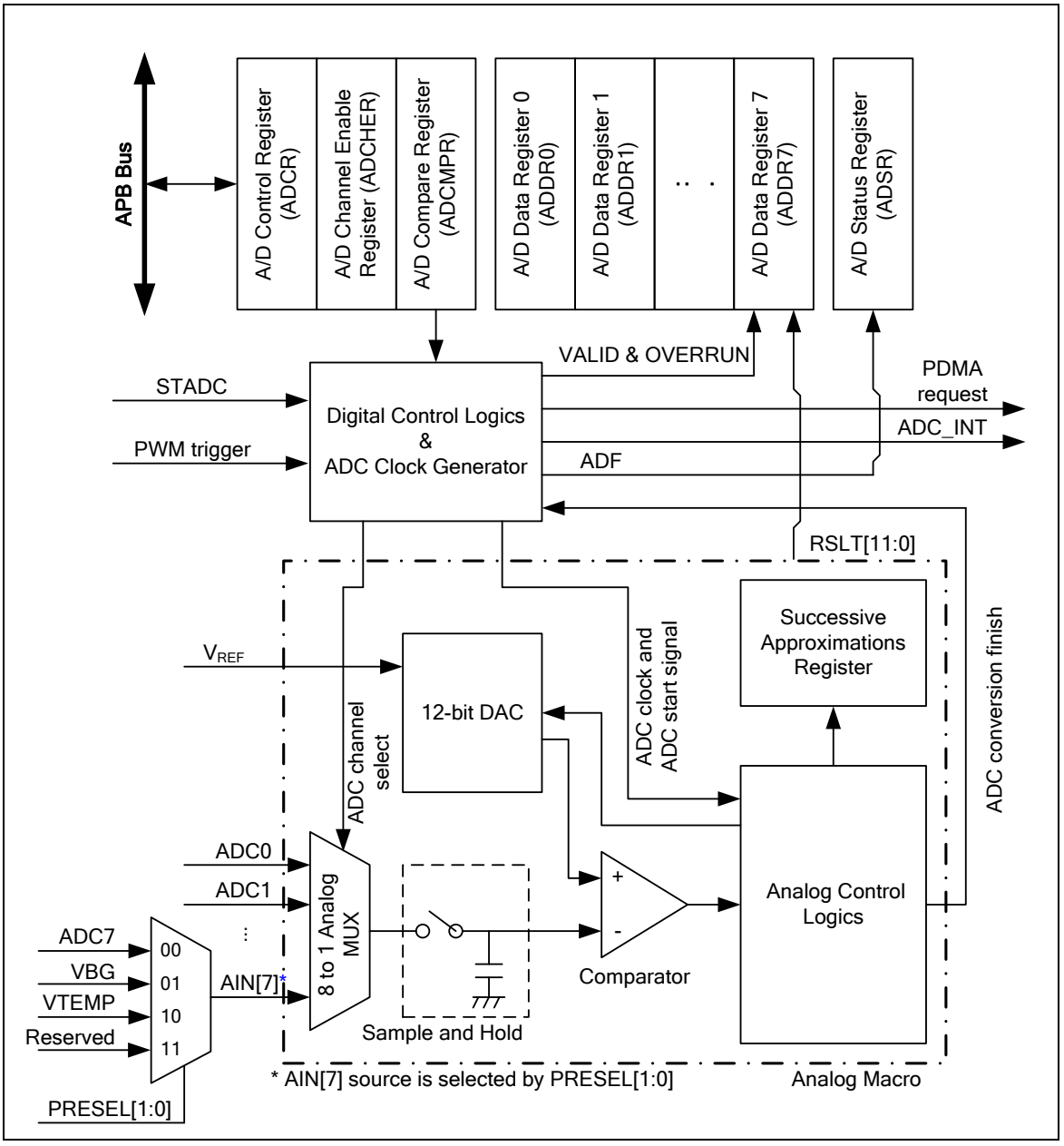


图 5-152 ADC 控制器模块图

5.21.4 基本配置

ADC 模块控制器的时钟源是由ADC_EN(CLK_APBCLK[28])控制使能的。用户将GPA_MFP寄存器配置为ADC模拟输入管脚后，还需要设置OFFD (GPIOA_OFFD [23:16]) = 1，将管脚数字输入通道关闭。

5.21.5 功能描述

A/D转换器采用逐次逼近转换方式，转换结果为12-位数据。 ADC有三种工作模式：单一模式，单周期扫描模式和连续扫描模式。当需要改变转换模式或模拟输入通道时，为防止错误操作，软件必须先清ADST（ADCR[11]）为0 。

5.21.5.1 *ADC时钟发生器*

最大采样率可达1 MSPS。ADC有4个时钟源，通过ADC_S(CLKSEL1[3:2]) 进行选择，ADC时钟频率按如下公式进行8位预分频：

ADC时钟频率 = (ADC时钟源频率) / (ADC_N (CLKDIV[23:16])+1);

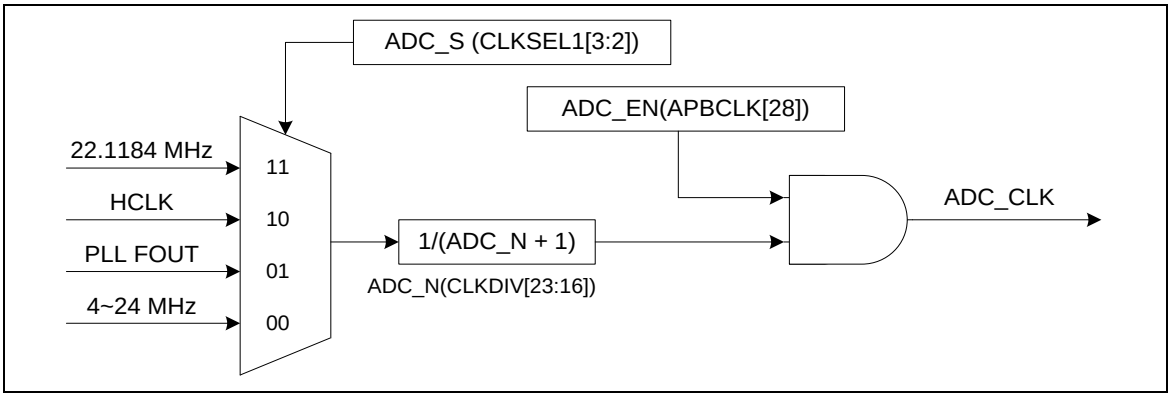


图 5-153 ADC 时钟控制

5.21.5.2 *单一模式*

在单一模式下，A/D转换器仅对指定的一个通道进行一次转换。操作流程为：

1. 当ADST (ADCR[11]) 位被软件置位时，A/D转换开始。
2. 当A/D转换完成，转换结果将存储到与通道对应的A/D数据寄存器中。
3. A/D转换完成后， ADF (ADSR[0]) 位会被置1，如果此时ADIE(ADCR[1])=1,则产生ADC中断。
4. 在A/D转换期间，ADST位一直保持为1。当A/D转换结束，ADST位会自动清0， A/D转换器进入IDLE状态。

注意：如果在单次模式下，软件使能了多个通道，则编号最小的通道将会被选中进行转换，其他通道将被忽略

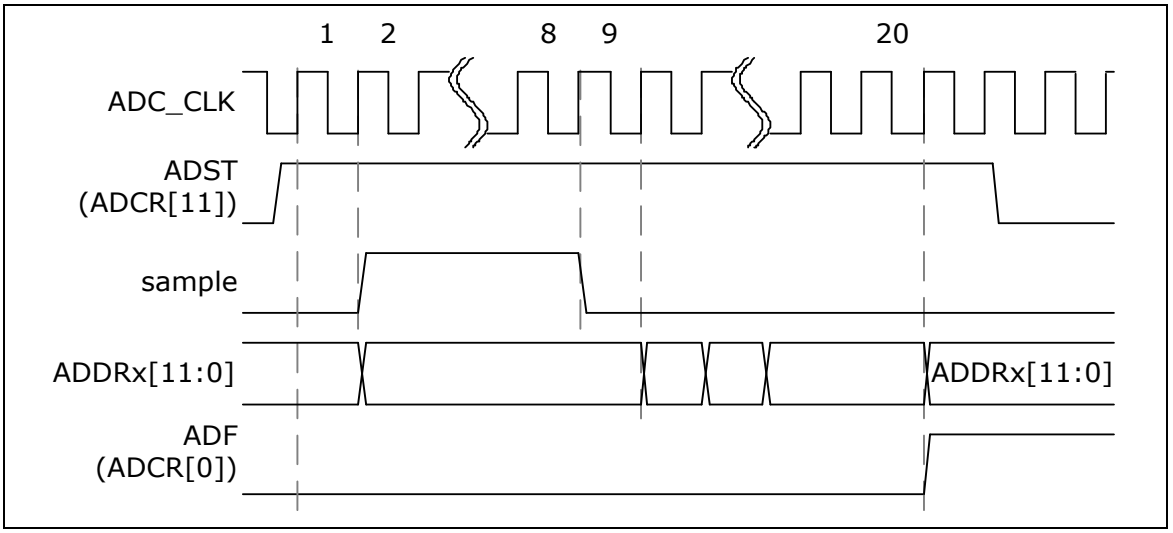


图 5-154 单一转换模式时序图

5.21.5.3 *单周期扫描模式*

在单周期扫描模式下，A/D转换器会对所有指定的通道进行一次采样和转换，通道转换顺序为编号最小的通道到编号最大的通道依次开始。

1. 当 ADST (ADCR[11]) 位被软件或外部触发输入置为 1 时， A/D 将从最小编号的通道开始转换。
2. 每路A/D转换完成后，转换结果将会传输到相应通道的A/D数据寄存器中。
3. 当所有选中的通道都完成转换后，ADF (ADSR[0]) 位会被置1。若此时ADC中断功能使能，则产生ADC中断。
4. A/D转换结束后，ADST位自动清零， A/D转换器进入IDLE状态。 如果在所有使能ADC通道转换完成之前，ADST位被清除为0，则ADC控制器将完成当前转换，并将转换结果存储到当前转换通道 的ADDRx中。

下图为使能多个通道（0，2，3，7）的单周期扫描模式时序图：

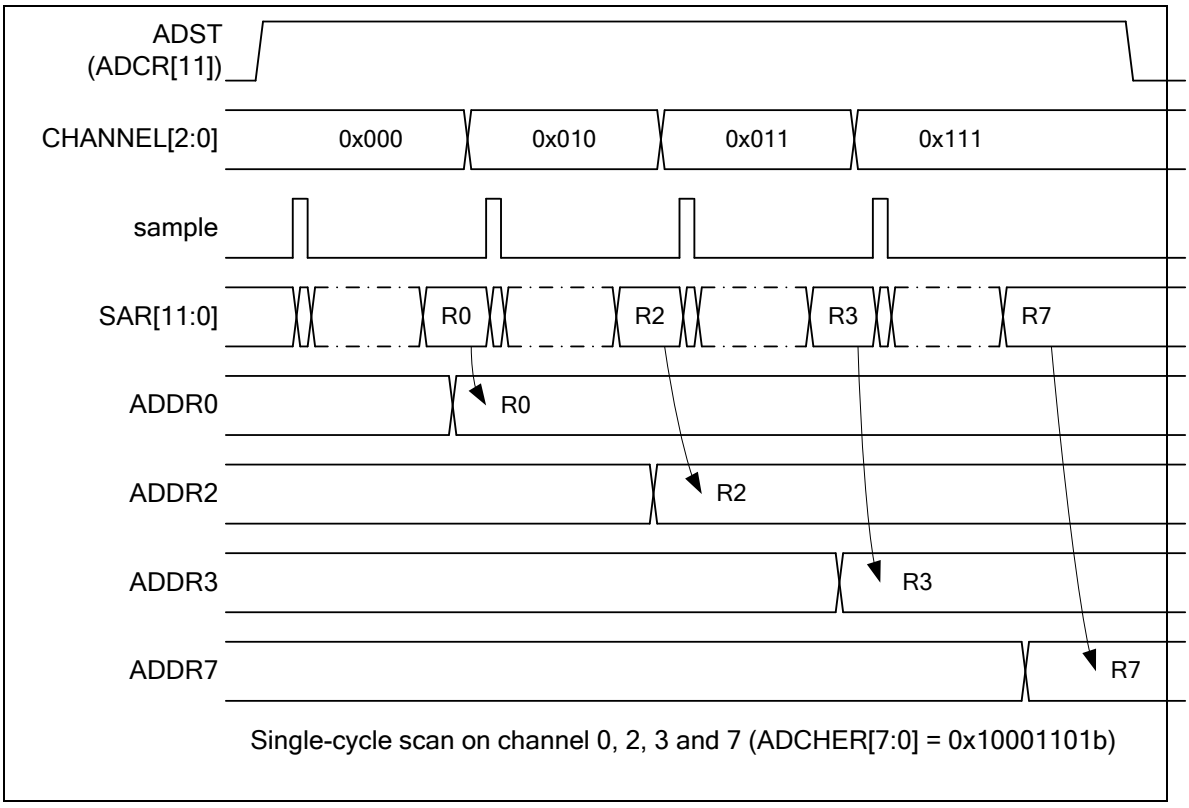


图 5-155 使能通道上的单周期扫描模式时序图

5.21.5.4 *连续扫描模式*

在连续扫描模式下，A/D转换器会对所有由CHEN (ADCHER[7：0]) 使能的通道进行连续的循环扫描。操作如下：

1. 当ADCR的ADST(ADCR[11])被软件设置为1 时，A/D转换将会从编号最小的通道开始。

2. 每路使能通道的A/D转换完成后，A/D转换结果将被装载到相应通道的A/D数据寄存器中。
3. 当所有被使能的通道依次完成一次A/D转换后，ADF (ADSR[0]) 位将会被置1。如果此时ADC中断功能已经使能，则产生中断。如果软件没有清除ADST位， 则又会从最小编号的使能通道开始新的转换。
4. 只要ADST保持为1，就会不断重复步骤2到步骤3 。 当ADST被清为0时， ADC转换器将停止转换。

下图为使能多个通道（0，2，3，7）的连续扫描模式时序图

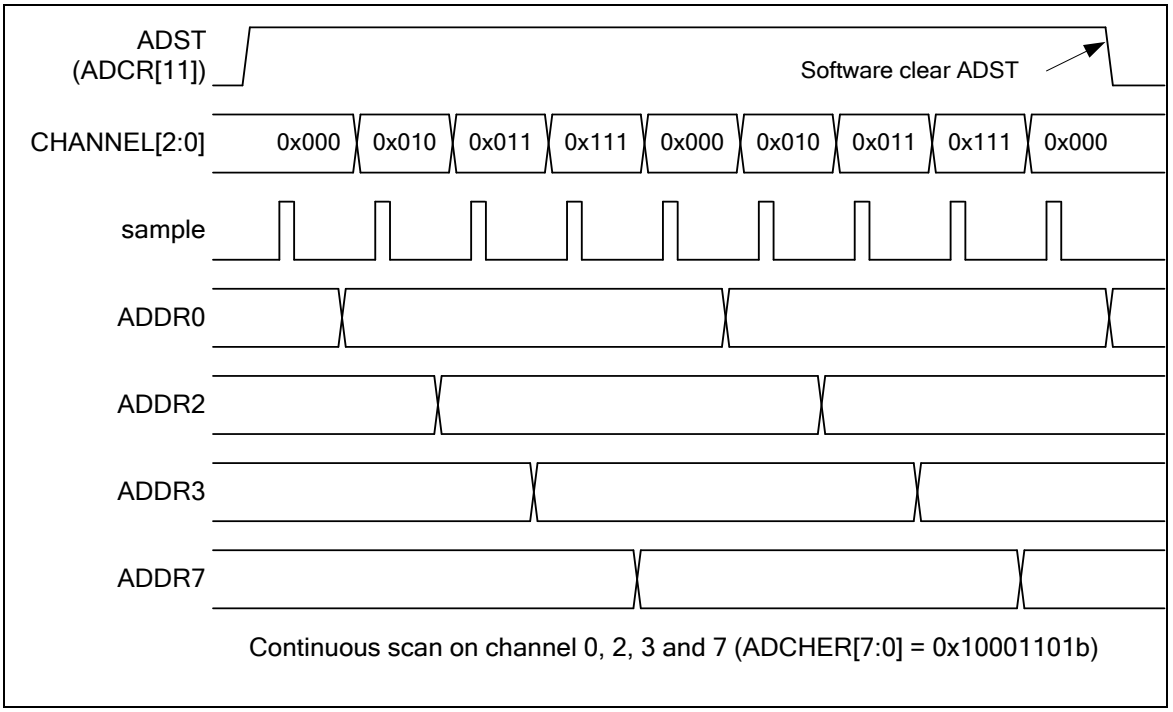


图 5-156 使能通道上的连续扫描模式时序图

5.21.5.5 *外部触发输入采样和A/D转换时间*

在单周期扫描模式下，A/D转换可由外部管脚触发转换。当设置TRGEN (ADCR[8])为1用来使能ADC外部触发功能时，设定TRGS (ADCR[5:4])为00b选择STADC管脚做为外部触发输入， 再设定TRGCOND (ADCR[7:6]) 来设置触发条件是上/下边缘触发还是高/低电平触发。如果选择电平触发，STADC管脚需要保持设定状态至少8个PCLKs。 ADST将在第9个PCLK时被置为1并且开始进行转换。 在电平触发模式状态下，如果外部触发输入维持在有效状态，转换将会持续进行。仅当外部触发条件消失才停止。 若选择边缘触发模式， 高或低的状态至少需要保持4个PCLKs，低于该值的脉冲将被忽略。

5.21.5.6 *PWM中央对齐触发*

在单周期扫描模式下，如果设置TRGEN (ADCR[8])为1并且将TRGS (ADCR[5:4])设置为11b， PWM就可以作为ADC的触发源。当使能PWM触发ADC功能时，如果PWM计数器计到PWM中点时将产生触发信号给ADC。

5.21.5.7 *比较监控转换结果*

ADC控制器提供两组比较寄存器ADCMPR0和ADCMPR1，用于监控A/D转换控制器（最多支持）2路通道的转换结果值，可参考图6-18。软件可通过设定CMPCH (ADCMPRx[5：0]来选择监控哪路通道，而CMPCOND (ADCMPR0/1[2])则用于检查转换值小于或大于（等于）CMPD[11：0]的指定值。当CMPCH指定的通道转换完成时，比较行为将会被自动地触发一次。当比较结果和设定值相匹配，比较匹配计数器将加1，否则匹配计数器将会被清0。当计数器的值和设定值(CMPMATCNT (ADCMPR0/1 [11:8])+1)匹配，CMPF0/1 位 (ADSR[1]/[2])将会置1。如果CMPIE 位 (ADCMPR0/1 [1])为1，将产生ADC_INT中断请求。使用这个功能，在扫描模式下，无需软件介入就可监控外部模拟输入管脚电压变化。具体逻辑框图如下：

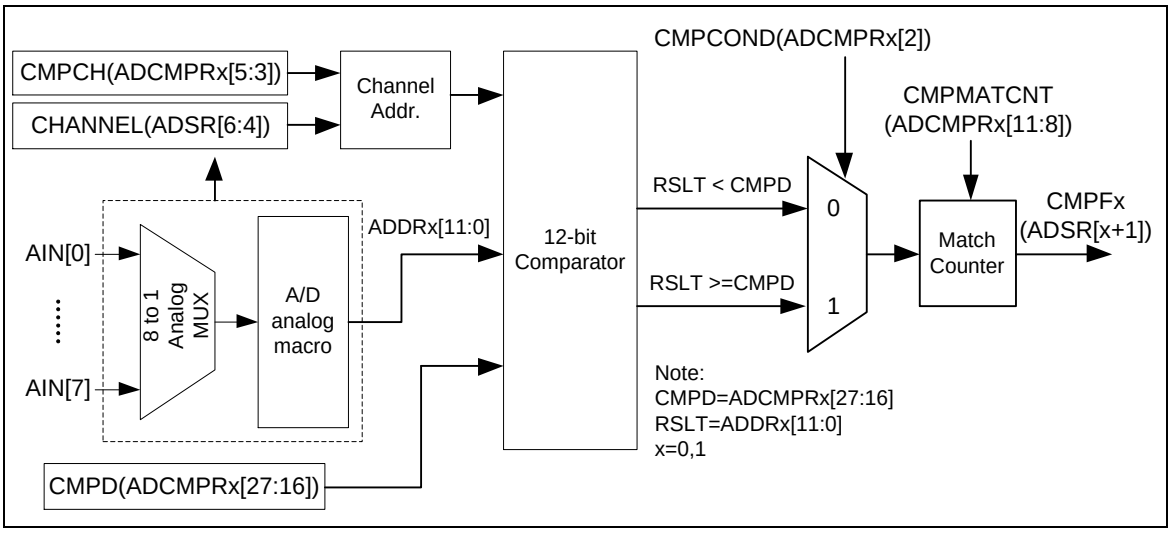


图 5-157 A/D 转换结果监控逻辑图

5.21.5.8 *中断源*

A/D转换器有三个中断源。当某个ADC操作模式结束其转换时，A/D转换的结束标志ADF位将被置1。CMPF0 (ADSR[1]) 和CMPF1 (ADSR[2])是比较功能的比较标志。当A/D转换结果同ADCMPR0/1寄

存器设定值匹配时，相应的比较标志（CMPF0/1）会被置1。当ADF、CMPF0和CMPF1中任意一个标志被置1，且其相应中断使能位ADIE(ADCR[1]) 及CMPIE(ADCMPR0/1[1])置1时，将产生ADC中断。软件可通过清除标志位以撤销中断请求。

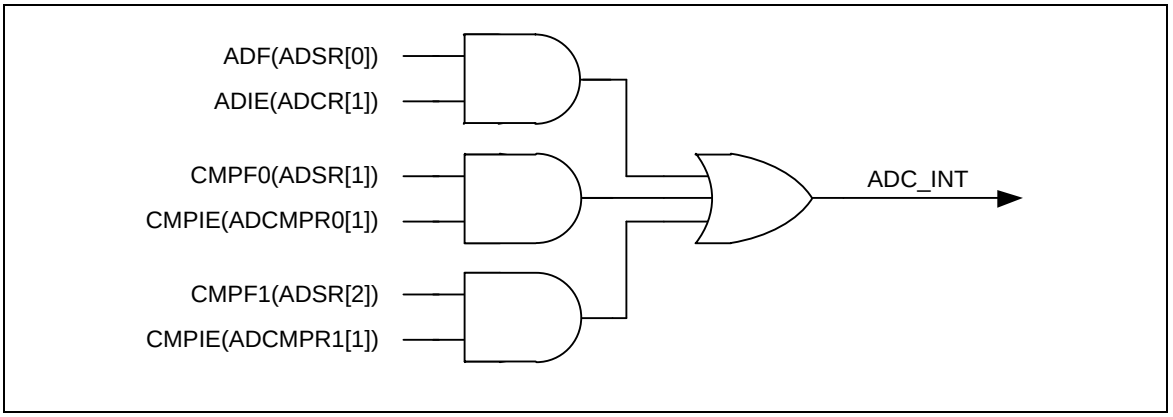


图 5-158 A/D 控制中断

5.21.5.9 *外设DMA请求*

当A/D转换完成时，转换结果将被装载到ADDR寄存器中且VALID位会被置1。如果PTEN 位(ADCR[9])被置1，ADC控制器将产生请求到PDMA。用户可使用PDMA将转换结果传输到用户指定的内存空间，而无需CPU参与。不管选择哪个通道，PDMA操作的源地址都为ADPDMA。如果ADC工作在单周期或连续扫描模式，当PDMA传输转换结果时，ADC将继续转换所选择的下一个通道。用户可以通过读寄存器ADPDMA监控当前PDMA传输数据。如果ADC完成所选通道的转换，且同通道上次的转换结果还没有被PDMA传输，则相应通道的OVERRUN位将置1，上次ADC转换结果将被新的ADC转换结果覆盖。PDMA将传输所选通道的最后数据到用户指定的目的地址。

5.21.6 寄存器映射

R: 只读, **W**: 只写, **R/W**:可读/写

寄存器	偏移地址	R/W	描述	复位值
ADC 基地址: **ADC_BA = 0x400E_0000**				
ADDR0	ADC_BA+0x00	R	A/D数据寄存器0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D数据寄存器1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D数据寄存器2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D数据寄存器3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D数据寄存器4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D数据寄存器5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D数据寄存器6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D数据寄存器7	0x0000_0000
ADCR	ADC_BA+0x20	R/W	ADC控制寄存器	0x0000_0000
ADCHER	ADC_BA+0x24	R/W	ADC 通道使能寄存器	0x0000_0000
ADCMPR0	ADC_BA+0x28	R/W	ADC 比较寄存器0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	ADC比较寄存器1	0x0000_0000
ADSR	ADC_BA+0x30	R/W	ADC 状态寄存器	0x0000_0000
ADPDMA	ADC_BA+0x40	R	ADC PDMA 当前传输数据寄存器	0x0000_0000

5.21.7 寄存器描述

ADC数据寄存器 (ADDR0 ~ ADDR7)

寄存器	偏移地址	R/W	描述	复位值
ADDR0	ADC_BA+0x00	R	A/D数据寄存器0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D数据寄存器1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D数据寄存器2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D数据寄存器3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D数据寄存器4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D数据寄存器5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D数据寄存器6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D数据寄存器7	0x0000_0000

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

位	描述	
[31:18]	**Reserved**	保留.
[17]	**VALID**	**有效标志** 0=RSLT(ADDRx[15:0], x=0~7)中的数据无效 1=RSLT(ADDRx[15:0], x=0~7)中的数据有效 当相应的模拟通道转换完成后，该位置1。读ADDR寄存器后，该位由硬件自动清除。 该位只读
[16]	**OVERRUN**	**Overrun 标志** **0=** RSLT (ADDRx[15:0], x=0~7)中的数据是最新的转换结果 1= RSLT (ADDRx[15:0], x=0~7)中的数据被覆盖过 如果在新的转换结果载入到RSLT (ADDRx[11:0]， x=0~7)寄存器之前,已经在RSLT (ADDRx[11:0]， x=0~7)的转换结果还没有被读取，则OVERRUN 标志将被置1，之前的转换结果也会丢失。在读ADDR寄存器后，该位由硬件自动清零。 该位只读

[15:0]	**RSLT**	**A/D 转换结果** 该域存放ADC的转换结果 当DMOF位(ADCR[31])设置为0， 12-位ADC转换结果为无符号 格式，且存放在RSLT (ADDRx[11:0], x=0~7)，而RSLT (ADDRx[15:12], x=0~7)用0填充 。 当DMOF位(ADCR[31])设置为1， 12-位ADC转换结果为二进制的补码格式，且存放在RSLT (ADDRx[11:0], x=0~7)，而RSLT (ADDRx[15:12], x=0~7)将存放符号位 。

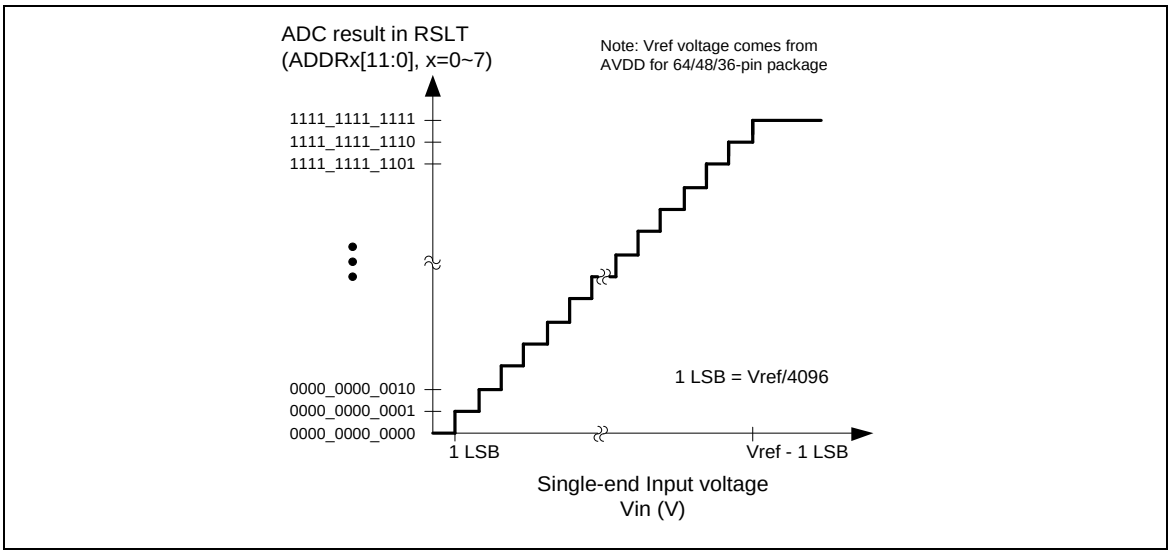


图 5-159 ADC 单端输入电压和转换结果图

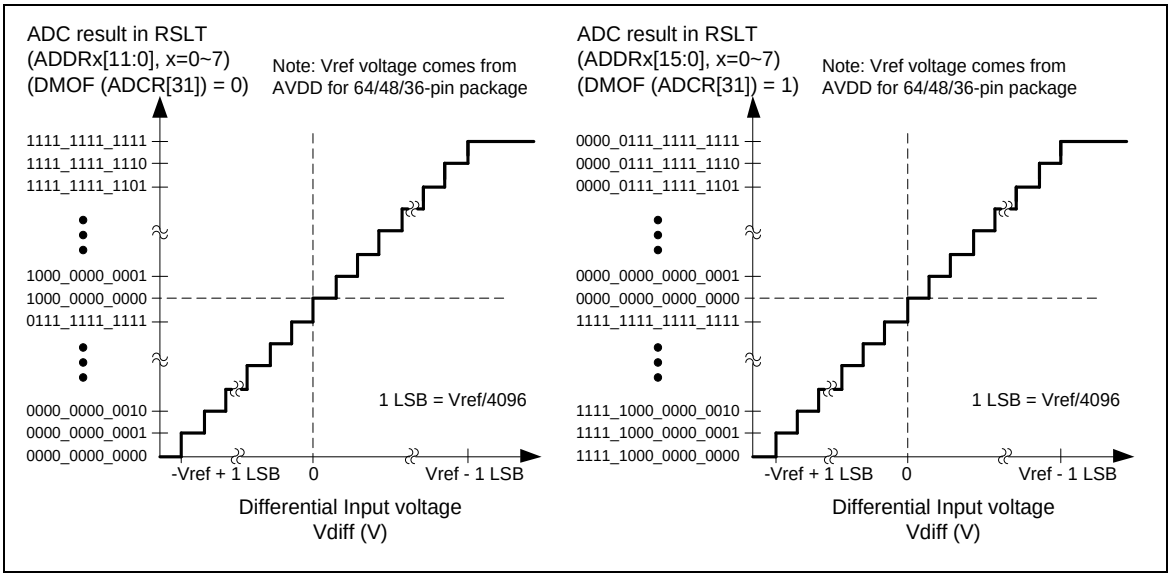


图 5-160 ADC 差分输入电压和转换结果图

ADC控制寄存器（ADCR)

寄存器	偏移地址	R/W	描述	复位值
ADCR	ADC_BA+0x20	R/W	ADC 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
DMOF	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**ADST**	**DIFFEN**	**PTEN**	**TRGEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TRGCOND		**TRGS**		**ADMD**		**ADIE**	**ADEN**

位	描述	
[31]	**DMOF**	**A/D差分输入模式的输出格式** 0=A/D转换结果存储在ADDRX寄存器的RSLT，格式为无符号格式 1= A/D转换结果存储在ADDRX寄存器的RSLT，格式为二进制补码格式
[30:12]	**Reserved**	保留.
[11]	**ADST**	**A/D转换开始** 0=转换停止，A/D转换器进入空闲状态 1=转换开始 ADST位可以通过三种方式设置为1，分别为：软件设定，PWM中央对齐触发和外部STADC管脚触发。单一模式和单周期模式下，在转换结束后，ADST将被硬件自动清除。在连续扫描模式下，A/D转换将一直进行直到软件向该位写0或芯片复位。
[10]	**DIFFEN**	**差分输入模式使能** 0=单端模拟输入模式 1=差分模拟输入模式

差分输入对通道	ADC 模拟输入	
	V_{plus}	V_{minus}
0	ADC0	ADC1
1	A C2	ADC3
2	A C4	ADC5
3	ADC6	AD 7

差分输入电压(V_{diff}) = V_{plus} - $V_{minu.}$ 其中V_{plus}是模拟正向输入，V_{minus}是模拟反向输入。

在差分输入模式下，只需要在ADCHER使能两个相应通道的偶数通道即可。转换结果将放置于相应的使能通道的寄存器里

[9]	**PTEN**	**PDMA传输使能** 0=禁止使用PDMA数据传输 1=使能PDMA传输ADDR0-7中的数据 当A/D完成转换后，转换的结果将会载入到ADDR0-7， 软件可以使能该位来产生PDMA数据传输请求。 当PTEN=1，软件必须设定ADIE=0 ，禁止产生A/D 中断
[8]	**TRGEN**	**硬件触发使能位** 使能或禁止通过硬件方式（外部STADC管脚或PWM 中央对齐触发）触发A/D转换 0=禁止 1=使能 ADC硬件触发功能 只能在单周期扫描模式下支持
[7:6]	**TRGCOND**	**外部触发条件** 这2位决定使用外部管脚STADC触发条件为电平触发还是边沿触发。电平触发信号必须保持至少8 PCLKs的稳定状态，而边沿触发信号必须保持至少4 PCLKs的高或低状态，触发条件才能成立。 00=低电平 01=高电平 10=下降沿 11=上升沿
[5:4]	**TRGS**	**硬件触发源** 00=A/D转换由外部STADC管脚触发开始 11=A/D转换由PWM中央对齐触发开始 其他=保留 改变TRGS前，软件必须先禁用TRGEN和ADST
[3:2]	**ADMD**	**A/D转换器工作模式** 00=单一转换 01=保留 10=单周期扫描 11=连续扫描 当改变工作模式时，软件须先禁止ADST位。
[1]	**ADIE**	**A/D中断使能** 0=禁止A/D中断 1=使能A/D中断 如果ADIE 位 (ADCR[1])被置1，则A/D转换结束后将产生中断请求
[0]	**ADEN**	**A/D装换器使能位** 0=禁止 1=使能 在开始A/D转换前，该位需设置为1。清除该位为0将禁用A/D转换器模拟电路从而节省功耗。

ADC通道使能寄存器 (ADCHER)

寄存器	偏移地址	R/W	描述	复位值
ADCHER	ADC_BA+0x24	R/W	ADC 通道使能寄存器	0x0000_0000

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

位	描述	
[31:10]	**Reserved**	保留
[9:8]	**PRESEL**	**模拟输入通道7选择** 00=外部模拟输入 01=内部带隙电压 10=内部温度传感器 11=保留
[7:0]	**CHEN**	**模拟输入通道使能位** 设置CHEN[7:0]用于使能对应的模拟输入通道7~0。如果DIFFEN位(ADCR[10])设置为1，只有偶数位通道需要使能 0=禁止ADC输入通道 1=使能ADC输入通道

ADC比较寄存器 0/1 (ADCMPR0/1)

寄存器	偏移地址	R/W	描述	复位值
ADCMPR0	ADC_BA+0x28	R/W	ADC比较寄存器0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	ADC 比较寄存器1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPD[11:8]			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPD[7:0]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				CMPMATCNT			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		CMPCH			CMPCOND	CMPIE	CMPEN

位	描述	
[31:28]	**Reserved**	保留.
[27:16]	**CMPD**	**比较数值** 此12-BIT数据用于和指定通道的转换结果进行比较 当DMOF位(ADCR[31])设置为0时，CMPD将与无符号格式转换结果进行比较。CMPD中数据格式也必须是无符号格式。 当DMOF位设置为1时，CMPD将与二进制补码格式转换结果进行比较。CMPD中数据格式也必须是二进制补码格式
[15:12]	**Reserved**	保留
[11:8]	**CMPMATCNT**	**比较匹配计数** 当指定A/D通道的模拟转换值和比较条件CMPCOND (ADCMPR0/1[2])相匹配时，内部计数器将加1，当内部计数器达到设定值(CMPMATCNT (ADCMPR0/1[11:8]) +1)时，将置CMPF0/1位(ADSR[1]/[2])为1
[7:6]	**Reserved**	保留
[5:3]	**CMPCH**	**比较通道选择** 000=选择通道0转换结果进行比较 001=选择通道1转换结果进行比较 010=选择通道2转换结果进行比较 011=选择通道3转换结果进行比较 100=选择通道4转换结果进行比较 101=选择通道5转换结果进行比较 110=选择通道6转换结果进行比较 111=选择通道7转换结果进行比较

[2]	**CMPCOND**	**比较条件** 0=设置比较条件为：当12-位A/D转换结果小于 12-位CMPD (ADCMPR0/1[27:16])，内部匹配计数器将加1 1=设置比较条件为：当12-位A/D转换结果大于或等于 12-位CMPD (ADCMPR0/1[27:16])，内部匹配计数器将加1 **注意：**当内部计数器达到(CMPMATCNT (ADCMPR0/1[11:8])+1)，CMPF0/1 bit (ADSR[1]/[2])将会被置1
[1]	**CMPIE**	**比较中断使能位** 0=禁止比较中断 1=使能比较中断 如果使能了比较功能，而且比较条件满足CMPCOND和CMPMATCNT中的设定，则CMPF位将会被置1，同时，如果CMPIE为1，将产生比较中断请求
[0]	**CMPEN**	**比较使能位** 0=禁止比较功能 1=使能比较功能 设置此位为1可使能ADC控制器 将指定通道转换结果载入到ADDR寄存器后与CMPD (ADCMPR0/1[27:16])中数据进行比较。

ADC状态寄存器（ADSR)

寄存器	偏移地址	R/W	描述	复位值
ADSR	ADC_BA+0x30	R/W	ADC 状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
OVERRUN							
15	14	13	12	11	10	9	8
VALID							
7	6	5	4	3	2	1	0
Reserved	CHANNEL			BUSY	CMPF1	CMPF0	ADF

位	描述	
[31:24]	**Reserved**	保留
[23:16]	**OVERRUN**	**Overrun标志** **该位为ADDRX中OVERRUN位的镜像** **该位只读**
[15:8]	**VALID**	**Data Valid标志** **该位为ADDRX中Data Valid位的镜像** **该位只读**
[7]	**Reserved**	保留
[6:4]	**CHANNEL**	**当前转换通道** **当BUSY=1时，该域反映当前转换通道号。当BUSY=0时，它表示下一个要转换的通道号** 该位只读
[3]	**BUSY**	**BUSY/IDLE** 0=A/D转换器处于空闲状态 1=A/D转换器处于忙状态 该位为ADST位(ADCR[11])的镜像位. 该位只读.
[2]	**CMPF1**	**比较标志** **当所选择的通道的A/D转换结果和ADCMPR1的设定条件匹配时，该位置1。该位写1清除 。** **0=ADDR中的转换结果和ADCMPR1 的设定值不匹配** **1=ADDR中的转换结果和ADCMP1的设定值匹配**
[1]	**CMPF0**	**比较标志** **当所选择的通道的A/D转换结果和ADCMPR0的设定条件匹配时，该位置1。该位写1清除 。** **0=ADDR中的转换结果和ADCMPR0 的设定值不匹配** **1=ADDR中的转换结果和ADCMP0的设定值匹配**

[0]	**ADF**	**A/D转换结束标志** **该状态标志指示A/D转换结束** **ADF在以下两个条件下被置1：** **1. 当在单次模式下 A/D 转换结束** **2. 在扫描模式下，所有指定的通道 A/D 转换结束** **该标志写1清除**

ADC PDMA当前传输数据寄存器 (ADPDMA)

寄存器	偏移地址	R/W	描述	复位值
ADPDMA	ADC_BA+0x40	R	ADC PDMA 当前传输数据寄存器	0x0000_0000

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

位	描述	
[31:18]	**Reserved**	保留
[17:0]	**AD_PDMA**	**ADC PDMA当前传输数据寄存器** 当PDMA传输时，读该寄存器可以监测当前PDMA的传输数据。 当前PDMA传输数据为ADDR0 ~ ADDR7中内容 此为只读寄存器

5.22 模拟比较器 (ACMP)

5.22.1 概述

NuMicro™ NUC200系列包含有2路模拟比较器，可按照比较器配置应用于不同的场合。当正极输入电压大于负极输入电压时，比较器输出逻辑1，否则输出逻辑0。当比较器输出值改变时，每路比较器都可通过配置产生中断。比较器模块框图见图6-10。

5.22.2 特性

- 模拟输入电压范围：0~ V_{DDA} (即AV_{DD} 引脚的电压)
- 支持迟滞功能（Hysteresis function）
- 每路比较器的负极可选择内部参考电压输入

5.22.3 模块图

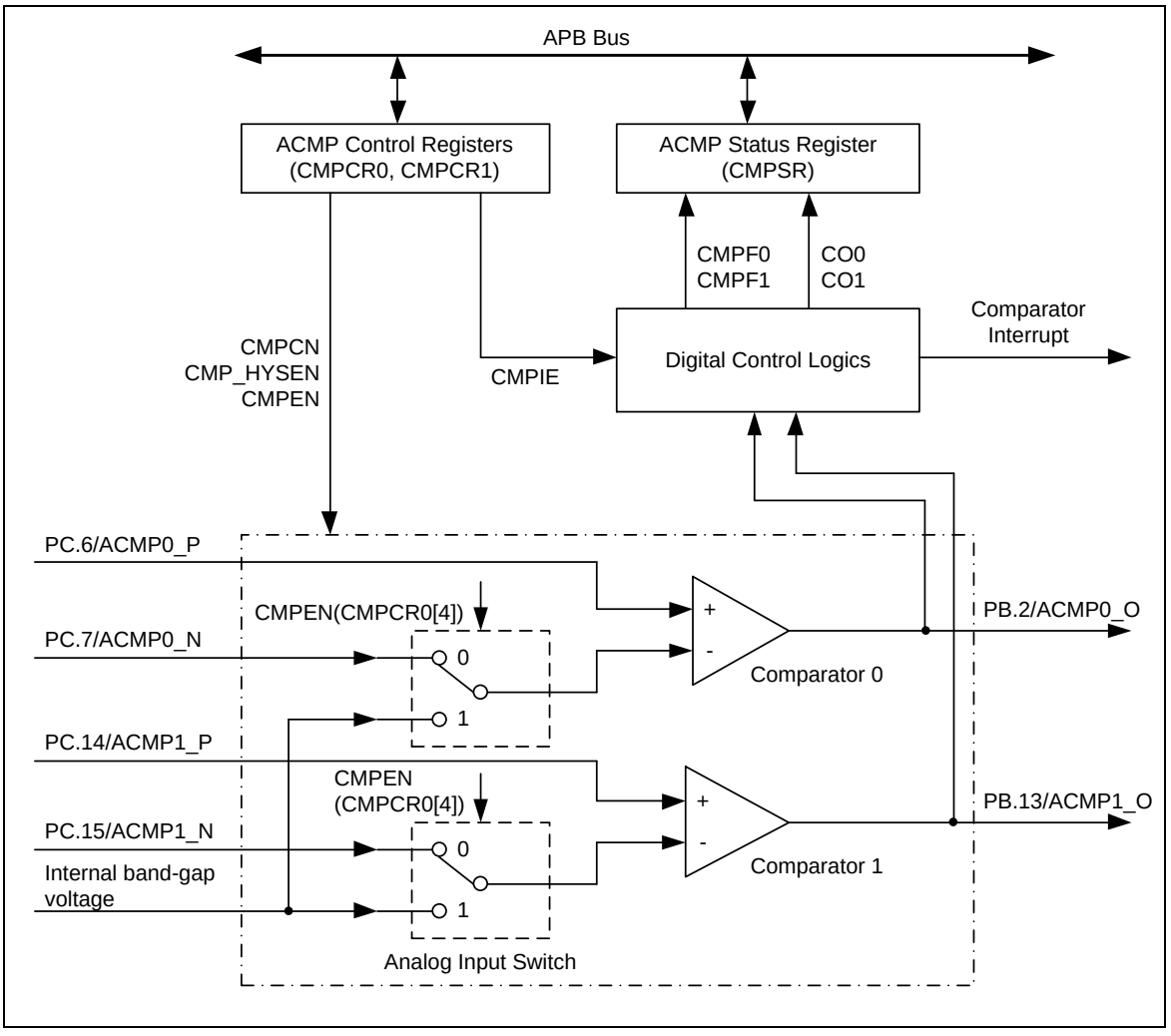


图 5-161 模拟比较器模块框图

5.22.4 基本配置

模拟比较器的功能引脚由寄存器GPB_MFP, GPC_MFP, ALT_MFP, ALT_MFP1 和 ALT_MFP2 中的配置决定。如果被定义为比较器模拟输入引脚，建议将该引脚的数字输入通道关闭，以避免产生漏电流。引脚数字输入通道可通过配置寄存器GPIOC_OFFD关闭。

寄存器APBCLK[30]用于使能模拟比较器的模块时钟。

5.22.5 功能描述

5.22.5.1 *中断源*

比较器输出在通过 PCLK采样后，结果会被写入到 寄存器 CMPSR的CO1和CO2中。如果CMPOCR/CMP1CR的CMP0IE/CMP1IE设置为1，则模拟比较器的中断将被使能。 当比较器的输出状态改变时，则会产生比较器中断请求，相应的比较器中断标志CMPF0或CMPF1 将被置位。软件可写1到标志位清除该位。

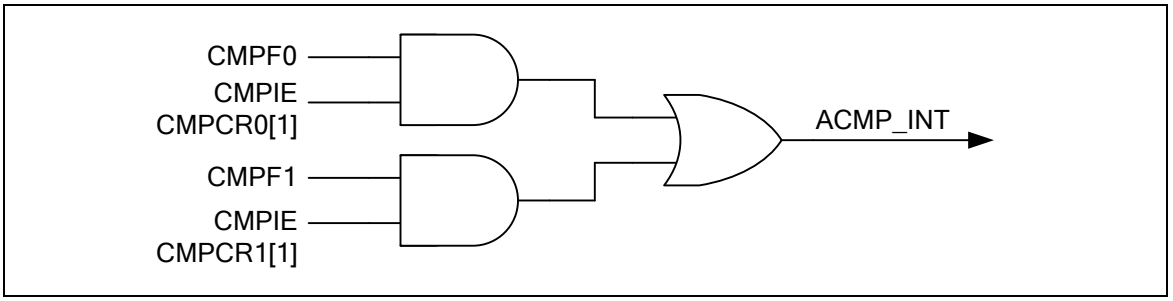


图 5-162 比较器控制器中断源

5.22.5.2 *迟滞功能*

模拟比较器提供迟滞功能用于使比较器转换输出变化更平稳。当比较器输出为0时，比较器将会一直输出为0直到正极输入电压与负极输入电压的电压差大于等于迟滞电压为止。同样的，当比较器输出为1时，比较器将会一直输出为1直到正极输入电压值与负极输入电压的电压差小于等于迟滞电压为止。

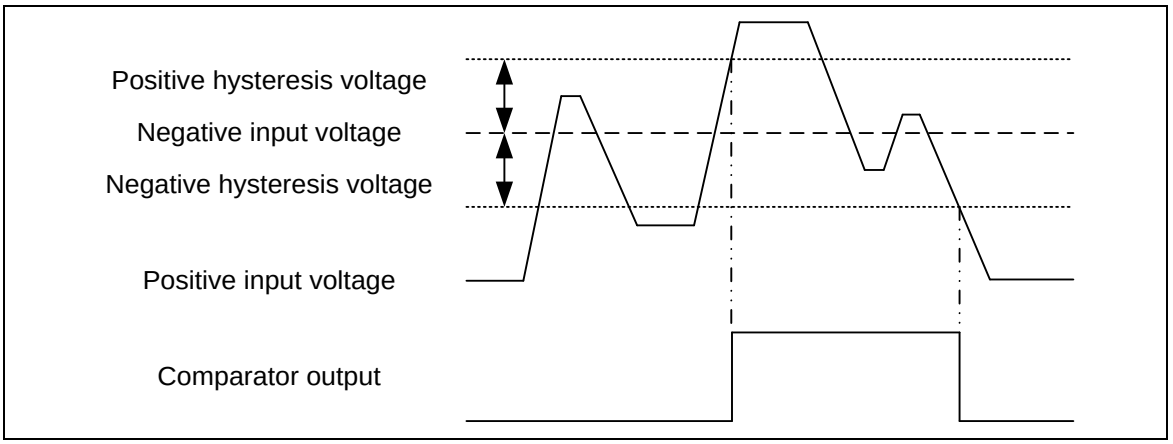


图 5-163 比较器的迟滞功能

5.22.6 寄存器表

R：只读，W：只写， R/W：可读/写

寄存器	偏移地址	R/W	描述	复位值
ACMP基地址: **ACMP_BA = 0x400D_0000**				
CMPCR0	ACMP_BA+0x00	R/W	模拟比较器**0**控制寄存器	0x0000_0000
CMPCR1	ACMP_BA+0x04	R/W	模拟比较器**1**控制寄存器	0x0000_0000
CMPSR	ACMP_BA+0x08	R/W	模拟比较器状态寄存器	0x0000_0000

5.22.7 寄存器描述

CMP控制寄存器 0/1 (CMPCR0/1)

寄存器	偏移地址	R/W	描述	复位值
CMPCR0	ACMP_BA+0x00	R/W	模拟比较器0控制寄存器	0x0000_0000
CMPCR1	ACMP_BA+0x04	R/W	模拟比较器1控制寄存器	0x0000_0000

31	30	29	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	**CMPINV**	**Reserved**	**CMPCN**	**Reserved**	**CMP_HYSEN**	**CMPIE**	**CMPEN**

位	描述	
[31:5]	**Reserved**	保留
[6]	**CMPINV**	**比较器输出反向使能位** 0=禁止比较器输出反向 1=使能比较器输出反向
[5]	**Reserved**	保留
[4]	**CMPCN**	**比较器负极输入选择** 0=选择ACMPn_(n = 0, 1)管脚作为比较器的负极输入源 1=选择内部带隙基准电压作为比较器的负极输入源
[3]	**Reserved**	保留
[2]	**CMP_HYSEN**	**比较器迟滞使能** 0=禁用迟滞功能（默认） 1=使能迟滞功能
[1]	**CMPIE**	**比较器中断使能位** 0=禁用中断功能 1=使能中断功能
[0]	**CMPEN**	**比较器使能位** 0=禁用比较器 1=使能比较器

CMP状态寄存器 (CMPSR)

寄存器	偏移地址	R/W	描述	复位值
CMPSR	ACMP_BA+0x08	R/W	模拟比较器状态寄存器	0x0000_0000

31	30	29	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				**CO1**	**CO0**	**CMPF1**	**CMPF0**

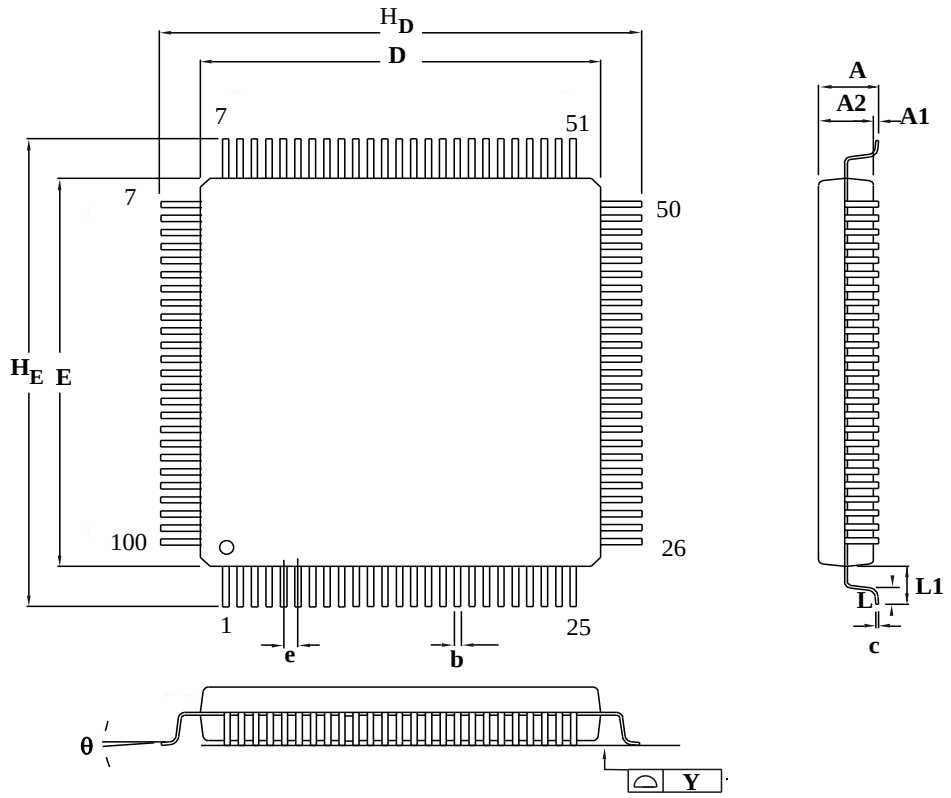
位	描述	
[31:4]	**Reserved**	保留
[3]	**CO1**	**比较器1输出** 同步APB时钟并允许软件访问，当比较器禁用（CMPCR1[0] = 0）时清除该位
[2]	**CO0**	**比较器0输出** 同步APB时钟并允许软件访问，当比较器禁用（CMPCR0[0] = 0）时清除该位
[1]	**CMPF1**	**比较器1中断标志** 每当比较器1输出状态改变时，该位将被硬件置位。如果CMPCR1[1]=1，则产生中断。 写1清除该位为0。
[0]	**CMPF0**	**比较器0中断标志** 每当比较器0输出状态改变时，该位将被硬件置位。如果CMPCR0[1]=1，则产生中断。 写1清除该位为0。

6 电气特性

如需NuMicro™ NUC230/240 系列芯片电气特性, 请参看NuMicro™NUC230/240 系列Datasheet.

7 封装尺寸

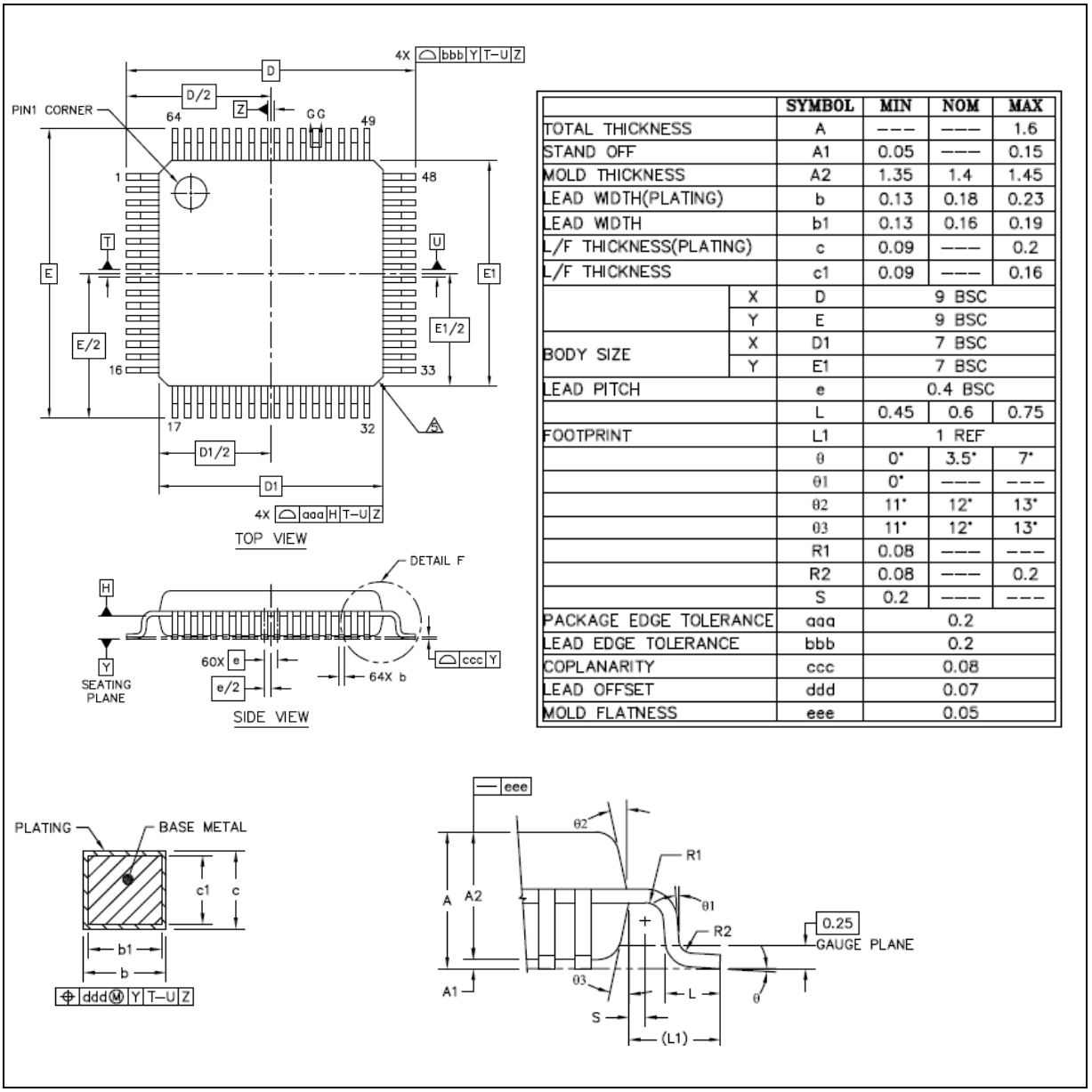
7.1 100-pin LQFP (14x14x1.4 mm 封装 2.0 mm)



Controlling Dimension : Millimeters

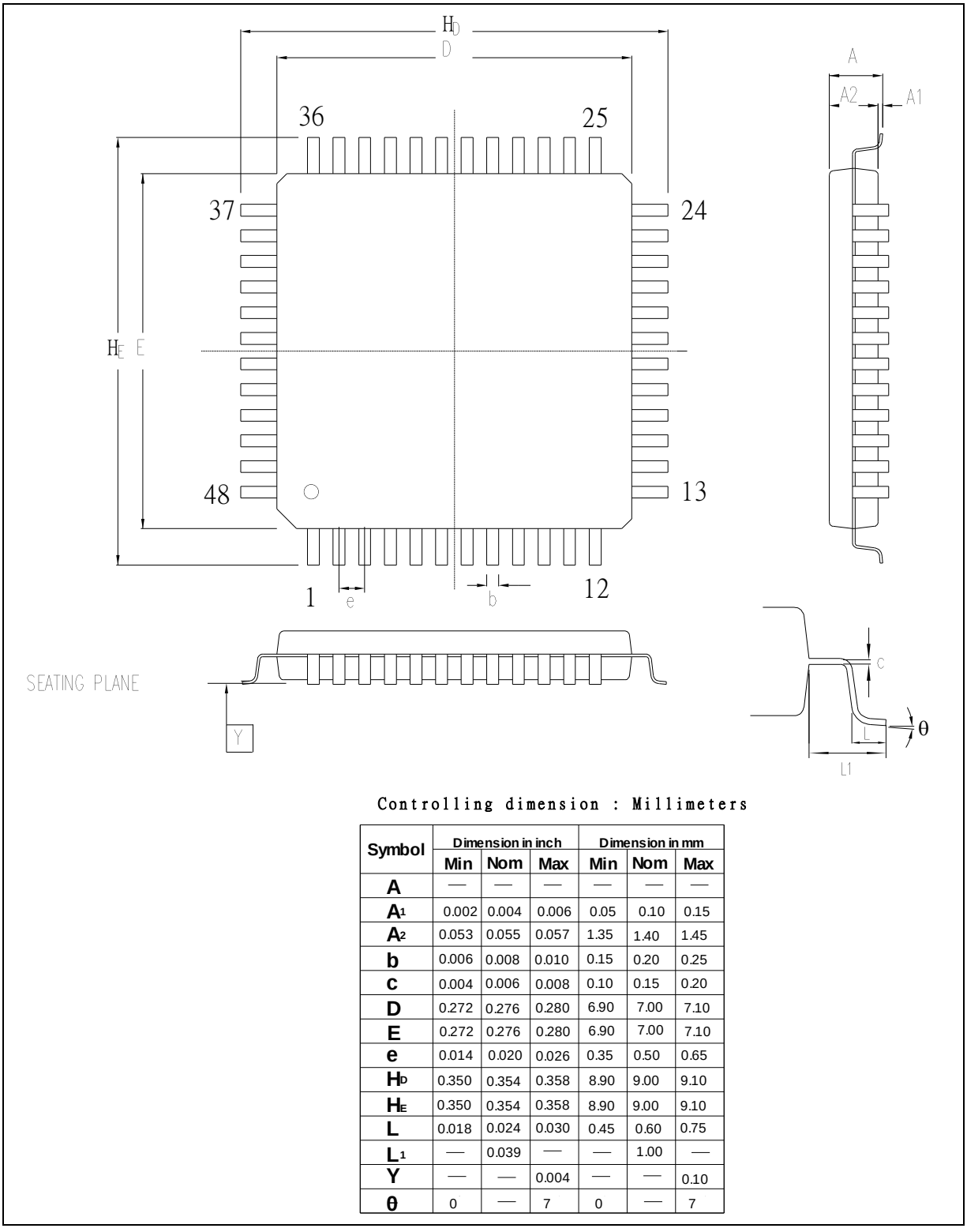
Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	0.063	—	—	1.60
A1	0.002	—	—	0.05	—	—
A2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.007	0.009	0.011	0.17	0.22	0.27
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.547	0.551	0.556	13.90	14.00	14.10
E	0.547	0.551	0.556	13.90	14.00	14.10
e	—	0.020	—	—	0.50	—
H_D	0.622	0.630	0.638	15.80	16.00	16.20
H_E	0.622	0.630	0.638	15.80	16.00	16.20
L	0.018	0.024	0.030	0.45	0.60	0.75
L1	—	0.039	—	—	1.00	—
y	—	—	0.004	—	—	0.10
θ	0°	—	7°	0°	—	7°

7.2 64-pin LQFP (7x7x1.4 mm 封装 2.0 mm)



		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		

7.3 48-pin LQFP (7x7x1.4 mm 封装 2.0 mm)



Controlling dimension : Millimeters

Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	—	—	—	—
A_1	0.002	0.004	0.006	0.05	0.10	0.15
A_2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.006	0.008	0.010	0.15	0.20	0.25
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.272	0.276	0.280	6.90	7.00	7.10
E	0.272	0.276	0.280	6.90	7.00	7.10
e	0.014	0.020	0.026	0.35	0.50	0.65
H_D	0.350	0.354	0.358	8.90	9.00	9.10
H_E	0.350	0.354	0.358	8.90	9.00	9.10
L	0.018	0.024	0.030	0.45	0.60	0.75
L_1	—	0.039	—	—	1.00	—
Y	—	—	0.004	—	—	0.10
θ	0	—	7	0	—	7

8 修订历史

日期	版本	描述
2014.08.07	1.00	1. 初版
2015.01.28	1.02	1. 增加 EBI 功能 2. 章节重新排序

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.