



NuMicro™ NUC131 系列 技术参考手册

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

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

All data and specifications are subject to change without notice.

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

www.nuvoton.com



目录

1 概述	14
2 特性	15
3 缩写表	19
4 编号信息列表与管脚定义	20
4.1 NuMicro™ NUC131 系列选型编码	20
4.2 NuMicro™ NUC131 系列选型指南	21
4.3 管脚配置	22
4.3.1 NuMicro™ NUC131 管脚图	22
4.4 管脚描述	24
4.4.1 NuMicro™ NUC131 管脚描述	24
5 方块图	30
5.1 NuMicro™ NUC131 方块图	30
6 功能描述	31
6.1 ARM® Cortex™-M0 内核	31
6.2 系统管理器	33
6.2.1 概述	33
6.2.2 系统复位	33
6.2.3 系统电源分配	34
6.2.4 系统内存映射	35
6.2.5 寄存器映射	37
6.2.6 寄存器描述	38
6.2.7 系统定时器(SysTick)	82
6.2.8 嵌套向量中断控制器 (NVIC)	87
6.2.9 系统控制	113
6.3 时钟控制器	121
6.3.1 概述	121
6.3.2 系统时钟和SysTick 时钟	123
6.3.3 掉电模式时钟	124
6.3.4 分频器输出	125
6.3.5 寄存器映射	126
6.3.6 寄存器描述	127



6.4 Flash存储控制器 (FMC)	149
6.4.1 概述.....	149
6.4.2 特性.....	149
6.4.3 框图.....	150
6.4.4 功能描述	151
6.4.5 寄存器映射.....	162
6.4.6 寄存器描述.....	163
6.5 通用 I/O (GPIO).....	172
6.5.1 概述.....	172
6.5.2 特性.....	172
6.5.3 基本配置	172
6.5.4 功能描述	173
6.5.5 寄存器映射.....	175
6.5.6 寄存器描述.....	178
6.6 定时器控制器(TIMER)	191
6.6.1 概述.....	191
6.6.2 特性.....	191
6.6.3 框图.....	192
6.6.4 基本配置	193
6.6.5 功能描述	193
6.6.6 寄存器映射.....	196
6.6.7 寄存器描述.....	198
6.7 PWM发生器和捕捉时钟(PWM)	207
6.7.1 概述.....	207
6.7.2 特性.....	207
6.7.3 框图.....	209
6.7.4 基本配置	212
6.7.5 功能描述	212
6.7.6 寄存器映射.....	233
6.7.7 寄存器描述.....	237
6.8 基本 PWM 发生器和捕获定时器 (BPWM)	289
6.8.1 概述.....	289
6.8.2 特性.....	289



6.8.3 框图.....	291
6.8.4 基本配置	293
6.8.5 功能描述	293
6.8.6 寄存器描述.....	312
6.9 看门定时狗 (WDT)	345
6.9.1 概述.....	345
6.9.2 特性.....	345
6.9.3 框图.....	346
6.9.4 基本配置	347
6.9.5 功能描述	347
6.9.6 寄存器表	349
6.9.7 寄存器描述.....	350
6.10 窗口看门狗定时器(WWDT)	353
6.10.1 概述.....	353
6.10.2 特性.....	353
6.10.3 框图.....	354
6.10.4 基本配置	355
6.10.5 功能描述	355
6.10.6 寄存器映射.....	357
6.10.7 寄存器描述.....	358
6.11UART 接口控制器 (UART)	363
6.11.1 概述.....	363
6.11.2 特性.....	363
6.11.3 框图.....	364
6.11.4 基本配置	366
6.11.5 功能描述	366
6.11.6 寄存器表	390
6.11.7 寄存器描述.....	392
6.12I²C 总线控制器 (I²C).....	421
6.12.1 概述.....	421
6.12.2 特征.....	421
6.12.3 基本配置	421
6.12.4 框图.....	422



6.12.5 功能描述	422
6.12.6 EEPROM随机读取例子	445
6.12.7 寄存器映射	447
6.12.8 寄存器描述	448
6.13串行外围设备接口 (SPI)	458
6.13.1 概述	458
6.13.2 特性	458
6.13.3 框图	459
6.13.4 基本配置	459
6.13.5 功能描述	460
6.13.6 时序图	468
6.13.7 编程例子	470
6.13.8 寄存器映射	472
6.13.9 寄存器描述	473
6.14控制器局域网(CAN)	488
6.14.1 概述	488
6.14.2 特性	488
6.14.3 框图	488
6.14.4 基本配置	489
6.14.5 功能描述	490
6.14.6 测试模式	491
6.14.7 CAN通信	493
6.14.8 CAN接口复位状态	510
6.14.9 寄存器描述	514
6.14.10 寄存器映射	514
6.15模拟数字转换(ADC)	550
6.15.1 概述	550
6.15.2 特性	550
6.15.3 框图	551
6.15.4 基本配置	551
6.15.5 功能描述	551
6.15.6 寄存器映射	557
6.15.7 寄存器描述	558



7 电器特性	567
8 封装尺寸	568
8.1 64-pin LQFP (7x7x1.4 mm footprint 2.0 mm).....	568
8.2 48-pin LQFP (7x7x1.4 mm footprint 2.0 mm).....	569
9 修订历史	570



图集

图 4-1 NuMicro™ NUC131 系列选型编码	20
图 4-2 NuMicro™ NUC131SxxAE LQFP 64-pin 管脚图	22
图 4-3 NuMicro™ NUC131LxxAE LQFP 48-pin 管脚图	23
图 5-1 NuMicro™ NUC131 方块图	30
图 6-1 功能控制器框图	31
图 6-2 NuMicro™ NUC131 电源分布框图	34
图 6-3 时钟发生器框图	121
图 6-4 时钟发生器全局视图	122
图 6-5 系统时钟框图	123
图 6-6 SysTick 时钟控制框图	123
图 6-7 分频器的时钟源	125
图 6-8 分频器框图	125
图 6-9 Flash 存储器控制框图 (DFVSEN = 1)	150
图 6-10 Flash 存储器控制框图 (DFVSEN = 0)	150
图 6-11 Flash 存储器组织结构 (DFVSEN = 1)	152
图 6-12 Flash 存储器组织结构 (DFVSEN = 0)	153
图 6-13 APROM 和 LDROM 启动的程序执行范围	158
图 6-14 IAP 使能时代码的执行范围	159
图 6-15 BS 位启动选择流程示例	160
图 6-16 ISP 流程示例	161
图 6-17 推挽输出	173
图 6-18 开漏输出	173
图 6-19 准双向 I/O 模式	174
图 6-20 定时器控制器框图	192
图 6-21 定时器控制器时钟源	192
图 6-22 连续计数模式	194
图 6-23 PWM 发生器概述框图	209
图 6-24 PWM 系统时钟源控制	209
图 6-25 PWM 时钟源控制	210
图 6-26 PWM 独立模式架构图	211
图 6-27 PWM 比较模式架构图	212
图 6-28 PWM_CH0 CLKPSC 波形	213



图 6-29 PWM 向上计数方式	213
图 6-30 PWM 向下计数方式	214
图 6-31 PWM上下计数方式	214
图 6-32 在上下计数方式中的 PWM CMPDAT 事件	215
图 6-33 PWM双缓存图解	215
图 6-34 向上计数方式的周期装载模式	216
图 6-35 向上计数方式的立即装载模式	217
图 6-36 上下计数方式的中心装载模式	218
图 6-37 PWM脉冲产生	219
图 6-38 PWM 0% 到 100%占空比脉冲发生	219
图 6-39 PWM 独立模式波形	221
图 6-40 PWM 互补模式波形	221
图 6-41 独立模式 PWM_CH0 输出控制	222
图 6-42 互补模式 PWM_CH0 和 PWM_CH1 输出控制	222
图 6-43 死区插入	223
图 6-44 屏蔽控制波形图解	223
图 6-45 刹车噪音滤波器框图	224
图 6-46 PWM_CH0 和 PWM_CH1 通道组 刹车框图	225
图 6-47 PWM_CH0 和 PWM_CH1 通道组的边缘检测波形	226
图 6-48 PWM_CH0 和 PWM_CH1 通道组的电平检测波形	226
图 6-49 刹车源框图	227
图 6-50 系统故障刹车框图	227
图 6-51 带上升沿死区插入初始状态和极性控制	228
图 6-52 PWM_CH0 和 PWM_CH1 组中断架构图	229
图 6-53 PWM_CH0 和 PWM_CH1 组触发ADC框图	230
图 6-54上下计数方式 PWM 触发 ADC 时序波形	230
图 6-55 PWM_CH0 捕捉框图	231
图 6-56 捕捉操作波形	232
图 6-57 BPWM 产生器框图	291
图 6-58 BPWM 系统时钟源控制	291
图 6-59 BPWM 时钟源控制	292
图 6-60 BPWM 独立模式框图	293
图 6-61 BPWM_CH0 CLKPSC 波形图	294
图 6-62 BPWM 递增计数器类型	294



图 6-63 BPWM 递减计数器类型	295
图 6-64 BPWM 可逆计数器类型	295
图 6-65 BPWM CMPDAT 中断在可逆计数器类型	296
图 6-66 BPWM 双重缓冲区说明	296
图 6-67 在递增计数器中使用周期加载模式.....	297
图 6-68 在递增计数器中使用立刻加载模式.....	298
图 6-69 可逆计数器类型的中间加载模式	299
图 6-70 BPWM 脉冲产生	300
图 6-71 BPWM 0% 到 100% 脉冲产生	300
图 6-72 BPWM_CH0 输出控制的三个步骤.....	301
图 6-73 屏蔽控制波形图.....	302
图 6-74 初始状态和极性控制	303
图 6-75 BPWM_CH0 和 BPWM_CH1一对中断架构图	304
图 6-76 BPWM_CH0 和 BPWM_CH1一对触发ADC框图	305
图 6-77 在可逆计数器类型中 BPWM 触发 ADC 的波形图.....	305
图 6-78 BPWM_CH0 捕获框图	306
图 6-79 捕获操作波形图.....	307
图 6-80 看门狗定时器时钟控制	346
图 6-81 看门狗定时器框图	346
图 6-82 看门狗定时器定时溢出间隔和复位周期时序图.....	348
图 6-83 窗口看门狗定时器时钟控制图	354
图 6-84 窗口看门狗定时器时钟框图.....	354
图 6-85 窗口看门狗定时器复位和重载过程.....	356
图 6-86 UART 串口时钟控制框图	364
图 6-87 UART串口模块框图	365
图 6-88 自动测量波特率.....	369
图 6-89 发送延时操作	369
图 6-90 自动流控模块框图	372
图 6-91 UART CTS 自动流控使能	373
图 6-92 UART RTS 自动流控使能	374
图 6-93 UART RTS 软件控制的流控.....	374
图 6-94 IrDA 控制模块框图	375
图 6-95 IrDA TX/RX 时序图	376
图 6-96 LIN 的帧结构	376

图 6-97 LIN 字节结构	377
图 6-98 LIN 模式下 Break 检测	379
图 6-99 LIN 帧 ID 和奇偶校验格式	380
图 6-100 LIN 同步域的测量	382
图 6-101 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 ..	383
图 6-102 LINS_DUM_EN (UA_LIN_CTL[3])=0 时自动重新同步模式下UA_BAUD的更新次序 ..	383
图 6-103 RS-485 自动方向模式下的 RS-485 RTS 驱动电平	388
图 6-104 RS-485 RTS 软件控制下的驱动电平	388
图 6-105 RS-485 帧结构	389
图 6-106 I ² C 控制器模块框图	422
图 6-107 I ² C 总线时序	422
图 6-108 I ² C 协议	423
图 6-109 起始 (START) 和停止 (STOP) 信号	425
图 6-110 总线上的位传输	427
图 6-111 总线上的应答信号	427
图 6-112 主机向从机传输数据	428
图 6-113 主机向从机读取数据	428
图 6-114 根据 I ² C 当前状态控制总线	429
图 6-115 主机传输模式流程控制	430
图 6-116 主机接收模式控制流程	431
图 6-117 从机模式控制流程	432
图 6-118 广播呼叫模式 (GC)	434
图 6-119 仲裁丢失	436
图 6-120 I ² C 数据移位方向	439
图 6-121 I ² C 超时计数器框图	443
图 6-122 EEPROM 随机读取	445
图 6-123 EEPROM 随机读协议	446
图 6-124 SPI 框图	459
图 6-125 SPI 主机模式应用框图	460
图 6-126 SPI 从机模式应用框图	461
图 6-127 一次传输32位长度 (主模式下)	461
图 6-128 可变总线时钟频率	463
图 6-129 字节重排序功能	464
图 6-130 字节休眠时序波形(主模式)	464

图 6-131 双输出模式的位序列	465
图 6-132 双输入模式的位序列	465
图 6-133 FIFO 模式框图	466
图 6-134 SPI 主机模式下的时序	468
图 6-135 SPI 主机模式下的时序(交替SPI时钟相位)	468
图 6-136 SPI 从机模式下的时序	469
图 6-137 SPI 从机模式下的时序(交替SPI时钟相位)	469
图 6-138 CAN 外设框图	489
图 6-139 CAN 内核静默模式	491
图 6-140 CAN 内核环回模式	492
图 6-141 CAN 内核环回模式和静默模式的组合	492
图 6-142 IFn 寄存器和报文 RAM 间的数据传输	495
图 6-143 应用软件处理 FIFO 缓存	500
图 6-144 位时序	502
图 6-145 传播时间段	503
图 6-146 同步在“late” 和 “early” 边沿	505
图 6-147 过滤短显性毛刺	506
图 6-148 CAN 内核的协议控制器结构	507
图 6-149 ADC 控制器框图	551
图 6-150 ADC 时钟控制	552
图 6-151 单一转换模式时序图	553
图 6-152 使能通道上的单周期扫描模式时序图	554
图 6-153 使能信道上的连续扫描模式时序图	555
图 6-154 A/D 转换结果监控逻辑图	556
图 6-155 A/D 控制器中断	556
图 6-156 ADC 单端输入电压和转换结果图	559
图 6-157 ADC 差分输入电压和转换结果图	559



表集

表 3-1 缩写表	19
表 6-1 片上控制器地址空间分配	36
表 6-2 异常模式.....	88
表 6-3 系统中断映射	89
表 6-4 向量表格式.....	90
表 6-5 芯片空闲/掉电模式控制表	129
表 6-6 存储器地址映像 (DFVSEN = 1)	151
表 6-7 存储器地址映像 (DFVSEN = 0)	152
表 6-8 ISP 命令表	161
表 6-9 PWM & BPWM 特性比较表	208
表 6-10 PWM系统时钟源控制寄存器设置表.....	210
表 6-11 向上计数器PWM脉冲发生事件优先级	219
表 6-12 向下计数器 PWM 脉冲发生优先级.....	220
表 6-13 上下计数器PWM脉冲产生事件优先级	220
表 6-14 PWM & BPWM 特性比较表	290
表 6-15 BPWM 系统时钟源控制寄存器设定表.....	292
表 6-16 递增计数器中 BPWM 脉冲中断优先级.....	301
表 6-17 递减计数器重 BPWM 脉冲中断优先级.....	301
表 6-18 可逆计数器中 BPWM 脉冲中断优先级.....	301
表 6-19 看门狗定时器定时溢出间隔周期	347
表 6-20 窗口看门狗定时器 预分频值选择	355
表 6-21 WINCMP 寄存器设置限制.....	356
表 6-22 串口接口控制脚.....	366
表 6-23 UART 波特率公式	367
表 6-24 UART 控制器波特率参数设置表	367
表 6-25 UART 控制器波特率寄存器 (UA_BAUD) 设置表	368
表 6-26 UART 控制器中断源	371
表 6-27 UART 线的数据位和停止位长度设置	371
表 6-28 UART 线奇偶校验设置	372
表 6-29 LIN 在主模式下的报头选择	377
表 6-30 I ² C 状态码描述	441
表 6-31 初始化发送对象.....	497



表 6-32 初始化接收对象.....	497
表 6-33 CAN 位时间参数	502
表 6-34 CAN寄存器表对应各个位的功能	513
表 6-35 错误代码.....	519
表 6-36 中断源	522
表 6-37 IF1 和 IF2 报文接口寄存器	525
表 6-38 报文内存中报文对象的结构.....	539



1 概述

NuMicro™ NUC131 系列是内嵌 ARM® Cortex™-M0 内核的 32 位微控制器，最高可运行到 50MHz，内建 36K/68K 字节的 Flash，8K 字节的 SRAM 以及用来存储升级代码的 4K 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，UART，SPI，I2C，PWM，GPIO，LIN 总线，CAN 总线，800KSPS 高速的 12 位 ADC，低压复位，掉电侦测等。

2 特性

- ARM® Cortex™-M0 内核
 - 最高可运行到 50 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
 - 36K/68K 字节 flash 存储器用来存储程序代码
 - 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)升级代码更新
 - 支持 512 字节页擦除
 - 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 字节内嵌 SRAM
- 时钟控制
 - 针对不同应用可灵活选择时钟
 - 内置 22.1184 MHz 高速振荡器可用于系统运行
 - 精度范围 $\pm 1\%$ ($+25^\circ\text{C}$, $V_{DD} = 5\text{ V}$)
 - 精度范围 $\pm 3\%$ ($-40^\circ\text{C} \sim +105^\circ\text{C}$, $V_{DD} = 2.5\text{ V} \sim 5.5\text{ V}$)
 - 内置 10 kHz 低速振荡器用于看门狗及掉电唤醒等功能
 - 支持一组高至 200MHz 的 PLL 输出，BPWM/PWM 时钟频率高至 100MHz，系统操作频率高至 50MHz
 - 外部 4~24 MHz 高速晶振用于精准的时序操作
- 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
 - 系统时钟(HCLK)
 - 内部 10 kHz 振荡器 (LIRC)
 - 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)
 - 可用作掉电模式或空闲模式的唤醒
 - 看门狗溢出事件可以触发中断或者复位芯片



- 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
- BPWM/捕捉
 - 支持时钟频率最高达 100MHz
 - 支持两组 BPWM，每组都有一个 16 位计数器和 6 个输出通道
 - 支持 BPWM 输出/捕捉输入独立模式
 - 支持 12 位从 1 到 4096 的预分频
 - 支持 BPWM 计数器 16 位分辨率
 - 向上，向下和上下计数操作类型
 - 每个 BPWM 管脚支持掩码功能和三态使能
 - 支持下列事件中断：
 - BPWM 计数器的值为 0，周期值或者比较的值
 - 支持下列事件触发 ADC 转换：
 - BPWM 计数器的值为 0，周期值或者比较值。
 - 支持 12 个 16 位分辨率的捕捉输入信道
 - 支持上升沿，下降沿和双边沿的捕捉条件
 - 支持上升沿，下降沿和双边沿输入捕捉中断
 - 支持上升沿，下降沿和双边沿捕捉计数器加载选项
- PWM/捕捉
 - 支持时钟频率最高达 100MHz
 - 支持两组 PWM，每组都有 3 个 16 位计数器和 6 个输出通道
 - 支持 PWM 输出/捕捉输入独立模式
 - 支持 3 对互补模式输出信道
 - 12 位分辨率的死区插入
 - 每个周期有两个比较值
 - 支持 12 位从 1 到 4096 的预分频
 - 支持 PWM 计数器 16 位分辨率
 - 向上，向下和上下计数操作类型
 - 每个 PWM 管脚支持掩码功能和三态使能
 - 支持刹车功能
 - 刹车源来自管脚和系统的安全事件(时钟失败，欠压检测和 CPU 锁住)
 - 刹车源管脚有噪声滤波器
 - 通过边沿侦测刹车源来控制刹车状态直到刹车中断清除
 - 在刹车条件解除后通过电平侦测刹车源来自动恢复功能
 - 支持下列事件中断：
 - PWM 计数器的值为 0，周期值或者比较的值
 - 发生刹车条件
 - 支持下列事件触发 ADC 转换：
 - PWM 计数器的值为 0，周期值或者比较值。
 - 支持 16 个 16 位分辨率的捕捉输入信道
 - 支持上升沿，下降沿和双边沿的捕捉条件
 - 支持上升沿，下降沿和双边沿输入捕捉中断
 - 支持上升沿，下降沿和双边沿捕捉计数器加载选项
- UART
 - 最多 6 个串口控制器
 - UART0 和 UART1 端口带流控功能(TXD, RXD, nCTS 和 nRTS)
 - UART0, UART1 和 UART2 带 16 字节的缓存
 - 支持 IrDA(SIR)和 LIN 功能
 - 支持 RS-485 9 位模式和方向控制



- 支持自动波特率发生器
- SPI
 - 1路 SPI 控制器
 - 支持 SPI 主/从机模式
 - 全双工同步串行数据传输
 - 传输数据长度可为 8 到 32 位
 - 首位数据传输可为 MSB 或 LSB
 - 在时钟上升沿或者下降沿收发数据可独立配置
 - 支持 32 位传输模式下字节睡眠
 - 支持三线，无从机片选信号，双向接口
- I²C
 - 最多 2 组 I²C 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
- CAN 2.0
 - 支持一组 Can 设备
 - 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
 - 支持掉电唤醒系统功能
- ADC
 - 12 位 SAR ADC 快至 800KSPS
 - 多至 8 信道单端输入或 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转换结果寄存器
 - 只对使能的通道扫描
 - 阈电压检测
 - 软件编程或外部输入可以触发 ADC 开始转换
- 96 位唯一 ID(UID)
- 128 位唯一客户 ID (UCID)
- 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°C ~ 105°C
- 封装:
 - 无铅封装(RoHS)



- LQFP 64-pin / 48-pin



3 缩写表

缩写	描述
ADC	Analog-to-Digital Converter
APB	Advanced Peripheral Bus
AHB	Advanced High-Performance Bus
BOD	Brown-out Detection
BPWM	Basic Pulse Width Modulation
CAN	Controller Area Network
DAP	Debug Access Port
FIFO	First In, First Out
FMC	Flash Memory Controller
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
PLL	Phase-Locked Loop
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
SPS	Samples per Second
TMR	Timer Controller
UART	Universal Asynchronous Receiver/Transmitter
UCID	Unique Customer ID
WDT	Watchdog Timer
WWDT	Window Watchdog Timer

表 3-1 缩写表

4 编号信息列表与管脚定义

4.1 NuMicro™ NUC131 系列选型编码

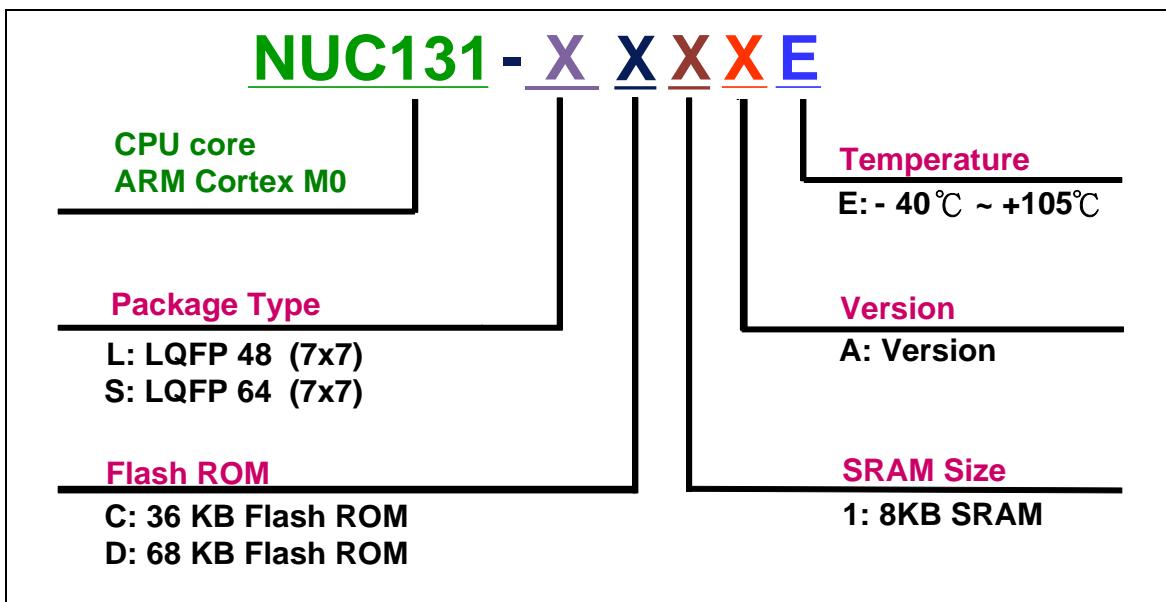


图 4.1-1 NuMicro™ NUC131 系列选型编码



4.2 NuMicro™ NUC131 系列选型指南

Part Number	APROM (KB)	RAM (KB)	Data Flash (KB)	Connectivity												Package
				ISP ROM (KB)	I/O	Timer (32-bit)	UART	SPI	I ² C	LIN	CAN	PWM (16-bit)	ADC (12-bit)	ISP/ICP/IAP		
NUC131LC2AE	36	8	Configurable	4	42	4	6	1	2	3	1	24	8 ch	✓	LQFP48	
NUC131LD2AE	68	8	Configurable	4	42	4	6	1	2	3	1	24	8 ch	✓	LQFP48	
NUC131SC2AE	36	8	Configurable	4	56	4	6	1	2	3	1	24	8 ch	✓	LQFP64	
NUC131SD2AE	68	8	Configurable	4	56	4	6	1	2	3	1	24	8 ch	✓	LQFP64	

4.3 管脚配置

4.3.1 NuMicro™ NUC131 管脚图

4.3.1.1 NuMicro™ NUC131SxxAE LQFP 64 管脚 (7 mm * 7mm)

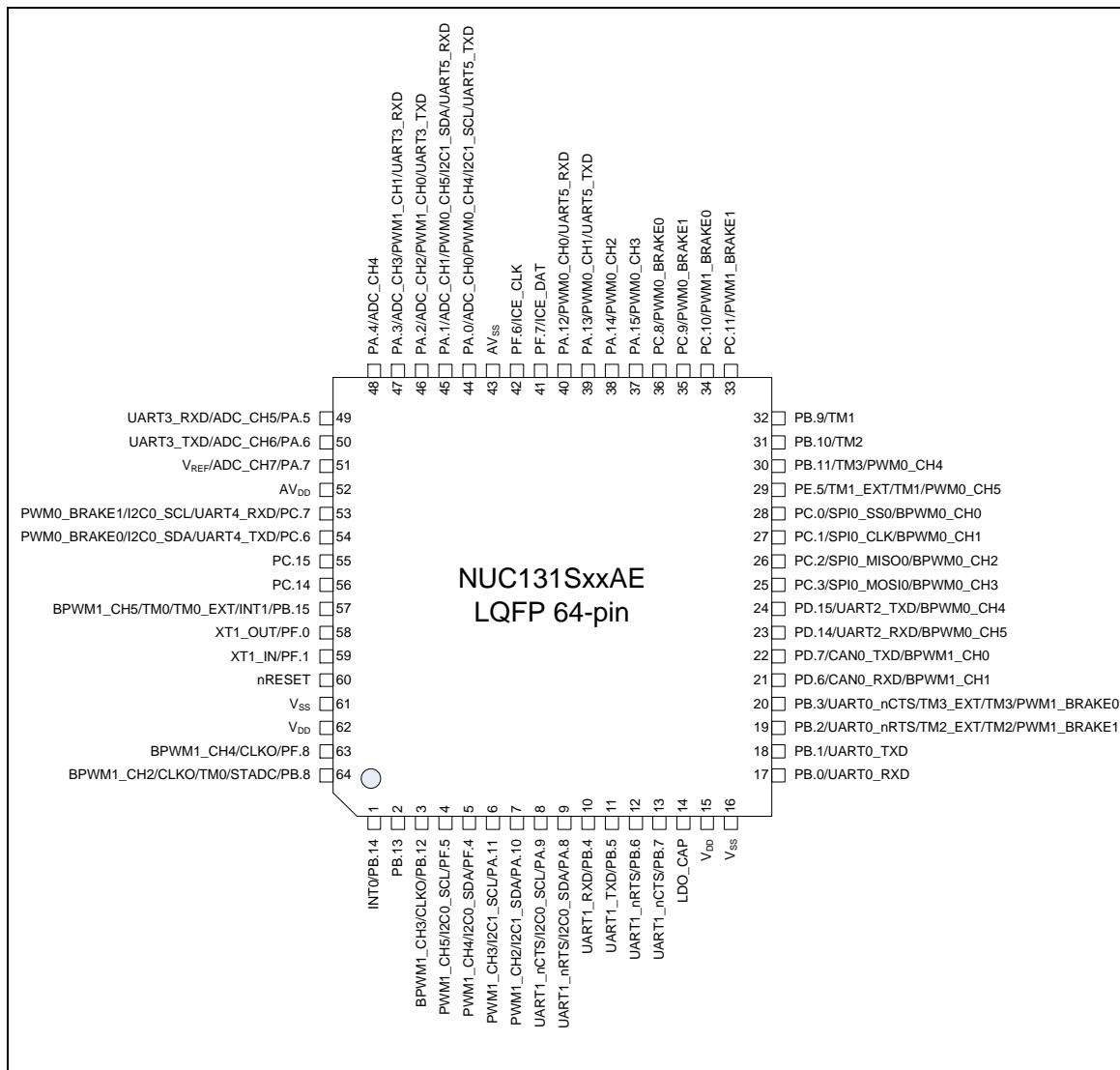


图 4-2 NuMicro™ NUC131SxxAE LQFP 64-pin 管脚图

4.3.1.2 NuMicro™ NUC131LxxAE LQFP 48 管脚

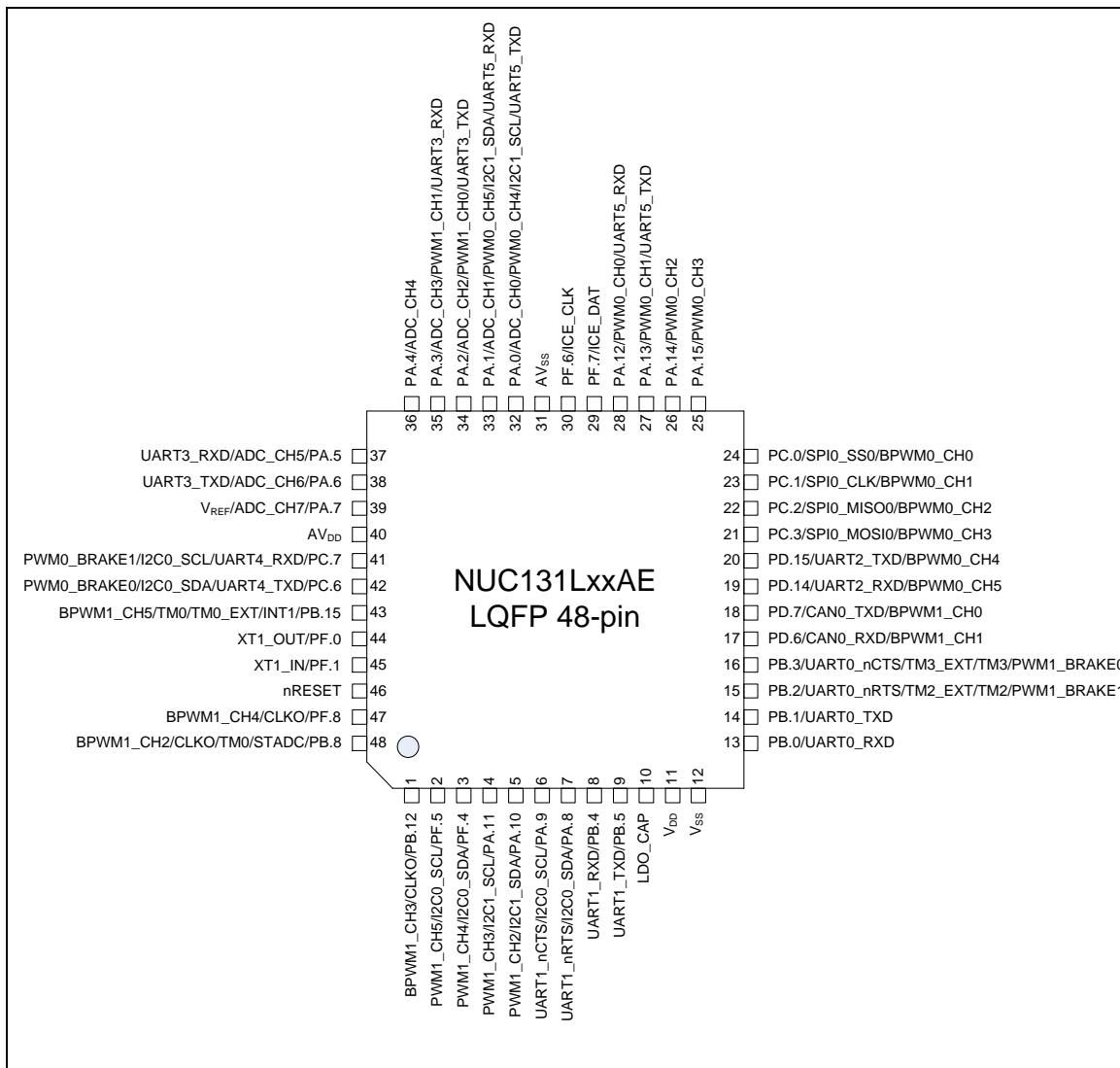


图 4-3 NuMicro™ NUC131LxxAE LQFP 48-pin 管脚图



4.4 管脚描述

4.4.1 NuMicro™ NUC131 管脚描述

管脚号.		管脚名称	管脚类型	管脚描述
LQFP 64-pin	LQFP 48-pin			
1		PB.14	I/O	通用数字输入/输出管脚
		INT0	I	外部中断0输入管脚
2		PB.13	I/O	通用数字输入/输出管脚
3	1	PB.12	I/O	通用数字输入/输出管脚
		CLKO	O	时钟频率分频输出管脚
		BPWM1_CH3	I/O	BPWM1 CH3输出/捕捉输入管脚
4	2	PF.5	I/O	通用数字输入/输出管脚
		I2C0_SCL	I/O	I ² C0 时钟管脚
		PWM1_CH5	I/O	PWM1 CH5输出/捕捉输入管脚
5	3	PF.4	I/O	通用数字输入/输出管脚
		I2C0_SDA	I/O	I ² C0 数据输入/输出管脚
		PWM1_CH4	I/O	PWM1 CH4输出/捕捉输入管脚
6	4	PA.11	I/O	通用数字输入/输出管脚
		I2C1_SCL	I/O	I ² C1 时钟管脚
		PWM1_CH3	I/O	PWM1 CH3输出/捕捉输入管脚
7	5	PA.10	I/O	通用数字输入/输出管脚
		I2C1_SDA	I/O	I ² C1 数据输入/输出管脚
		PWM1_CH2	I/O	PWM1 CH2输出/捕捉输入管脚
8	6	PA.9	I/O	通用数字输入/输出管脚
		I2C0_SCL	I/O	I ² C0 时钟管脚
		UART1_nCTS	I	UART1清零发送输入管脚
9	7	PA.8	I/O	通用数字输入/输出管脚
		I2C0_SDA	I/O	I ² C0 数据输入/输出管脚
		UART1_nRTS	O	UART1请求发送输出管脚
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
11	9	PB.5	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 64-pin	LQFP 48-pin			
		UART1_TXD	O	UART1数据发送输出管脚
12		PB.6	I/O	通用数字输入/输出管脚
		UART1_nRTS	O	UART1请求发送输出管脚
13		PB.7	I/O	通用数字输入/输出管脚
		UART1_nCTS	I	UART1清零发送输入管脚
14	10	LDO_CAP	P	LDO 输出管脚.
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
16	12	V _{SS}	P	数字电路地
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
18	14	PB.1	I/O	通用数字输入/输出管脚
		UART0_TXD	O	UART0数据发送输出管脚
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
		PWM1_BRAKE1	I	PWM1 刹车输入管脚
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
		PWM1_BRAKE0	I	PWM1刹车输入管脚
21	17	PD.6	I/O	通用数字输入/输出管脚
		CAN0_RXD	I	CAN0数据接收输入管脚
		BPWM1_CH1	I/O	BPWM1 CH1输出/捕捉输入管脚
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
		BPWM1_CH0	I/O	BPWM1 CH0输出/捕捉输入管脚
23	19	PD.14	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 64-pin	LQFP 48-pin			
		UART2_RXD	I	UART2数据接收器输入管脚
		BPWM0_CH5	I/O	BPWM0 CH5 输出/捕捉输入管脚
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
		BPWM0_CH4	I/O	BPWM0 CH4 输出/捕捉输入管脚
25	21	PC.3	I/O	通用数字输入/输出管脚
		SPI0_MOSI0	I/O	SPI0 MOSI (主出,从入) 管脚
		BPWM0_CH3	O	BPWM0 CH3 输出/捕捉输入管脚.
26	22	PC.2	I/O	通用数字输入/输出管脚
		SPI0_MISO0	I/O	SPI0 MISO (主入,从出) 管脚
		BPWM0_CH2	I	BPWM0 CH2 输出/捕捉输入管脚
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
		BPWM0_CH1	I/O	BPWM0 CH1 输出/捕捉输入管脚
28	24	PC.0	I/O	通用数字输入/输出管脚
		SPI0_SS0	I/O	SPI0从选择管脚
		BPWM0_CH0	I/O	BPWM0 CH0输出/捕捉输入管脚
29		PE.5	I/O	通用数字输入/输出管脚
		PWM0_CH5	I/O	PWM0 CH5 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
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
		PWM0_CH4	I/O	PWM0 CH4 输出/捕捉输入管脚
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
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
33		PC.11	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 64-pin	LQFP 48-pin			
		PWM1_BRAKE1	I	PWM1 刹车输入管脚
34		PC.10	I/O	通用数字输入/输出管脚
		PWM1_BRAKE0	I	PWM1 刹车输入管脚
35		PC.9	I/O	通用数字输入/输出管脚
		PWM0_BRAKE1	I	PWM0 刹车输入管脚
36		PC.8	I/O	通用数字输入/输出管脚
		PWM0_BRAKE0	I	PWM0 刹车输入管脚
37	25	PA.15	I/O	通用数字输入/输出管脚
		PWM0_CH3	I/O	PWM0 CH3输出/捕捉输入管脚
38	26	PA.14	I/O	通用数字输入/输出管脚
		PWM0_CH2	I/O	PWM0 CH2输出/捕捉输入管脚
39	27	PA.13	I/O	通用数字输入/输出管脚
		PWM0_CH1	I/O	PWM0 CH1输出/捕捉输入管脚
		UART5_TXD	O	UART5数据发送器输出管脚
40	28	PA.12	I/O	通用数字输入/输出管脚
		PWM0_CH0	I/O	PWM0 CH0 输出/捕捉输入管脚
		UART5_RXD	I	UART5数据接收器输入管脚
41	29	PF.7	I/O	通用数字输入/输出管脚
		ICE_DAT	I/O	SWD调试接口数据管脚
42	30	PF.6	I/O	通用数字输入/输出管脚
		ICE_CLK	I	SWD调试接口时钟管脚
43	31	AV _{ss}	AP	模拟电路地管脚
44	32	PA.0	I/O	通用数字输入/输出管脚
		ADC_CH0	AI	ADC_CH0模拟输入管脚
		PWM0_CH4	I/O	PWM0 CH4输出/捕捉输入管脚
		I2C1_SCL	I/O	I ² C1时钟管脚
		UART5_TXD	O	UART5数据发送器输出管脚
45	33	PA.1	I/O	通用数字输入/输出管脚
		ADC_CH1	AI	ADC_CH1模拟输入管脚
		PWM0_CH5	I/O	PWM0 CH5输出/捕捉输入管脚



管脚号.		管脚名称	管脚类型	管脚描述
LQFP 64-pin	LQFP 48-pin			
		I2C1_SDA	I/O	I ² C1数据输入/输出管脚
		UART5_RXD	I	UART5数据接收器输入管脚
46	34	PA.2	I/O	通用数字输入/输出管脚
		ADC_CH2	AI	ADC_CH2模拟输入管脚
		PWM1_CH0	I/O	PWM1 CH0输出/捕捉输入管脚
		UART3_TXD	O	UART3数据发送器输出管脚
47	35	PA.3	I/O	通用数字输入/输出管脚
		ADC_CH3	AI	ADC_CH3模拟输入管脚
		PWM1_CH1	I/O	PWM1 CH1输出/捕捉输入管脚
		UART3_RXD	I	UART3数据接收器输入管脚
48	36	PA.4	I/O	通用数字输入/输出管脚
		ADC_CH4	AI	ADC_CH4模拟输入管脚
49	37	PA.5	I/O	通用数字输入/输出管脚
		ADC_CH5	AI	ADC_CH5模拟输入管脚
		UART3_RXD	I	UART3数据接收器输入管脚
50	38	PA.6	I/O	通用数字输入/输出管脚
		ADC_CH6	AI	ADC_CH6模拟输入管脚
		UART3_TXD	O	UART3数据发送器输出管脚
51	39	PA.7	I/O	通用数字输入/输出管脚
		ADC_CH7	AI	ADC_CH7模拟输入管脚
		V _{REF}	AP	ADC 参考电压输入管脚
52	40	A _{VDD}	AP	内部模拟电路电源输入管脚
53	41	PC.7	I/O	通用数字输入/输出管脚
		UART4_RXD	I	UART4数据接收器输入管脚
		I2C0_SCL	I/O	I ² C0 时钟管脚
		PWM0_BRAKE1	I	PWM0 刹车输入管脚
54	42	PC.6	I/O	通用数字输入/输出管脚
		UART4_TXD	O	UART4数据发送器输出管脚
		I2C0_SDA	I/O	I ² C0 数据输入/输出管脚
		PWM0_BRAKE0	I	PWM0 刹车输入管脚



管脚号.		管脚名称	管脚类型	管脚描述
LQFP 64-pin	LQFP 48-pin			
55		PC.15	I/O	通用数字输入/输出管脚
56		PC.14	I/O	通用数字输入/输出管脚
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	Timer0外部捕捉输入管脚
		TM0	O	Timer0 toggle输出管脚
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
		BPWM1_CH4	I/O	BPWM1 CH4输出/捕捉输入管脚
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
		BPWM1_CH5	I/O	BPWM1 CH5输出/捕捉输入管脚
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
61		V _{SS}	P	数字电路地管脚
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
63	47	PF.8	I/O	通用数字输入/输出管脚
		CLKO	O	时钟频率分频输出管脚
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/输出管脚
		CLKO	O	时钟频率分频输出管脚
		BPWM1_CH2	I/O	BPWM1 CH2输出/捕捉输入管脚

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

5 方块图

5.1 NuMicro™ NUC131 方块图

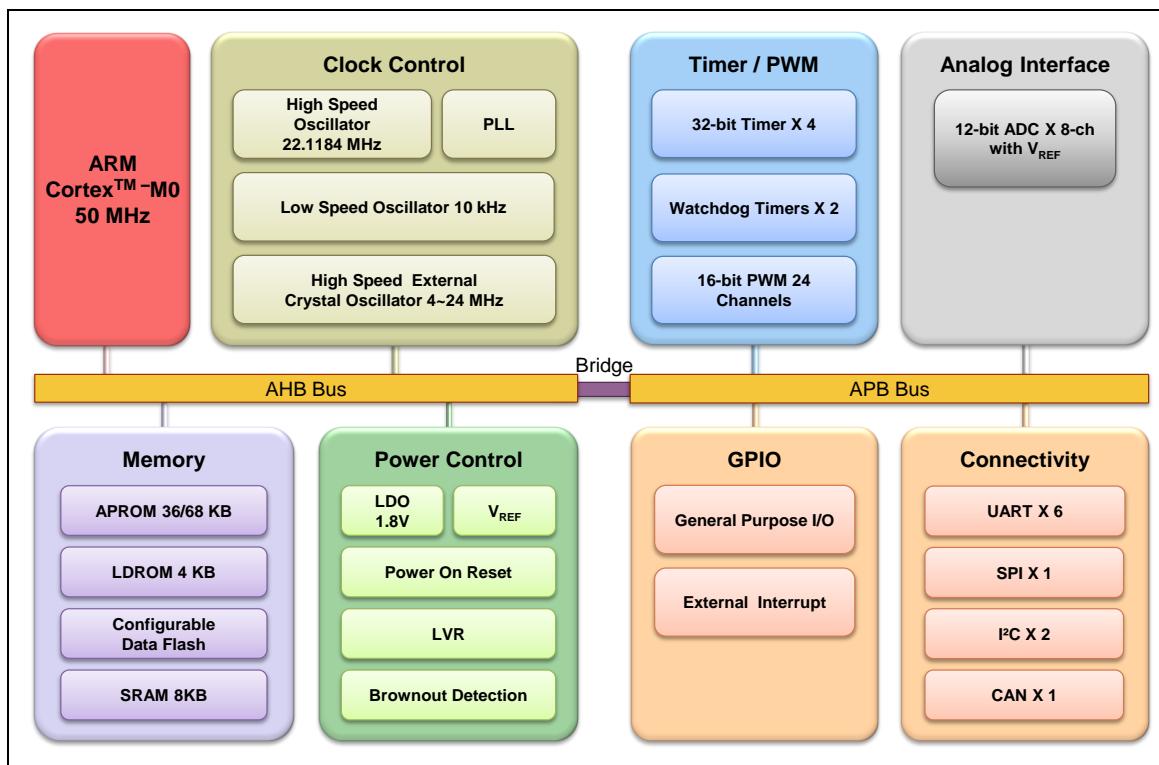


图 5-1 NuMicro™ NUC131 方块图

6 功能描述

6.1 ARM® Cortex™-M0 内核

The 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 6-1 为处理器的功能图

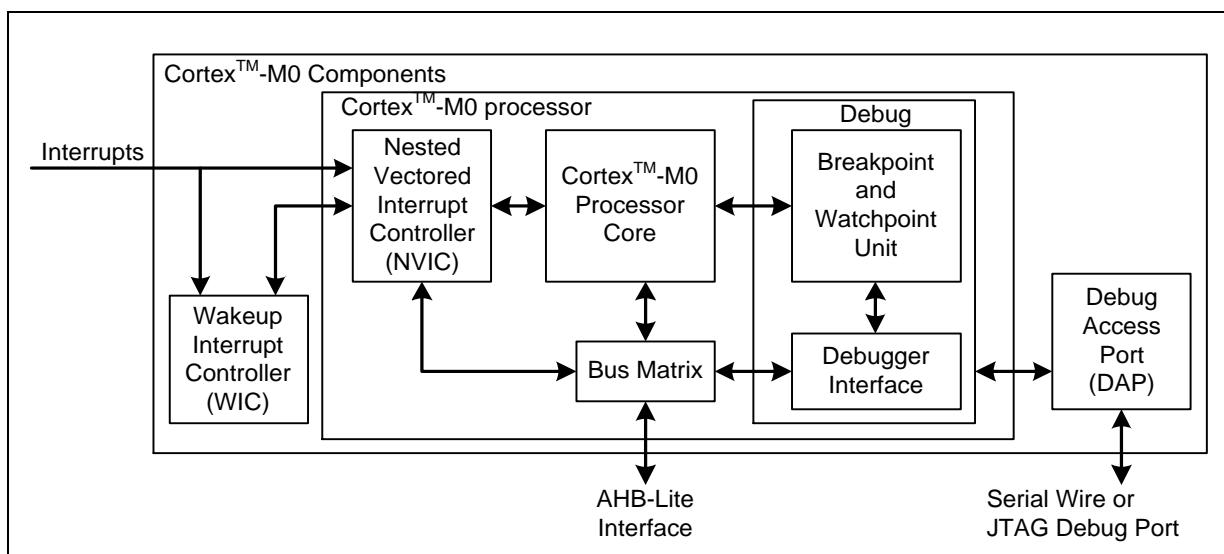


图 6-1 功能控制器框图

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



6.2 系统管理器

6.2.1 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

6.2.2 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。

6.2.3 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

内部的电压调节器LDO 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™ NUC131的电源分布。

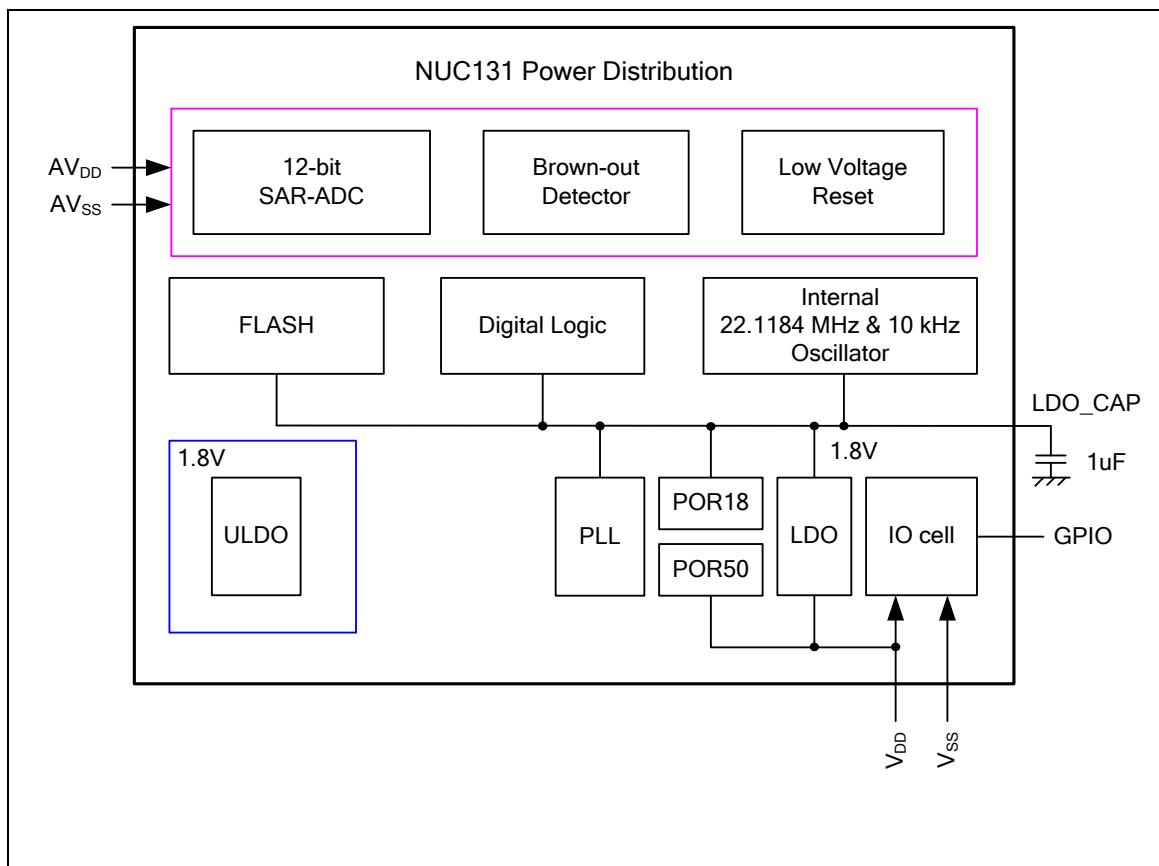


图 6-2 NuMicro™ NUC131 电源分布框图



6.2.4 系统内存映射

NuMicro™ NUC131 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™ NUC131 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_0FFF	FLASH_BA	FLASH 存储空间 (68 KB)
0x2000_0000 – 0x2000_3FFF	SRAM_BA	SRAM 存储空间 (8 KB)
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_C000 – 0x5000_FFFF	FMC_BA	Flash 内存控制寄存器
APB1 Controllers Space (0x4000_0000 ~ 0x400F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	看门狗控制寄存器
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 控制寄存器
0x4002_0000 – 0x4002_3FFF	I2C0_BA	I ² C0 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
0x4004_0000 – 0x4004_3FFF	PWM0_BA	PWM0 控制寄存器
0x4004_4000 – 0x4004_7FFF	BPWM0_BA	BPWM0 控制寄存器
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0 控制寄存器
0x4005_4000 – 0x4005_7FFF	UART3_BA	UART3 控制寄存器
0x4005_8000 – 0x4005_BFFF	UART4_BA	UART4 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)		
0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3 控制寄存器
0x4012_0000 – 0x4012_3FFF	I2C1_BA	I ² C1 接口控制寄存器
0x4014_0000 – 0x4014_3FFF	PWM1_BA	PWM1 控制寄存器
0x4014_4000 – 0x4014_7FFF	BPWM1_BA	BPWM1 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
0x4015_8000 – 0x4015_BFFF	UART5_BA	UART5 控制寄存器
0x4018_0000 – 0x4018_3FFF	CAN0_BA	CAN0 总线控制寄存器
System Controllers Space (0xE000_E000 ~ 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SCS_BA	系统定时器控制寄存器s
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 6-1 片上控制器地址空间分配



6.2.5 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_038X
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_XXXX
VREFCR	GCR_BA+0x28	R/W	参考电压控制寄存器	0x0000_0010
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_00CX
ALT_MFP	GCR_BA+0x50	R/W	复用多功能引脚控制寄存器	0x0000_0000
ALT_MFP2	GCR_BA+0x5C	R/W	复用多功能引脚控制寄存器2	0x0000_0000
ALT_MFP3	GCR_BA+0x60	R/W	复用多功能引脚控制寄存器3	0x0000_0000
ALT_MFP4	GCR_BA+0x64	R/W	复用多功能引脚控制寄存器4	0x0000_0000
REGWRPROT	GCR_BA+0x100	R/W	寄存器写保护寄存器	0x0000_0000

注意: [1] 依据每个产品型号而定

6.2.6 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							
23	22	21	20	19	18	17	16
PDID							
15	14	13	12	11	10	9	8
PDID							
7	6	5	4	3	2	1	0
PDID							

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						CPU_RST	CHIP_RST

位	描述	
[31:2]	Reserved	保留
[1]	CPU_RST	<p>CPU内核复位 (写保护)</p> <p>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。</p> <p>0= CPU 正常工作 1= CPU 复位</p> <p>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)</p>
[0]	CHIP_RST	<p>CHIP 复位(写保护)</p> <p>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。</p> <p>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。</p> <p>CHIP_RST和SYSRESETREQ的区别, 请参考章节6.2.2</p> <p>0= CHIP正常工作 1= CHIP复位</p> <p>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)</p>



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			ADC_RST	Reserved			CAN0_RST
23	22	21	20	19	18	17	16
Reserved					UART2_RST	UART1_RST	UART0_RST
15	14	13	12	11	10	9	8
Reserved			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	
[31:29]	Reserved	保留
[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
[27:25]	Reserved	保留.
[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
[23:19]	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
[15:13]	Reserved	保留.



[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	I²C1 控制器复位 0 = I ² C1 控制器正常工作 1 = I ² C1 控制器复位
[8]	I2C0_RST	I²C0 控制器复位 0 = I ² C0 控制器正常工作 1 = I ² C0 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			BPWM1_RST	BPWM0_RST	PWM1_RST	PWM0_RST	
15	14	13	12	11	10	9	8
Reserved					UART5_RST	UART4_RST	UART3_RST
7	6	5	4	3	2	1	0
Reserved							

位	描述
[31:20]	Reserved 保留.
[19]	BPWM1_RST BPWM1 控制器复位 0 = BPWM1 控制器正常工作 1 = BPWM1 控制器复位
[18]	BPWM0_RST BPWM0 控制器复位 0 = BPWM0 控制器正常工作 1 = BPWM0 控制器复位
[17]	PWM1_RST PWM1 控制器复位 0 = PWM1 控制器正常工作 1 = PWM1 控制器复位
[16]	PWM0_RST PWM0 控制器复位 0 = PWM0 控制器正常工作 1 = PWM0 控制器复位
[15:11]	Reserved 保留.
[10]	UART5_RST UART5 控制器复位 0 = UART5 控制器正常工作 1 = UART5 控制器复位
[9]	UART4_RST UART4 控制器复位 0 = UART4 控制器正常工作

		1 = UART4 控制器复位
[8]	UART3_RST	UART3 控制器复位 0 = UART3 控制器正常工作 1 = UART3 控制器复位
[7:0]	Reserved	保留



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_038X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	LVRDGSEL			Reserved	BODDGSEL		
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	
[31:15]	Reserved	保护.
[14:12]	LVRDGSEL	<p>低电压输出滤波时间选择(写保护)</p> <p>000 = 无滤波功能</p> <p>001 = 4 个系统时钟(HCLK).</p> <p>010 = 8 个系统时钟(HCLK).</p> <p>011 = 16 个系统时钟 (HCLK).</p> <p>100 = 32 个系统时钟 (HCLK).</p> <p>101 = 64 个系统时钟 (HCLK).</p> <p>110 = 128 个系统时钟 (HCLK).</p> <p>111 = 256 个系统时钟 (HCLK).</p> <p>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)</p>
[11]	Reserved	保留.
[10:8]	BODDGSEL	<p>欠压检测器输出滤波时间选择(写保护)</p> <p>000 = 欠压检测器输出使用内部10K时钟采样</p> <p>001 = 4 个系统时钟(HCLK).</p> <p>010 = 8 个系统时钟 (HCLK).</p> <p>011 = 16 个系统时钟(HCLK).</p> <p>100 = 32 个系统时钟 (HCLK).</p>

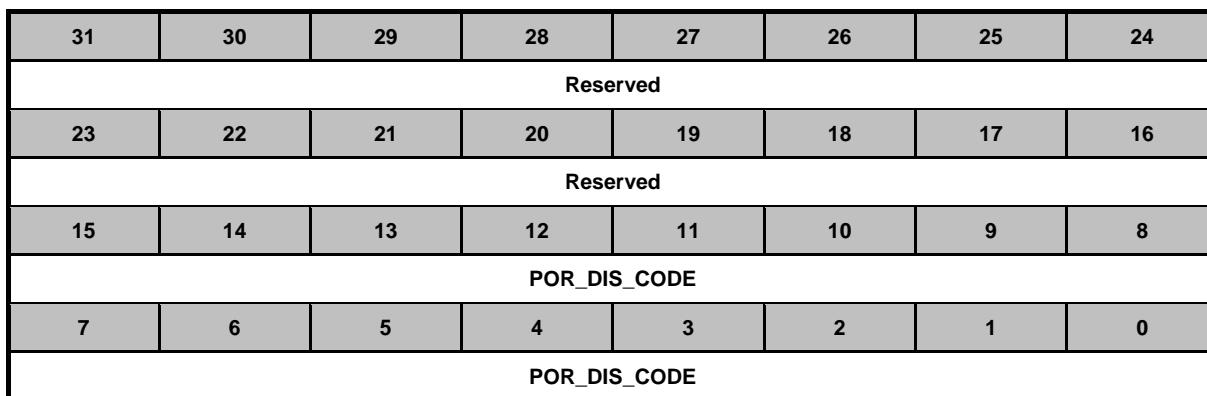
		<p>101 = 64 个系统时钟 (HCLK). 110 = 128 系统时钟 (HCLK). 111 = 256 系统时钟 (HCLK).</p> <p>注: 该位受保护, 编程该位时, 需要依次向地址0x5000_0100写入“59h”, “16h”, “88h”。该操作参考寄存器REGWRPROT(地址GCR_BA+0x100)</p>
[7]	LVR_EN	<p>低电压复位使能位 (写保护位)</p> <p>当输入电源电压低于LVR电路设置时, LVR复位芯片。低电压复位功能是默认使能的。</p> <p>0 = 禁用低电压复位功能 1 = 使能低电压复位功能, 使能该位100us后, 低电压复位输出稳定, LVR功能生效(默认)</p> <p>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)</p>
[6]	BOD_OUT	<p>欠压检测器输出状态位</p> <p>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, 该位通常响应为0。</p>
[5]	BOD_LPM	<p>欠压检测器低功耗模式 (写保护位)</p> <p>0 = BOD 工作在正常模式 (默认) 1 = BOD工作于低功耗模式</p> <p>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。</p> <p>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)</p>
[4]	BOD_INTF	<p>欠压检测中断标志</p> <p>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。</p> <p>注: 向该位写1清零。</p>
[3]	BOD_RSTEN	<p>欠压复位使能 (写保护位)</p> <p>0 = 使能欠压“中断”功能 1 = 使能欠压“复位”功能</p> <p>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。</p> <p>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。</p> <p>注2: 默认值由用户配置flash 控制寄存器CBORST(CONFIG0[20]) 时设置。</p> <p>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)</p>
[2:1]	BOD_VL	<p>欠压检测阈电压选择 (写保护位)</p> <p>默认值由用户在配置flash控制器CBOV(CONFIG0[22:21]) 时设置</p>

		<p>00 = 欠压是 2.2V. 01 = 欠压是 2.7V. 10 = 欠压是 3.7V. 11 = 欠压是 4.4V.</p> <p>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)</p>
[0]	BOD_EN	<p>欠压检测使能位 (写保护位)</p> <p>默认值由用户在配置flash控制器CBODEN(CONFIG0[23])时设置.</p> <p>0 = 禁用欠压检测功能 1 = 使能欠压检测功能</p> <p>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)</p>



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



位	描述	
[31:16]	Reserved	保留
[15:0]	POR_DIS_CODE	<p>上电复位使能位（写保护）</p> <p>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。</p> <p>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：</p> <p>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。</p> <p>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)</p>



参考电压控制寄存器(VREFCR)

寄存器	偏移地址	R/W	描述	复位值
VREFCR	GCR_BA+0x28	R/W	参考电压控制寄存器	0x0000_0010

31	30	29	28	27	26	25	24
Reserved							
23	22	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			ADC_VREFSEL	Reserved			

位	描述	
[31:5]	Reserved	保留
[4]	ADC_VREFSEL	<p>ADC 参考电压源控制 (写保护)</p> <p>0 = ADC 参考电压源来自 V_{REF} 管脚 1 = ADC 参考电压源来自 AV_{DD}</p> <p>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)</p>
[3:0]	Reserved	保留



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							
23	22	21	20	19	18	17	16
GPA_TYPE							
15	14	13	12	11	10	9	8
GPA_MFP							
7	6	5	4	3	2	1	0
GPA_MFP							

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 管脚功能选择 GPA_MFP15 位决定PA.15管脚功能 0 = 选用为GPIO功能 1 = 选用为PWM0_CH3功能
[14]	GPA_MFP14 PA.14 管脚功能选择 GPA_MFP14 位决定PA.14管脚功能 0 = 选用为GPIO功能 1 = 选用为PWM0_CH2功能
[13]	GPA_MFP13 PA.13 管脚功能选择 PA13_UR5TXD (ALT_MFP4[9])位和GPA_MFP13 位决定PA.13管脚功能 (PA13_UR5TXD, GPA_MFP13)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为PWM0_CH1功能 (1, 1) = 选用为UART5_TXD功能

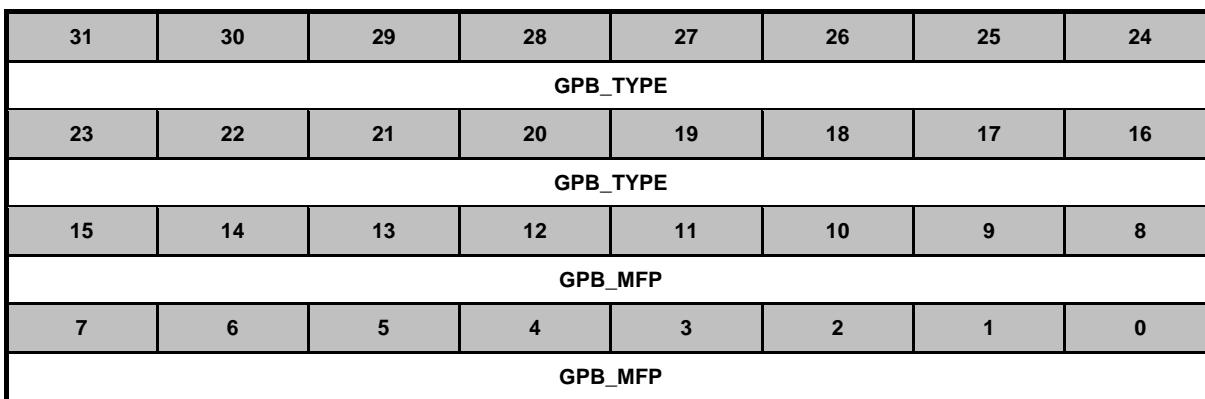
[12]	GPA_MFP12	PA.12 管脚功能选择 PA12_UR5RXD (ALT_MFP4[8])位和GPA_MFP12 位决定PA.12管脚功能 (PA12_UR5RXD, GPA_MFP12)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为PWM0_CH0功能 (1, 1) = 选用为UART5_RXD功能
[11]	GPA_MFP11	PA.11 管脚功能选择 PA11_PWM13 (ALT_MFP3[9])位和GPA_MFP11位决定PA.11管脚功能 (PA11_PWM13, GPA_MFP11)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C1_SCL功能 (1, 1) = 选用为PWM1_CH3功能
[10]	GPA_MFP10	PA.10 管脚功能选择 PA10_PWM12 (ALT_MFP3[8])位和GPA_MFP10位决定PA.10管脚功能 (PA10_PWM12, GPA_MFP10)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C1_SDA功能 (1, 1) = 选用为PWM1_CH2功能
[9]	GPA_MFP9	PA.9 管脚功能选择 PA9_UR1CTS (ALT_MFP4[1])位和GPA_MFP9位决定PA.9管脚功能 (PA9_UR1CTS, GPA_MFP9)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C0_SCL功能 (1, 1) = 选用为UART1_nCTS功能
[8]	GPA_MFP8	PA.8 管脚功能选择 PA8_UR1RTS (ALT_MFP4[0])位和GPA_MFP8位决定PA.8管脚功能 (PA8_UR1RTS, GPA_MFP8)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C0_SDA功能 (1, 1) = 选用为UART1_nRTS功能
[7]	GPA_MFP7	PA.7 管脚功能选择 PA7_VREF (ALT_MFP4[14])位和GPA_MFP7位决定PA.7管脚功能 (PA7_VREF, GPA_MFP7)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为ADC7功能 (1, 1) = 选用为Vref功能

[6]	GPA_MFP6	PA.6 管脚功能选择 PA6_UR3TXD (ALT_MFP4[5])位和GPA_MFP6位决定PA.6管脚功能 (PA6_UR3TXD, GPA_MFP6)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为ADC6功能 (1, 1) = 选用为UART3_TXD功能
[5]	GPA_MFP5	PA.5 管脚功能选择 PA5_UR3RXD (ALT_MFP4[4])位和GPA_MFP5位决定PA.5管脚功能 (PA5_UR3RXD, GPA_MFP4)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为ADC5功能 (1, 1) = 选用为UART3_RXD功能
[4]	GPA_MFP4	PA.4 管脚功能选择 GPA_MFP4 位决定PA.4管脚功能 0 = 选用为GPIO功能 1 = 选用为ADC4功能
[3]	GPA_MFP3	PA.3 管脚功能选择 PA3_PWM11 (ALT_MFP3[7])位和PA3_UR3RXD (ALT_MFP4[2])位和GPA_MFP3位决定PA.3管脚功能 (PA3_PWM11, PA3_UR3RXD, GPA_MFP3)的值和功能映像如下列所示 (0, 0, 0) = 选用为GPIO功能 (0, 0, 1) = 选用为ADC3功能 (0, 1, 1) = 选用为UART3_RXD功能 (1, 0, 1) = 选用为PWM1_CH1功能
[2]	GPA_MFP2	PA.2 管脚功能选择 PA2_PWM10 (ALT_MFP3[6])位和PA2_UR3TXD (ALT_MFP4[3])位和GPA_MFP2位决定PA.2管脚功能 (PA2_PWM10, PA2_UR3TXD, GPA_MFP2)的值和功能映像如下列所示 (0, 0, 0) = 选用为GPIO功能 (0, 0, 1) = 选用为ADC2功能 (0, 1, 1) = 选用为UART3_TXD功能 (1, 0, 1) = 选用为PWM1_CH0功能

[1]	GPA_MFP1	<p>PA.1 管脚功能选择</p> <p>PA1_PWM05 (ALT_MFP3[5])位, PA1_UR5RXD (ALT_MFP4[6])位, PA1_I2C1SDA (ALT_MFP4[13])位 和 GPA_MFP1位决定PA.1管脚功能</p> <p>(PA1_PWM05, PA1_UR5RXD, PA1_I2C1SDA, GPA_MFP1)的值和功能映像如下列所示</p> <ul style="list-style-type: none"> (0, 0, 0, 0) = 选用为GPIO功能 (0, 0, 0, 1) = 选用为ADC1功能 (0, 0, 1, 1) = 选用为I2C1_SDA功能 (0, 1, 0, 1) = 选用为UART5_RXD功能 (1, 0, 0, 1) = 选用为PWM0_CH5功能
[0]	GPA_MFP0	<p>PA.0 管脚功能选择</p> <p>PA0_PWM04 (ALT_MFP3[4])位, PA0_UR5TXD (ALT_MFP4[7])位, PA0_I2C1SCL (ALT_MFP4[12])位 和 GPA_MFP0位决定PA.0管脚功能</p> <p>(PA0_PWM04, PA0_UR5TXD, PA0_I2C1SCL, GPA_MFP0)的值和功能映像如下列所示</p> <ul style="list-style-type: none"> (0, 0, 0, 0) = 选用为GPIO功能 (0, 0, 0, 1) = 选用为ADC0功能 (0, 0, 1, 1) = 选用为I2C1_SCL功能 (0, 1, 0, 1) = 选用为UART5_TXD功能 (1, 0, 0, 1) = 选用为PWM0_CH4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



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 管脚功能选择 PB15_TOEX (ALT_MFP[24])位, PB15_TM0 (ALT_MFP2[2]) 位和GPB_MFP15位决定 PB.15管脚功能 (PB15_TOEX, PB15_TM0, GPB_MFP15)的值和功能映像如下列所示 (0, 0, 0) = 选用为GPIO功能 (0, 0, 1) = 选用为INT1功能 (0, 1, 1) = 选用为TM0功能 (1, 0, 1) = 选用为TM0_EXT功能
[14]	GPB_MFP14 PB.14 管脚功能选择 GPB_MFP14 位决定 PB.14管脚功能 0 = 选用为GPIO功能 1 = 选用为INT0功能
[13]	Reserved 保留
[12]	GPB_MFP12 PB.12 管脚功能选择 PB12_BPWM13 (ALT_MFP3[21]) 位和 GPB_MFP12位决定 PB.12管脚功能 (PB12_BPWM13, GPB_MFP12)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为CLKO功能 (1, 1) = 选用为BPWM1_CH3功能

[11]	GPB_MFP11	PB.11 管脚功能选择 PB11_PWM04 (ALT_MFP3[24]) 位和GPB_MFP11位决定PB.11管脚功能 (PB11_PWM04, GPB_MFP11)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为TM3功能 (1, 1) = 选用为PWM0_CH4功能
[10]	GPB_MFP10	PB.10 管脚功能选择 GPB_MFP10 位决定PB.10管脚功能 0 = 选用为GPIO功能 1 = 选用为TM2功能
[9]	GPB_MFP9	PB.9 管脚功能选择 GPB_MFP9 位决定PB.9管脚功能 0 = 选用为GPIO功能 1 = 选用为TM1功能
[8]	GPB_MFP8	PB.8 管脚功能选择 PB8_BPWM12 (ALT_MFP3[20])位, PB8_CLKO (ALT_MFP[29]) 位和GPB_MFP8位决定PB.8管脚功能 (PB8_BPWM12, PB8_CLKO, GPB_MFP8)的值和功能映像如下列所示 (0, 0,0) = 选用为GPIO功能 (0, 0,1) = 选用为TM0功能 (0, 1,1) = 选用为CLKO功能 (1, 0,1) = 选用为BPWM1_CH2功能
[7]	GPB_MFP7	PB.7 管脚功能选择 GPB_MFP7 位决定PB.7管脚功能 0 = 选用为GPIO功能 1 = 选用为UART1_nCTS功能
[6]	GPB_MFP6	PB.6 管脚功能选择 GPB_MFP6位决定PB.6管脚功能 0 = 选用为GPIO功能 1 = 选用为UART1_nRTS功能
[5]	GPB_MFP5	PB.5管脚功能选择 GPB_MFP5位决定PB.5管脚功能 0 = 选用为GPIO功能 1 = 选用为UART1_TxD功能
[4]	GPB_MFP4	PB.4管脚功能选择 GPB_MFP4位决定PB.4管脚功能 0 = 选用为GPIO功能



		1 = 选用为UART1_RXD功能
[3]	GPB_MFP3	<p>PB.3管脚功能选择</p> <p>PB3_TM3 (ALT_MFP2[5])位, PB3_PWM1BK0 (ALT_MFP3[30])位, PB3_T3EX (ALT_MFP[27])位和 GPB_MFP3位决定PB.3管脚功能</p> <p>(PB3_TM3, PB3_PWM1BK0, PB3_T3EX, GPB_MFP3)的值和功能映像如下列所示</p> <p>(0, 0,0,0) = 选用为GPIO功能</p> <p>(0, 0,0,1) = 选用为UART0_nCTS功能</p> <p>(0, 0,1,1) = 选用为TM3_EXT功能</p> <p>(0, 1,0,1) = 选用为PWM1_BRAKE0功能</p> <p>(1, 0,0,1) = 选用为TM3功能</p>
[2]	GPB_MFP2	<p>PB.2管脚功能选择</p> <p>PB2_TM2 (ALT_MFP2[4])位, PB2_PWM1BK1 (ALT_MFP3[31])位, PB2_T2EX (ALT_MFP[26])位和 GPB_MFP2位决定PB.2管脚功能</p> <p>(PB2_TM2, PB2_PWM1BK1, PB2_T2EX, GPB_MFP2)的值和功能映像如下列所示</p> <p>(0, 0,0,0) = 选用为GPIO功能</p> <p>(0, 0,0,1) = 选用为UART0_nRTS功能</p> <p>(0, 0,1,1) = 选用为TM2_EXT功能</p> <p>(0, 1,0,1) = 选用为PWM1_BRAKE1功能</p> <p>(1, 0,0,1) = 选用为TM2功能</p>
[1]	GPB_MFP1	<p>PB.1管脚功能选择</p> <p>GPB_MFP1位决定PB.1管脚功能</p> <p>0 = 选用为GPIO功能</p> <p>1 = 选用为UART0_TXD功能</p>
[0]	GPB_MFP0	<p>PB.0管脚功能选择</p> <p>GPB_MFP0位决定PB.0管脚功能</p> <p>0 = 选用为GPIO功能</p> <p>1 = 选用为UART0_RXD功能</p>



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							
23	22	21	20	19	18	17	16
GPC_TYPE							
15	14	13	12	11	10	9	8
GPC_MFP							
7	6	5	4	3	2	1	0
GPC_MFP							

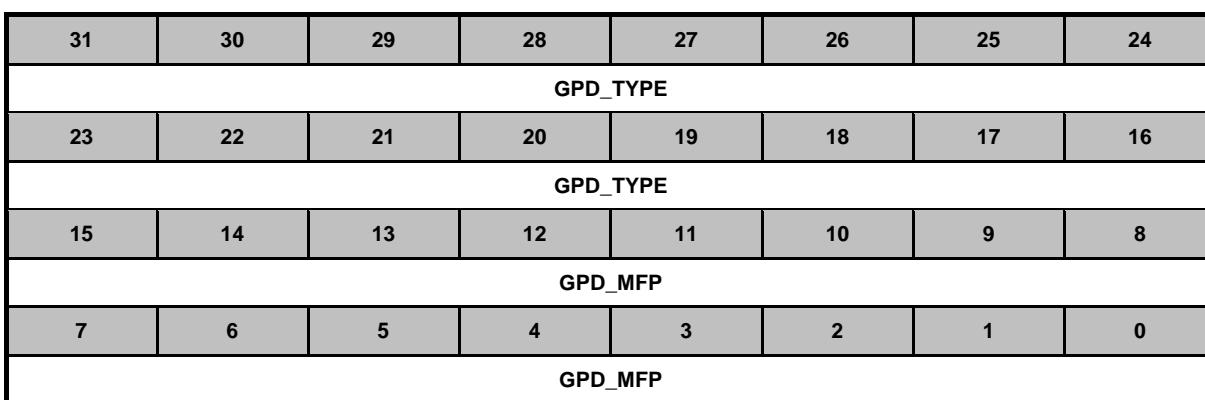
位	描述	
[31:16]	GPC_TYPEn	触发功能选择 0 = 禁用GPIOC[15:0] I/O Schmitt 触发输入 1 = 使能GPIOC[15:0] I/O Schmitt 触发输入
[15:12]	Reserved	保留.
[11]	GPC_MFP11	PC.11管脚功能选择 GPC_MFP11位决定PC.11 管脚功能 0 = 选用为GPIO功能 1 = 选用为PWM1_BRAKE1功能
[10]	GPC_MFP10	PC.10管脚功能选择 GPC_MFP10位决定PC.10 管脚功能 0 = 选用为GPIO功能 1 = 选用为PWM1_BRAKE0功能
[9]	GPC_MFP9	PC.9管脚功能选择 GPC_MFP9位决定PC.9 管脚功能 0 = 选用为GPIO功能 1 = 选用为PWM0_BRAKE1功能
[8]	GPC_MFP8	PC.8管脚功能选择 GPC_MFP8位决定PC.8 管脚功能 0 = 选用为GPIO功能 1 = 选用为PWM0_BRAKE0功能

[7]	GPC_MFP7	<p>PC.7 管脚功能选择</p> <p>PC7_PWM0BK1 (ALT_MFP3[29])位, PC7_I2C0SCL (ALT_MFP4[11])位 和 GPC_MFP7位决定PC.7管脚功能</p> <p>(PC7_PWM0BK1, PC7_I2C0SCL, GPC_MFP7)的值和功能映像如下列所示</p> <p>(0, 0, 0) = 选用为GPIO功能</p> <p>(0, 0, 1) = 选用为UART4_RXD功能</p> <p>(0, 1, 1) = 选用为I2C0_SCL功能</p> <p>(1, 0, 1) = 选用为PWM0_BRAKE1功能</p>
[6]	GPC_MFP6	<p>PC.6 管脚功能选择</p> <p>PC6_PWM0BK0 (ALT_MFP3[28])位, PC6_I2C0SDA (ALT_MFP4[10])位 和 GPC_MFP6位决定PC.6管脚功能</p> <p>(PC6_PWM0BK0, PC6_I2C0SDA, GPC_MFP6)的值和功能映像如下列所示</p> <p>(0, 0, 0) = 选用为GPIO功能</p> <p>(0, 0, 1) = 选用为UART4_TXD功能</p> <p>(0, 1, 1) = 选用为I2C0_SDA功能</p> <p>(1, 0, 1) = 选用为PWM0_BRAKE0功能</p>
[5:4]	Reserved	保留
[3]	GPC_MFP3	<p>PC.3 管脚功能选择</p> <p>PC3_BPWM03 (ALT_MFP3[15])位 和 GPC_MFP3位决定PC.3管脚功能</p> <p>(PC3_BPWM03, GPC_MFP3)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能</p> <p>(0, 1) = 选用为SPI0_MOSI0功能</p> <p>(1, 1) = 选用为BPWM0_CH3功能</p>
[2]	GPC_MFP2	<p>PC.2管脚功能选择</p> <p>PC2_BPWM02 (ALT_MFP3[14])位 和 GPC_MFP2位决定PC.2管脚功能</p> <p>(PC2_BPWM02, GPC_MFP2)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能</p> <p>(0, 1) = 选用为SPI0_MISO0功能</p> <p>(1, 1) = 选用为BPWM0_CH2功能</p>
[1]	GPC_MFP1	<p>PC.1管脚功能选择</p> <p>PC1_BPWM01 (ALT_MFP3[13])位 和 GPC_MFP1位决定PC.1管脚功能</p> <p>(PC1_BPWM01, GPC_MFP1)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能</p> <p>(0, 1) = 选用为SPI0_CLK功能</p> <p>(1, 1) = 选用为BPWM0_CH1功能</p>

[0]	GPC_MFP0	PC.0管脚功能选择 PC0_BPWM00 (ALT_MFP3[12]) 位和GPC_MFP0位决定PC.0管脚功能 (PC0_BPWM00, GPC_MFP0)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为SPI0_SS0功能 (1, 1) = 选用为BPWM0_CH0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



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管脚功能选择 PD15_BPWM04 (ALT_MFP3[16]) 位和GPD_MFP15位决定PD.15管脚功能 (PD15_BPWM04, GPD_MFP15)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为UART2_TXD功能 (1, 1) = 选用为BPWM0_CH4功能
[14]	GPD_MFP14 PD.14管脚功能选择 PD14_BPWM05 (ALT_MFP3[17]) 位和 GPD_MFP14位决定PD.14管脚功能 (PD14_BPWM05, GPD_MFP14)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为UART2_RXD功能 (1, 1) = 选用为BPWM0_CH5功能
[13:8]	Reserved
[7]	GPD_MFP7 PD.7管脚功能选择 PD7_BPWM10 (ALT_MFP3[18]) 位和 GPD_MFP7位决定PD.7管脚功能 (PD7_BPWM10, GPD_MFP7)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为CAN0_TXD功能

		(1, 1) = 选用为BPWM1_CH0功能
[6]	GPD_MFP6	PD.6管脚功能选择 PD6_BPWM11 (ALT_MFP3[19]) 位和 GPD_MFP6位决定PD.6管脚功能 (PD6_BPWM11, GPD_MFP6)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为CAN0_RXD功能 (1, 1) = 选用为BPWM1_CH1功能
[5:0]	Reserved	保留



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
Reserved							
23	22	21	20	19	18	17	16
Reserved		Reserved					
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		Reserved					

位	描述	
[31:22]	Reserved	保留.
[21]	GPE_TYPE5	<p>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</p>
[20:6]	Reserved	保留.
[5]	GPE_MFP5	<p>PE.5引脚功能选择 PE5_T1EX (ALT_MFP[25])位, PE5_TM1 (ALT_MFP2[3]) 位和 GPE_MFP5位决定PE.5管脚功能 (PE5_T1EX, PE5_TM1, GPE_MFP5) 的值和功能映像如下列所示 (0, 0, 0) = 选用为GPIO功能 (0, 0, 1) = 选用为PWMO_CH5功能 (0, 1, 1) = 选用为TM1功能 (1, 0, 1) = 选用为TM1_EXT功能</p>
[4:0]	Reserved	保留



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_00CX

注：GPF_MFP[7]/GPF_MFP[6]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							GPF_TYPE
23	22	21	20	19	18	17	16
GPF_TYPE							
15	14	13	12	11	10	9	8
Reserved							GPF_MFP
7	6	5	4	3	2	1	0
GPF_MFP							

位	描述	
[31:25]	Reserved	保留.
[24:16]	GPF_TYPEn	触发功能选择 0 = 禁用GPIOF[8:0] I/O Schmitt 触发输入 1 = 使能GPIOF [8:0] I/O Schmitt 触发输入
[15:9]	Reserved	保留
[8]	GPF_MFP8	PF.8管脚功能选择 GPF_MFP8位决定PF.8 管脚功能 0 = 选用为GPIO功能 1 = 选用为CLKO功能
[7]	GPF_MFP7	PF.7管脚功能选择 GPF_MFP7位决定PF.7 管脚功能 0 = 选用为GPIO功能 1 = 选用为ICE_DAT功能
[6]	GPF_MFP6	PF.6管脚功能选择 GPF_MFP7位决定PF.6 管脚功能 0 = 选用为GPIO功能 1 = 选用为ICE_CLK功能
[5]	GPF_MFP5	PF.5管脚功能选择 PF5_PWM15 (ALT_MFP3[11]) 位和GPF_MFP5位决定PF.5管脚功能 (PF5_PWM15, GPF_MFP5)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能

		(0, 1) = 选用为I2C0_SCL功能 (1, 1) = 选用为PWM1_CH5功能
[4]	GPF_MFP4	PF.4管脚功能选择 PF4_PWM14 (ALT_MFP3[10]) 位和GPF_MFP4位决定PF.4管脚功能 (PF4_PWM14, GPF_MFP4)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C0_SDA功能 (1, 1) = 选用为PWM1_CH4功能
[3:2]	Reserved	保留.
[1]	GPF_MFP1	PF.1管脚功能选择 PF1_BPWM15 (ALT_MFP3[23]) 位和 GPF_MFP1位决定PF.1管脚功能 (PF1_BPWM15, GPF_MFP1)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为XT1_IN功能 (1, 0) = 选用为BPWM1_CH5功能 注: 该位只读且由用户配置CGPFMFP (CONFIG0[27])位决定
[0]	GPF_MFP0	PF.0管脚功能选择 PF0_BPWM14 (ALT_MFP3[22]) 位和 GPF_MFP0位决定PF.0管脚功能 (PF0_BPWM14, GPF_MFP0)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为XT1_OUT功能 (1, 0) = 选用为BPWM1_CH4功能 注: 该位只读且由用户配置CGPFMFP (CONFIG0[27])位决定

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		PB8_CLKO	Reserved	PB3_T3EX	PB2_T2EX	PE5_T1EX	PB15_T0EX
23	22	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]	Reserved	保留.
[29]	PB8_CLKO	<p>PB.8 复用多功能管脚功能选择</p> <p>PB8_BPWM12 (ALT_MFP3[20])位, PB8_CLKO (ALT_MFP[29]) 位和GPB_MFP8位决定PB.8管脚功能 (PB8_BPWM12, PB8_CLKO, GPB_MFP8)的值和功能映像如下列所示</p> <p>(0, 0, 0) = 选用为GPIO功能 (0, 0, 1) = 选用为TM0功能 (0, 1, 1) = 选用为CLKO功能 (1, 0, 1) = 选用为BPWM1_CH2功能</p>
[28]	Reserved	保留
[27]	PB3_T3EX	<p>PB.3 复用多功能管脚功能选择</p> <p>PB3_TM3 (ALT_MFP2[5])位, PB3_PWM1BK0 (ALT_MFP3[30])位, PB3_T3EX (ALT_MFP[27]) 位和 GPB_MFP3位决定PB.3管脚功能 (PB3_TM3, PB3_PWM1BK0, PB3_T3EX, GPB_MFP3)的值和功能映像如下列所示</p> <p>(0, 0, 0, 0) = 选用为GPIO功能 (0, 0, 0, 1) = 选用为UART0_nCTS功能 (0, 0, 1, 1) = 选用为TM3_EXT功能 (0, 1, 0, 1) = 选用为PWM1_BRAKE0功能 (1, 0, 0, 1) = 选用为TM3 功能</p>

[26]	PB2_T2EX	<p>PB.2 复用多功能管脚功能选择</p> <p>PB2_TM2 (ALT_MFP2[4])位, PB2_PWM1BK1 (ALT_MFP3[31])位, PB2_T2EX (ALT_MFP[26])位和 GPB_MFP2 位决定PB.2管脚功能</p> <p>(PB2_TM2, PB2_PWM1BK1, PB2_T2EX, GPB_MFP2)的值和功能映像如下列所示</p> <ul style="list-style-type: none"> (0, 0, 0, 0) = 选用为GPIO功能 (0, 0, 0, 1) = 选用为UART0_nRTS功能 (0, 0, 1, 1) = 选用为TM2_EXT功能 (0, 1, 0, 1) = 选用为PWM1_BRAKE1功能 (1, 0, 0, 1) = 选用为TM2 功能
[25]	PE5_T1EX	<p>PE.5 复用多功能管脚功能选择</p> <p>PE5_T1EX (ALT_MFP[25])位, PE5_TM1 (ALT_MFP2[3])位和 GPE_MFP5位决定PE.5管脚功能</p> <p>(PE5_T1EX, PE5_TM1, GPE_MFP5)的值和功能映像如下列所示</p> <ul style="list-style-type: none"> (0, 0, 0) = 选用为GPIO功能 (0, 0, 1) = 选用为PWMO_CH5功能 (0, 1, 1) = 选用为TM1功能 (1, 0, 1) = 选用为TM1_EXT功能
[24]	PB15_T0EX	<p>PB.15 复用多功能管脚功能选择</p> <p>PB15_T0EX (ALT_MFP[24])位, PB15_TM0 (ALT_MFP2[2])位和 GPB_MFP15位决定PB.15管脚功能</p> <p>(PB15_T0EX, PB15_TM0, GPB_MFP15)的值和功能映像如下列所示</p> <ul style="list-style-type: none"> (0, 0, 0) = 选用为GPIO功能 (0, 0, 1) = 选用为INT1功能 (0, 1, 1) = 选用为TM0功能 (1, 0, 1) = 选用为TM0_EXT功能
[23:0]	Reserved	保留.



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	Reserved	

位	描述	
[31:6]	Reserved	保留.
[5]	PB3_TM3	<p>PB.3 复用多功能管脚功能选择</p> <p>PB3_TM3 (ALT_MFP2[5])位, PB3_PWM1BK0 (ALT_MFP3[30])位, PB3_T3EX (ALT_MFP[27])位和 GPB_MFP3位决定PB.3管脚功能 (PB3_TM3, PB3_PWM1BK0, PB3_T3EX, GPB_MFP3)的值和功能映像如下列所示</p> <p>(0, 0,0,0) = 选用为GPIO功能 (0, 0,0,1) = 选用为UART0_nCTS功能 (0, 0,1,1) = 选用为TM3_EXT功能 (0, 1,0,1) = 选用为PWM1_BRAKE0功能 (1, 0,0,1) = 选用为TM3 功能</p>
[4]	PB2_TM2	<p>PB.2 复用多功能管脚功能选择</p> <p>PB2_TM2 (ALT_MFP2[4])位, PB2_PWM1BK1 (ALT_MFP3[31])位, PB2_T2EX (ALT_MFP[26])位和 GPB_MFP2位决定PB.2管脚功能 (PB2_TM2, PB2_PWM1BK1, PB2_T2EX, GPB_MFP2)的值和功能映像如下列所示</p> <p>(0, 0,0,0) = 选用为GPIO功能 (0, 0,0,1) = 选用为UART0_nRTS功能 (0, 0,1,1) = 选用为TM2_EXT功能 (0, 1,0,1) = 选用为PWM1_BRAKE1功能 (1, 0,0,1) = 选用为TM2 功能</p>

[3]	PE5_TM1	<p>PE.5 复用多功能管脚功能选择</p> <p>PE5_T1EX (ALT_MFP[25])位, PE5_TM1 (ALT_MFP2[3]) 位和GPE_MFP5位决定PE.5管脚功能 (PE5_T1EX, PE5_TM1, GPE_MFP5)的值和功能映像如下列所示</p> <p>(0, 0,0) = 选用为GPIO功能 (0, 0,1) = 选用为PWM0_CH5功能 (0, 1,1) = 选用为TM1功能 (1, 0,1) = 选用为TM1_EXT功能</p>
[2]	PB15_TM0	<p>PB.15 复用多功能管脚功能选择</p> <p>PB15_T0EX (ALT_MFP[24])位, PB15_TM0 (ALT_MFP2[2]) 位和 GPB_MFP15位决定PB.15管脚功能 (PB15_T0EX, PB15_TM0, GPB_MFP15)的值和功能映像如下列所示</p> <p>(0, 0,0) = 选用为GPIO功能 (0, 0,1) = 选用为INT1功能 (0, 1,1) = 选用为TM0功能 (1, 0,1) = 选用为TM0_EXT功能</p>
[1:0]	Reserved	保留.



复用多功能引脚控制寄存器 3 (ALT_MFP3)

寄存器	偏移地址	R/W	描述	复位值
ALT_MFP3	GCR_BA+0x60	R/W	复用多功能引脚控制寄存器 3	0x0000_0000

31	30	29	28	27	26	25	24
PB2_PWM1BK1	PB3_PWM1BK0	PC7_PWM0BK1	PC6_PWM0BK0	Reserved			PB11_PWM04
23	22	21	20	19	18	17	16
PF1_BPWM15	PF0_BPWM14	PB12_BPWM13	PB8_BPWM12	PD6_BPWM11	PD7_BPWM10	PD14_BPWM05	PD15_BPWM04
15	14	13	12	11	10	9	8
PC3_BPWM03	PC2_BPWM02	PC1_BPWM01	PC0_BPWM00	PF5_PWM15	PF4_PWM14	PA11_PWM13	PA10_PWM12
7	6	5	4	3	2	1	0
PA3_PWM11	PA2_PWM10	PA1_PWM05	PA0_PWM04	Reserved			

位	描述
[31]	<p>PB.2 复用多功能管脚功能选择</p> <p>PB2_TM2 (ALT_MFP2[4]位, PB2_PWM1BK1 (ALT_MFP3[31])位, PB2_T2EX (ALT_MFP[26])位和 GPB_MFP2位决定PB.2管脚功能</p> <p>(PB2_TM2, PB2_PWM1BK1, PB2_T2EX, GPB_MFP2)的值和功能映像如下列所示</p> <p>(0, 0, 0, 0) = 选用为GPIO功能 (0, 0, 0, 1) = 选用为UART0_nRTS功能 (0, 0, 1, 1) = 选用为TM2_EXT功能 (0, 1, 0, 1) = 选用为PWM1_BRAKE1功能 (1, 0, 0, 1) = 选用为TM2 功能</p>
[30]	<p>PB.3 复用多功能管脚功能选择</p> <p>PB3_TM3 (ALT_MFP2[5]位, PB3_PWM1BK0 (ALT_MFP3[30])位, PB3_T3EX (ALT_MFP[27])位和 GPB_MFP3位决定PB.3管脚功能</p> <p>(PB3_TM3, PB3_PWM1BK0, PB3_T3EX, GPB_MFP3)的值和功能映像如下列所示</p> <p>(0, 0, 0, 0) = 选用为GPIO功能 (0, 0, 0, 1) = 选用为UART0_nCTS功能 (0, 0, 1, 1) = 选用为TM3_EXT功能 (0, 1, 0, 1) = 选用为PWM1_BRAKE0功能 (1, 0, 0, 1) = 选用为TM3 功能</p>

[29]	PC7_PWM0BK1	<p>PC.7 复用多功能管脚功能选择</p> <p>PC7_PWM0BK1 (ALT_MFP3[29])位, PC7_I2C0SCL (ALT_MFP4[11]) 位和 GPC_MFP7位决定PC.7管脚功能</p> <p>(PC7_PWM0BK1, PC7_I2C0SCL, GPC_MFP7)的值和功能映像如下列所示</p> <p>(0, 0, 0) = 选用为GPIO功能</p> <p>(0, 0, 1) = 选用为UART4_RXD功能</p> <p>(0, 1, 1) = 选用为I2C0_SCL功能</p> <p>(1, 0, 1) = 选用为PWM0_BRAKE1功能</p>
[28]	PC6_PWM0BK0	<p>PC.6 复用多功能管脚功能选择</p> <p>PC6_PWM0BK0 (ALT_MFP3[28])位, PC6_I2C0SDA (ALT_MFP4[10]) 位和 GPC_MFP6位决定PC.6管脚功能</p> <p>(PC6_PWM0BK0, PC6_I2C0SDA, GPC_MFP6)的值和功能映像如下列所示</p> <p>(0, 0, 0) = 选用为GPIO功能</p> <p>(0, 0, 1) = 选用为UART4_TXD功能</p> <p>(0, 1, 1) = 选用为I2C0_SDA功能</p> <p>(1, 0, 1) = 选用为PWM0_BRAKE0功能</p>
[27:25]	Reserved	保留
[24]	PB11_PWM04	<p>PB.11 复用多功能管脚功能选择</p> <p>PB11_PWM04 (ALT_MFP3[24]) 位和GPB_MFP11位决定PB.11管脚功能</p> <p>(PB11_PWM04, GPB_MFP11)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能</p> <p>(0, 1) = 选用为TM3功能</p> <p>(1, 1) = 选用为PWM0_CH4功能</p>
[23]	PF1_BPWM15	<p>PF.1 复用多功能管脚功能选择</p> <p>PF1_BPWM15 (ALT_MFP3[23]) 位和 GPF_MFP1位决定PF.1管脚功能</p> <p>(PF1_BPWM15, GPF_MFP1)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能</p> <p>(0, 1) = 选用为XT1_IN功能</p> <p>(1, 0) = 选用为BPWM1_CH5功能</p>
[22]	PF0_BPWM14	<p>PF.0 复用多功能管脚功能选择</p> <p>PF0_BPWM14 (ALT_MFP3[22]) 位和 GPF_MFP0位决定PF.0管脚功能</p> <p>(PF0_BPWM14, GPF_MFP0)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能</p> <p>(0, 1) = 选用为XT1_OUT功能</p> <p>(1, 0) = 选用为BPWM1_CH4功能</p>

[21]	PB12_BPWM13	PB.12 复用多功能管脚功能选择 PB12_BPWM13 (ALT_MFP3[21]) 位和 GPB_MFP12位决定PB.12管脚功能 (PB12_BPWM13, GPB_MFP12)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为CLKO功能 (1, 1) = 选用为BPWM1_CH3功能
[20]	PB8_BPWM12	PB.8 复用多功能管脚功能选择 PB8_BPWM12 (ALT_MFP3[20])位, PB8_CLKO (ALT_MFP[29]) 位和GPB_MFP8位决定PB.8管脚功能 (PB8_BPWM12, PB8_CLKO, GPB_MFP8)的值和功能映像如下列所示 (0, 0,0) = 选用为GPIO功能 (0, 0,1) = 选用为TM0功能 (0, 1,1) = 选用为CLKO功能 (1, 0,1) = 选用为BPWM1_CH2功能
[19]	PD6_BPWM11	PD.6 复用多功能管脚功能选择 PD6_BPWM11 (ALT_MFP3[19]) 位和 GPD_MFP6位决定PD.6管脚功能 (PD6_BPWM11, GPD_MFP6)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为CAN0_RXD功能 (1, 1) = 选用为BPWM1_CH1功能
[18]	PD7_BPWM10	PD.7 复用多功能管脚功能选择 PD7_BPWM10 (ALT_MFP3[18]) 位和 GPD_MFP7位决定PD.7管脚功能 (PD7_BPWM10, GPD_MFP7)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为CAN0_TXD功能 (1, 1) = 选用为BPWM1_CH0功能
[17]	PD14_BPWM05	PD.14 复用多功能管脚功能选择 PD14_BPWM05 (ALT_MFP3[17]) 位和 GPD_MFP14位决定PD.14管脚功能 (PD14_BPWM05, GPD_MFP14)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为UART2_RXD功能 (1, 1) = 选用为BPWM0_CH5功能
[16]	PD15_BPWM04	PD.15 复用多功能管脚功能选择 PD15_BPWM04 (ALT_MFP3[16]) 位和GPD_MFP15位决定PD.15管脚功能 (PD15_BPWM04, GPD_MFP15)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为UART2_TXD功能 (1, 1) = 选用为BPWM0_CH4功能

[15]	PC3_BPWM03	PC.3 复用多功能管脚功能选择 PC3_BPWM03 (ALT_MFP3[15]) 位和 GPC_MFP3位决定PC.3管脚功能 (PC3_BPWM03, GPC_MFP3)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为SPI0_MOSI0功能 (1, 1) = 选用为BPWM0_CH3功能
[14]	PC2_BPWM02	PC.2 复用多功能管脚功能选择 PC2_BPWM02 (ALT_MFP3[14]) 位和 GPC_MFP2位决定PC.2管脚功能 (PC2_BPWM02, GPC_MFP2)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为SPI0_MISO0功能 (1, 1) = 选用为BPWM0_CH2功能
[13]	PC1_BPWM01	PC.1 复用多功能管脚功能选择 PC1_BPWM01 (ALT_MFP3[13]) 位和 GPC_MFP1位决定PC.1管脚功能 (PC1_BPWM01, GPC_MFP1)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为SPI0_CLK功能 (1, 1) = 选用为BPWM0_CH1功能
[12]	PC0_BPWM00	PC.0 复用多功能管脚功能选择 PC0_BPWM00 (ALT_MFP3[12]) 位和 GPC_MFP0位决定PC.0管脚功能 (PC0_BPWM00, GPC_MFP0)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为SPI0_SSO功能 (1, 1) = 选用为BPWM0_CH0功能
[11]	PF5_PWM15	PF.5 复用多功能管脚功能选择 PF5_PWM15 (ALT_MFP3[11]) 位和 GPF_MFP5位决定PF.5管脚功能 (PF5_PWM15, GPF_MFP5)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C0_SCL功能 (1, 1) = 选用为PWM1_CH5功能
[10]	PF4_PWM14	PF.4 复用多功能管脚功能选择 PF4_PWM14 (ALT_MFP3[10]) 位和 GPF_MFP4位决定PF.4管脚功能 (PF4_PWM14, GPF_MFP4)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C0_SDA功能 (1, 1) = 选用为PWM1_CH4功能

[9]	PA11_PWM13	PA.11 复用多功能管脚功能选择 PA11_PWM13 (ALT_MFP3[9])位和GPA_MFP11位决定PA.11管脚功能 (PA11_PWM13, GPA_MFP11)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C1_SCL功能 (1, 1) = 选用为PWM1_CH3功能
[8]	PA10_PWM12	PA.10 复用多功能管脚功能选择 PA10_PWM12 (ALT_MFP3[8])位和GPA_MFP10位决定PA.10管脚功能 (PA10_PWM12, GPA_MFP10)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C1_SDA功能 (1, 1) = 选用为PWM1_CH2功能
[7]	PA3_PWM11	PA.3 复用多功能管脚功能选择 PA3_PWM11 (ALT_MFP3[7])位和PA3_UR3RXD (ALT_MFP4[2])位和GPA_MFP3位决定PA.3管脚功能 (PA3_PWM11, PA3_UR3RXD, GPA_MFP3)的值和功能映像如下列所示 (0, 0,0) = 选用为GPIO功能 (0, 0,1) = 选用为ADC3功能 (0, 1,1) = 选用为UART3_RXD功能 (1, 0,1) = 选用为PWM1_CH1功能
[6]	PA2_PWM10	PA.2 复用多功能管脚功能选择 PA2_PWM10 (ALT_MFP3[6])位和PA2_UR3TXD (ALT_MFP4[3])位和GPA_MFP2位决定PA.2管脚功能 (PA2_PWM10, PA2_UR3TXD, GPA_MFP2)的值和功能映像如下列所示 (0, 0,0) = 选用为GPIO功能 (0, 0,1) = 选用为ADC2功能 (0, 1,1) = 选用为UART3_TXD功能 (1, 0,1) = 选用为PWM1_CH0功能
[5]	PA1_PWM05	PA.1 复用多功能管脚功能选择 PA1_PWM05 (ALT_MFP3[5])位, PA1_UR5RXD (ALT_MFP4[6])位, PA1_I2C1SDA (ALT_MFP4[13])位 和 GPA_MFP1位决定PA.1管脚功能 (PA1_PWM05, PA1_UR5RXD, PA1_I2C1SDA, GPA_MFP1)的值和功能映像如下列所示 (0, 0,0,0) = 选用为GPIO功能 (0, 0,0,1) = 选用为ADC1功能 (0, 0,1,1) = 选用为I2C1_SDA功能 (0, 1,0,1) = 选用为UART5_RXD功能 (1, 0,0,1) = 选用为PWM0_CH5功能

[4]	PA0_PWM04	<p>PA.0 复用多功能管脚功能选择</p> <p>PA0_PWM04 (ALT_MFP3[4])位, PA0_UR5TXD (ALT_MFP4[7])位, PA0_I2C1SCL (ALT_MFP4[12])位 和 GPA_MFP0位决定PA.0管脚功能</p> <p>(PA0_PWM04, PA0_UR5TXD, PA0_I2C1SCL, GPA_MFP0)的值和功能映像如下列所示</p> <ul style="list-style-type: none"> (0, 0,0,0) = 选用为GPIO功能 (0, 0,0,1) = 选用为ADC0功能 (0, 0,1,1) = 选用为I2C1_SCL功能 (0, 1,0,1) = 选用为UART5_TXD功能 (1, 0,0,1) = 选用为PWM0_CH4功能
[3:0]	Reserved	保留.



复用多功能引脚控制寄存器4 (ALT_MFP4)

寄存器	偏移地址	R/W	描述	复位值
ALT_MFP4	GCR_BA+0x64	R/W	复用多功能引脚控制寄存器 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	PA7_VREF	PA1_I2C1SDA	PA0_I2C1SCL	PC7_I2C0SCL	PC6_I2C0SDA	PA13_UR5TxD	PA12_UR5RXD
7	6	5	4	3	2	1	0
PA0_UR5TxD	PA1_UR5RXD	PA6_UR3TxD	PA5_UR3RXD	PA2_UR3TxD	PA3_UR3RXD	PA9_UR1CTS	PA8_UR1RTS

位	描述	
[31:15]	Reserved	保留.
[14]	PA7_VREF	<p>PA.7 复用多功能管脚功能选择</p> <p>PA7_VREF (ALT_MFP4[14])位和GPA_MFP7位决定PA.7管脚功能 (PA7_VREF, GPA_MFP7)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能 (0, 1) = 选用为ADC7功能 (1, 1) = 选用为Vref功能</p>
[13]	PA1_I2C1SDA	<p>PA.1 复用多功能管脚功能选择</p> <p>PA1_PWM05 (ALT_MFP4[5])位, PA1_UR5RXD (ALT_MFP4[6])位, PA1_I2C1SDA (ALT_MFP4[13])位 和 GPA_MFP1位决定PA.1管脚功能 (PA1_PWM05, PA1_UR5RXD, PA1_I2C1SDA, GPA_MFP1)的值和功能映像如下列所示</p> <p>(0, 0, 0, 0) = 选用为GPIO功能 (0, 0, 0, 1) = 选用为ADC1功能 (0, 0, 1, 1) = 选用为I2C1_SDA功能 (0, 1, 0, 1) = 选用为UART5_RXD功能 (1, 0, 0, 1) = 选用为PWM0_CH5功能</p>

[12]	PA0_I2C1SCL	PA.0 复用多功能管脚功能选择 PA0_PWM04 (ALT_MFP3[4])位, PA0_UR5TXD (ALT_MFP4[7])位, PA0_I2C1SCL (ALT_MFP4[12])位 和 GPA_MFP0位决定PA.0管脚功能 (PA0_PWM04, PA0_UR5TXD, PA0_I2C1SCL, GPA_MFP0)的值和功能映像如下列所示 (0, 0,0,0) = 选用为GPIO功能 (0, 0,0,1) = 选用为ADC0功能 (0, 0,1,1) = 选用为I2C1_SCL功能 (0, 1,0,1) = 选用为UART5_TXD功能 (1, 0,0,1) = 选用为PWM0_CH4功能
[11]	PC7_I2C0SCL	PC.7 复用多功能管脚功能选择 PC7_PWM0BK1 (ALT_MFP3[29])位, PC7_I2C0SCL (ALT_MFP4[11])位 和 GPC_MFP7位决定PC.7管脚功能 (PC7_PWM0BK1, PC7_I2C0SCL, GPC_MFP7)的值和功能映像如下列所示 (0, 0,0) = 选用为GPIO功能 (0, 0,1) = 选用为UART4_RXD功能 (0, 1,1) = 选用为I2C0_SCL功能 (1, 0,1) = 选用为PWM0_BRAKE1功能
[10]	PC6_I2C0SDA	PC.6 复用多功能管脚功能选择 PC6_PWM0BK0 (ALT_MFP3[28])位, PC6_I2C0SDA (ALT_MFP4[10])位 和 GPC_MFP6位决定PC.6管脚功能 (PC6_PWM0BK0, PC6_I2C0SDA, GPC_MFP6)的值和功能映像如下列所示 (0, 0,0) = 选用为GPIO功能 (0, 0,1) = 选用为UART4_TXD功能 (0, 1,1) = 选用为I2C0_SDA功能 (1, 0,1) = 选用为PWM0_BRAKE0功能
[9]	PA13_UR5TXD	PA.13 复用多功能管脚功能选择 PA13_UR5TXD (ALT_MFP4[9])位 和 GPA_MFP13 位决定PA.13管脚功能 (PA13_UR5TXD, GPA_MFP13)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为PWM0_CH1功能 (1, 1) = 选用为UART5_TXD功能
[8]	PA12_UR5RXD	PA.12 复用多功能管脚功能选择 PA12_UR5RXD (ALT_MFP4[8])位 和 GPA_MFP12 位决定PA.12管脚功能 (PA12_UR5RXD, GPA_MFP12)的值和功能映像如下列所示 (0, 0) = 选用为GPIO功能 (0, 1) = 选用为PWM0_CH0功能 (1, 1) = 选用为UART5_RXD功能

[7]	PA0_UR5TXD	<p>PA.0 复用多功能管脚功能选择</p> <p>PA0_PWM04 (ALT_MFP3[4])位, PA0_UR5TXD (ALT_MFP4[7])位, PA0_I2C1SCL (ALT_MFP4[12])位 和 GPA_MFP0位决定PA.0管脚功能</p> <p>(PA0_PWM04, PA0_UR5TXD, PA0_I2C1SCL, GPA_MFP0)的值和功能映像如下列所示</p> <p>(0, 0,0,0) = 选用为GPIO功能</p> <p>(0, 0,0,1) = 选用为ADC0功能</p> <p>(0, 0,1,1) = 选用为I2C1_SCL功能</p> <p>(0, 1,0,1) = 选用为UART5_TXD功能</p> <p>(1, 0,0,1) = 选用为PWM0_CH4功能</p>
[6]	PA1_UR5RXD	<p>PA.1 复用多功能管脚功能选择</p> <p>PA1_PWM05 (ALT_MFP3[5])位, PA1_UR5RXD (ALT_MFP4[6])位, PA1_I2C1SDA (ALT_MFP4[13])位 和 GPA_MFP1位决定PA.1管脚功能</p> <p>(PA1_PWM05, PA1_UR5RXD, PA1_I2C1SDA, GPA_MFP1)的值和功能映像如下列所示</p> <p>(0, 0,0,0) = 选用为GPIO功能</p> <p>(0, 0,0,1) = 选用为ADC1功能</p> <p>(0, 0,1,1) = 选用为I2C1_SDA功能</p> <p>(0, 1,0,1) = 选用为UART5_RXD功能</p> <p>(1, 0,0,1) = 选用为PWM0_CH5功能</p>
[5]	PA6_UR3TXD	<p>PA.6 复用多功能管脚功能选择</p> <p>PA6_UR3TXD (ALT_MFP4[5])位和GPA_MFP6位决定PA.6管脚功能</p> <p>(PA6_UR3TXD, GPA_MFP6)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能</p> <p>(0, 1) = 选用为ADC6功能</p> <p>(1, 1) = 选用为UART3_TXD功能</p>
[4]	PA5_UR3RXD	<p>PA.5 复用多功能管脚功能选择</p> <p>PA5_UR3RXD (ALT_MFP4[4])位和GPA_MFP5位决定PA.5管脚功能</p> <p>(PA5_UR3RXD, GPA_MFP4)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能</p> <p>(0, 1) = 选用为ADC5功能</p> <p>(1, 1) = 选用为UART3_RXD功能</p>
[3]	PA2_UR3TXD	<p>PA.2 复用多功能管脚功能选择</p> <p>PA2_PWM10 (ALT_MFP3[6])位和PA2_UR3TXD (ALT_MFP4[3])位和GPA_MFP2位决定PA.2管脚功能</p> <p>(PA2_PWM10, PA2_UR3TXD, GPA_MFP2)的值和功能映像如下列所示</p> <p>(0, 0,0) = 选用为GPIO功能</p> <p>(0, 0,1) = 选用为ADC2功能</p> <p>(0, 1,1) = 选用为UART3_TXD功能</p> <p>(1, 0,1) = 选用为PWM1_CH0功能</p>

[2]	PA3_UR3RXD	<p>PA.3 复用多功能管脚功能选择</p> <p>PA3_PWM11 (ALT_MFP3[7])位和PA3_UR3RXD (ALT_MFP4[2])位和GPA_MFP3位决定PA.3管脚功能 (PA3_PWM11, PA3_UR3RXD, GPA_MFP3)的值和功能映像如下列所示</p> <p>(0, 0, 0) = 选用为GPIO功能 (0, 0, 1) = 选用为ADC3功能 (0, 1, 1) = 选用为UART3_RXD功能 (1, 0, 1) = 选用为PWM1_CH1功能</p>
[1]	PA9_UR1CTS	<p>PA.9 复用多功能管脚功能选择</p> <p>PA9_UR1CTS (ALT_MFP4[1])位和GPA_MFP9位决定PA.9管脚功能 (PA9_UR1CTS, GPA_MFP9)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C0_SCL功能 (1, 1) = 选用为UART1_nCTS功能</p>
[0]	PA8_UR1RTS	<p>PA.8 复用多功能管脚功能选择</p> <p>PA8_UR1RTS (ALT_MFP4[0])位和GPA_MFP8位决定PA.8管脚功能 (PA8_UR1RTS, GPA_MFP8)的值和功能映像如下列所示</p> <p>(0, 0) = 选用为GPIO功能 (0, 1) = 选用为I2C0_SDA功能 (1, 1) = 选用为UART1_nRTS功能</p>



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。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，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REGPROTDIS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							REGWRPROT / REGPROTDIS

位	描述
[31:16]	Reserved 保留.
[7:0]	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REGWRPROT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 VREFCR: 地址 0x5000_0028 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)
FATCON: 地址 0x5000_C018
WTCR: 地址 0x4000_4000
WTCRALT: 地址 0x4000_4004
PWM_CTL0: 地址 0x4004_0000, 0x4014_0000
PWM_DTCTL0_1: 地址 0x4004_0070, 0x4014_0070
PWM_DTCTL2_3: 地址 0x4004_0074, 0x4014_0074
PWM_DTCTL4_5: 地址 0x4004_0078, 0x4014_0078
PWM_BRKCTL0_1: 地址 0x4004_00C8, 0x4014_00C8
PWM_BRKCTL2_3: 地址 0x4004_00CC, 0x4014_00CC
PWM_BRKCTL4_5: 地址 0x4004_00D0, 0x4014_00D0
PWM_SWBRK: 地址 0x4004_00DC, 0x4014_00DC
PWM_INTEN1: 地址 0x4004_00E4, 0x4014_00E4
PWM_INTSTS1: 地址 0x4004_00EC, 0x4014_00EC
BPWM_CTL0: 地址 0x4004_4000, 0x4014_4000
注: 这些位在寄存器描述旁注释为“写保护”。



6.2.7 系统定时器(SysTick)

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”。

6.2.7.1 系统定时器控制器寄存器映射

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	0xFFFF_FFFF
SYST_CVR	SCS_BA+0x18	R/W	SysTick 当前值寄存器	0xFFFF_FFFF



6.2.7.2 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

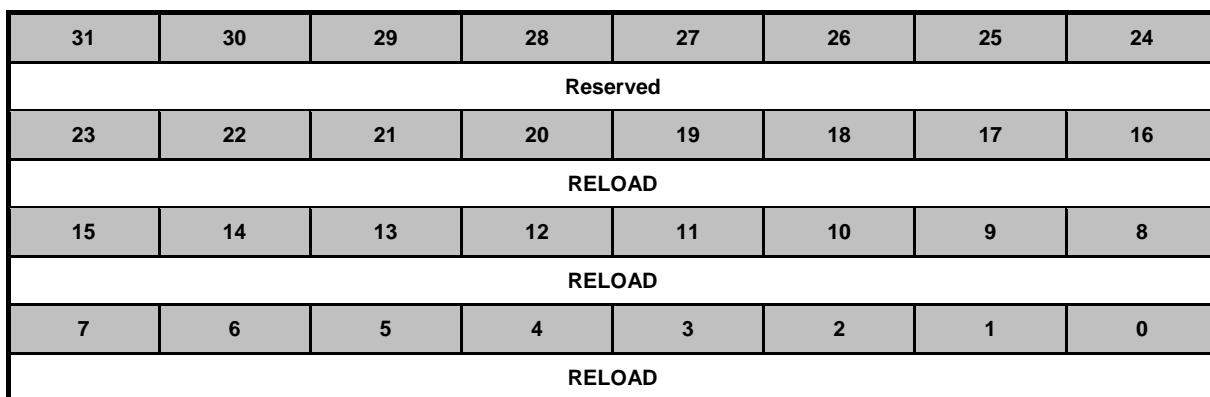
31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CLKSRC	TICKINT	ENABLE

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ICLKSR(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	描述
SYST_RVR	SCS_BA+0x14	R/W	SysTick 重新加载值寄存器	0xXXXX_XXXX

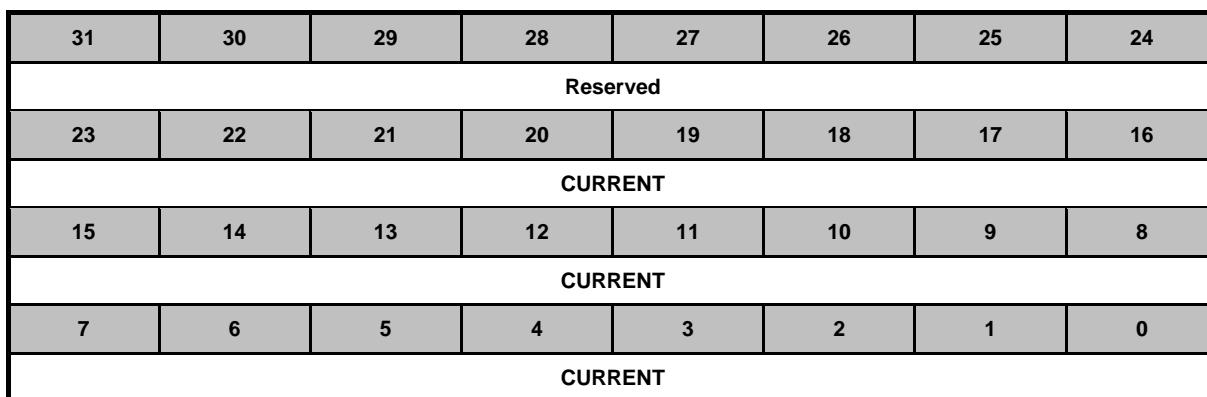


位	描述	
[31:24]	Reserved	保留.
[23:0]	RELOAD	当计数器达到 0 时, 这个值加载到当前值寄存器



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	0XXXX_XXXX



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0 (write-clear)。软件写入任何值将清寄存器为0。



6.2.8 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 支持末尾链接 “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”。



6.2.8.1 异常模式和系统中断映射

NuMicro™ NUC131系列支持如表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 6-2 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	源模块	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[8:0] 的外部信号中断
22	6	-	-	保留
23	7	-	-	保留
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	UART3_INT	UART3	UART3 中断
32	16	UART4_INT	UART4	UART4 中断
33	17	UART5_INT	UART5	UART5 中断
34	18	I2C0_INT	I ² C0	I ² C0 中断
35	19	I2C1_INT	I ² C1	I ² C1 中断
36	20	CAN0_INT	CAN0	CAN0 中断
37	21	-	-	保留
38	22	PWM0_INT	PWM0	PWM0 中断
39	23	PWM1_INT	PWM1	PWM1 中断
40	24	BPWM0_INT	BPWM0	BPWM0 中断
41	25	BPWM1_INT	BPWM1	BPWM1 中断
42	26	BRAKE0_INT	PWM0	PWM0 刹车中断
43	27	BRAKE1_INT	PWM1	PWM1 刹车中断
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	CKD_INT	CLKC	时钟检测中断
47	31	-	-	保留

表 6-3 系统中断映射



6.2.8.2 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 6-4 向量表格式

6.2.8.3 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，这些寄存器使用写1使能和写1清零机制，寄存器可以读取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。



6.2.8.4 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

6.2.8.5 NVIC 控制寄存器描述

IRQ0 ~ IRQ31 设置使能控制寄存器 (NVIC_ISER)

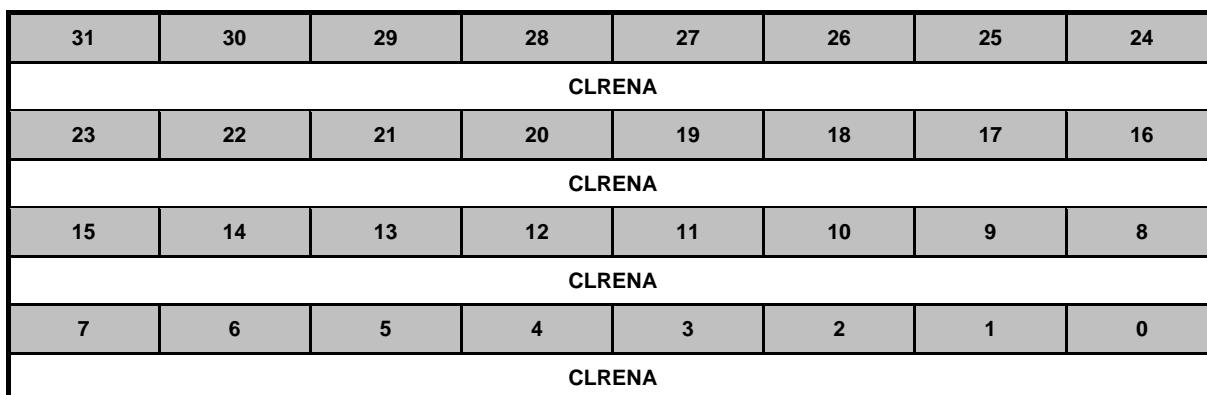
寄存器	偏移地址	R/W	描述	复位值
NVIC_ISER	SCS_BA+0x100	R/W	IRQ0 ~ IRQ31 设置使能控制寄存器	0x0000_0000

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

位	描述	
[31:0]	SETENA	<p>中断使能寄存器</p> <p>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）</p> <p>写操作：</p> <p>0 = 无效 1 = 写 1 使能相关中断</p> <p>读操作：</p> <p>0 = 相关中断状态被禁止 1 = 相关中断状态已使能</p> <p>读取该寄存器返回当前使能状态。</p>


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



位	描述
[31:0]	<p>中断禁用位</p> <p>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）</p> <p>写操作：</p> <p>0 = 无效 1 = 写1禁用相关中断</p> <p>读操作：</p> <p>0 = 相关中断状态被禁止 1 = 相关中断状态已使能</p> <p>读取该寄存器返回当前使能状态</p>

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							
23	22	21	20	19	18	17	16
SETPEND							
15	14	13	12	11	10	9	8
SETPEND							
7	6	5	4	3	2	1	0
SETPEND							

位	描述
[31:0]	<p>SETPEND</p> <p>设置中断挂起寄存器</p> <p>写操作:</p> <p>0 = 无效</p> <p>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）。</p> <p>读操作:</p> <p>0 = 相关中断不在挂起状态</p> <p>1 = 相关中断在挂起状态</p> <p>读取该寄存器返回当前等待处理的中断状态</p>

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							
23	22	21	20	19	18	17	16
CLRPEND							
15	14	13	12	11	10	9	8
CLRPEND							
7	6	5	4	3	2	1	0
CLRPEND							

位	描述
[31:0]	<p>CLRPEND</p> <p>清中断挂起寄存器</p> <p>写操作:</p> <p>0 = 无效</p> <p>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）</p> <p>读操作:</p> <p>0 = 相关寄存器不在挂起状态</p> <p>1 = 相关寄存器在挂起状态</p> <p>读取该寄存器返回当前等待处理的中断状态</p>



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		Reserved					
23	22	21	20	19	18	17	16
PRI_2		Reserved					
15	14	13	12	11	10	9	8
PRI_1		Reserved					
7	6	5	4	3	2	1	0
PRI_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		Reserved					
23	22	21	20	19	18	17	16
PRI_6		Reserved					
15	14	13	12	11	10	9	8
PRI_5		Reserved					
7	6	5	4	3	2	1	0
PRI_4		Reserved					

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		Reserved					
23	22	21	20	19	18	17	16
PRI_10		Reserved					
15	14	13	12	11	10	9	8
PRI_9		Reserved					
7	6	5	4	3	2	1	0
PRI_8		Reserved					

位	描述	
[31:30]	PRI_11	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	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	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	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		Reserved					
23	22	21	20	19	18	17	16
PRI_14		Reserved					
15	14	13	12	11	10	9	8
PRI_13		Reserved					
7	6	5	4	3	2	1	0
PRI_12		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		Reserved					
23	22	21	20	19	18	17	16
PRI_18		Reserved					
15	14	13	12	11	10	9	8
PRI_17		Reserved					
7	6	5	4	3	2	1	0
PRI_16		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		Reserved					
23	22	21	20	19	18	17	16
PRI_22		Reserved					
15	14	13	12	11	10	9	8
PRI_21		Reserved					
7	6	5	4	3	2	1	0
PRI_20		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		Reserved					
23	22	21	20	19	18	17	16
PRI_26		Reserved					
15	14	13	12	11	10	9	8
PRI_25		Reserved					
7	6	5	4	3	2	1	0
PRI_24		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		Reserved					
23	22	21	20	19	18	17	16
PRI_30		Reserved					
15	14	13	12	11	10	9	8
PRI_29		Reserved					
7	6	5	4	3	2	1	0
PRI_28		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



6.2.8.6 中断源寄存器映射

除了 NVIC 相关的中断控制寄存器外，NuMicro™ NUC131 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	0xFFFF_FFFF
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别	0xFFFF_FFFF
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 中断源识别	0xFFFF_FFFF
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别	0xFFFF_FFFF
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (GPA/B) 中断源识别	0xFFFF_FFFF
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GPC/D/E/F) 中断源识别	0xFFFF_FFFF
IRQ6_SRC	INT_BA+0x18	R	保留	0xFFFF_FFFF
IRQ7_SRC	INT_BA+0x1C	R	保留	0xFFFF_FFFF
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别	0xFFFF_FFFF
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别	0xFFFF_FFFF
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别	0xFFFF_FFFF
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别	0xFFFF_FFFF
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (UART0/2) 中断源识别	0xFFFF_FFFF
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (UART1) 中断源识别	0xFFFF_FFFF
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别	0xFFFF_FFFF
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (UART3) 中断源识别	0xFFFF_FFFF
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (UART4) 中断源识别	0xFFFF_FFFF
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (UART5) 中断源识别	0xFFFF_FFFF
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I ² C0) 中断源识别	0xFFFF_FFFF
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I ² C1) 中断源识别	0xFFFF_FFFF
IRQ20_SRC	INT_BA+0x50	R	IRQ20 (CAN0) 中断源识别	0xFFFF_FFFF
IRQ21_SRC	INT_BA+0x54	R	保留	0xFFFF_FFFF
IRQ22_SRC	INT_BA+0x58	R	IRQ22 (PWM0) 中断源识别	0xFFFF_FFFF

IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (PWM1) 中断源识别	0XXXXX_XXXX
IRQ24_SRC	INT_BA+0x60	R	IRQ24 (BPWM0) 中断源识别	0XXXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (BPWM1) 中断源识别	0XXXXX_XXXX
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (BRAKE0) 中断源识别	0XXXXX_XXXX
IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (BRAKE1) 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 (CKD) 中断源识别	0XXXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	R	保留	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



6.2.8.7 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	保留	0xXXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	保留	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 (UART3) 中断源识别	0xXXXX_XXXX
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (UART4) 中断源识别	0xXXXX_XXXX
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (UART5) 中断源识别	0xXXXX_XXXX
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I ² C0) 中断源识别	0xXXXX_XXXX
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I ² C1) 中断源识别	0xXXXX_XXXX
IRQ20_SRC	INT_BA+0x50	R	IRQ20 (CAN0) 中断源识别	0xXXXX_XXXX
IRQ21_SRC	INT_BA+0x54	R	保留	0xXXXX_XXXX
IRQ22_SRC	INT_BA+0x58	R	IRQ22 (PWM0) 中断源识别	0xXXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (PWM1) 中断源识别	0xXXXX_XXXX
IRQ24_SRC	INT_BA+0x60	R	IRQ24 (BPWM0) 中断源识别	0xXXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (BPWM1) 中断源识别	0xXXXX_XXXX
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (BRAKE0) 中断源识别	0xXXXX_XXXX

IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (BRAKE1) 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 (CKD) 中断源识别	0xXXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	R	保留	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			

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
[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: UART3_INT
[2:0]	INT_BA+0x40	16	Bit2: 0 Bit1: 0 Bit0: UART4_INT
[2:0]	INT_BA+0x44	17	Bit2: 0 Bit1: 0 Bit0: UART5_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+0x50	20	Bit2: 0 Bit1: 0 Bit0: CAN0_INT
[2:0]	INT_BA+0x58	22	Bit2: 0 Bit1: 0 Bit0: PWM0_INT
[2:0]	INT_BA+0x5C	23	Bit2: 0 Bit1: 0 Bit0: PWM1_INT
[2:0]	INT_BA+0x60	24	Bit2: 0 Bit1: 0 Bit0: BPWM0_INT
[2:0]	INT_BA+0x64	25	Bit2: 0 Bit1: 0 Bit0: BPWM1_INT
[2:0]	INT_BA+0x68	26	Bit2: 0 Bit1: 0 Bit0: BRAKE0_INT
[2:0]	INT_BA+0x6C	27	Bit2: 0 Bit1: 0 Bit0: BRAKE1_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: CKD_INT



NMI 中断源选择控制寄存器 (NMI_SEL)

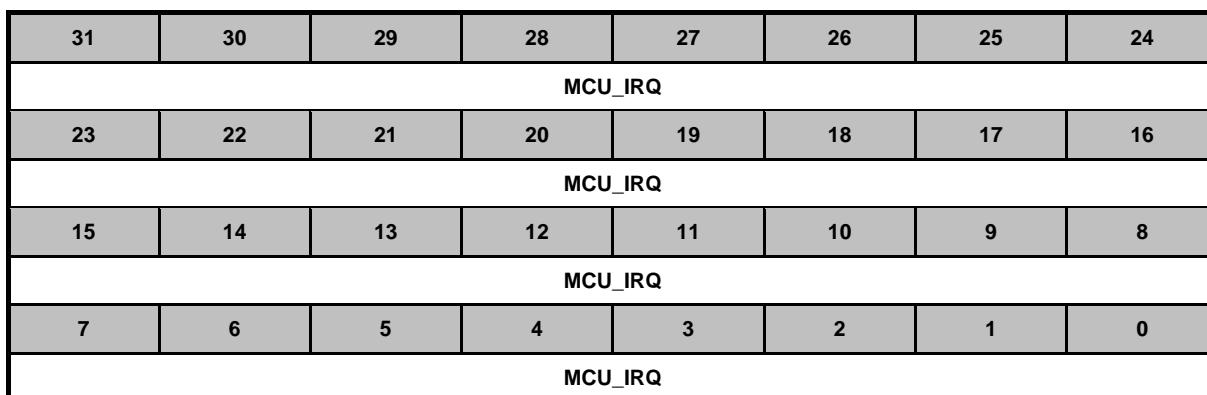
寄存器	偏移地址	R/W	描述	复位值
NMI_SEL	INT_BA+0x80	R/W	NMI 中断源选择控制寄存器	0x0000_0000

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							
7	6	5	4	3	2	1	0
Reserved			NMI_SEL				

位	描述	
[31:9]	Reserved	保留.
[8]	NMI_EN	<p>NMI 中断使能位 (写保护位) 0 = 禁用 NMI 中断 1 = 使能 NMI 中断</p> <p>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。</p>
[7:5]	Reserved	保留
[4:0]	NMI_SEL	<p>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。</p>

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



位	描述
[31:0]	<p>MCU_IRQ 源寄存器</p> <p>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。</p> <p>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。</p> <p>MCU_IRQ[n] 为 0 时：置 MCU_IRQ[n] 为 1，Cortex_M0 NVIC[n] 将产生一个中断。</p> <p>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。</p>

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	<p>快速IRQ等待使能位</p> <p>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。</p> <p>1 = MCU IRQ等待时间不固定，中断发生后，MCU进入IRQ中断。</p>



6.2.9 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”。

6.2.9.1 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



6.2.9.2 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							
23	22	21	20	19	18	17	16
Reserved				PART			
15	14	13	12	11	10	9	8
PARTNO							
7	6	5	4	3	2	1	0
PARTNO				REVISION			

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							
23	22	21	20	19	18	17	16
VECTORKEY							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					SYSRESETREQ	VECTCLKACITIVE	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	<p>挂起状态时的发送事件位</p> <p>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。</p> <p>1 = 使能事件和所有中断，包括禁用中断，都可以唤醒处理器。</p> <p>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。</p> <p>处理器也会在执行一个 SEV 指令或者一个外部事件时被唤醒。</p>
[3]	Reserved	保留
[2]	SLEEPDEEP	<p>系统深度休眠或休眠模式选择</p> <p>控制处理器使用休眠或者深度休眠作为它的低功耗模式：</p> <p>0 = 休眠</p> <p>1 = 深度休眠</p>
[1]	SLEEPONEXIT	<p>Sleep-On-Exit 使能位</p> <p>表示当从 Handler 模式切换到 Thread 模式时，是否使用 sleep-on-exit：</p> <p>0 = 当切换到 Thread 模式时不休眠。</p> <p>1 = 当从某个 ISR 切换到 Thread 模式时，进入休眠或者深度休眠。</p> <p>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。</p>
[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		Reserved					
23	22	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		Reserved					
23	22	21	20	19	18	17	16
PRI_14		Reserved					
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							

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

6.3 时钟控制器

6.3.1 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。下图展示了时钟发生器和时钟源控制的大概。

时钟发生器由如下4个时钟源组成：

- 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)

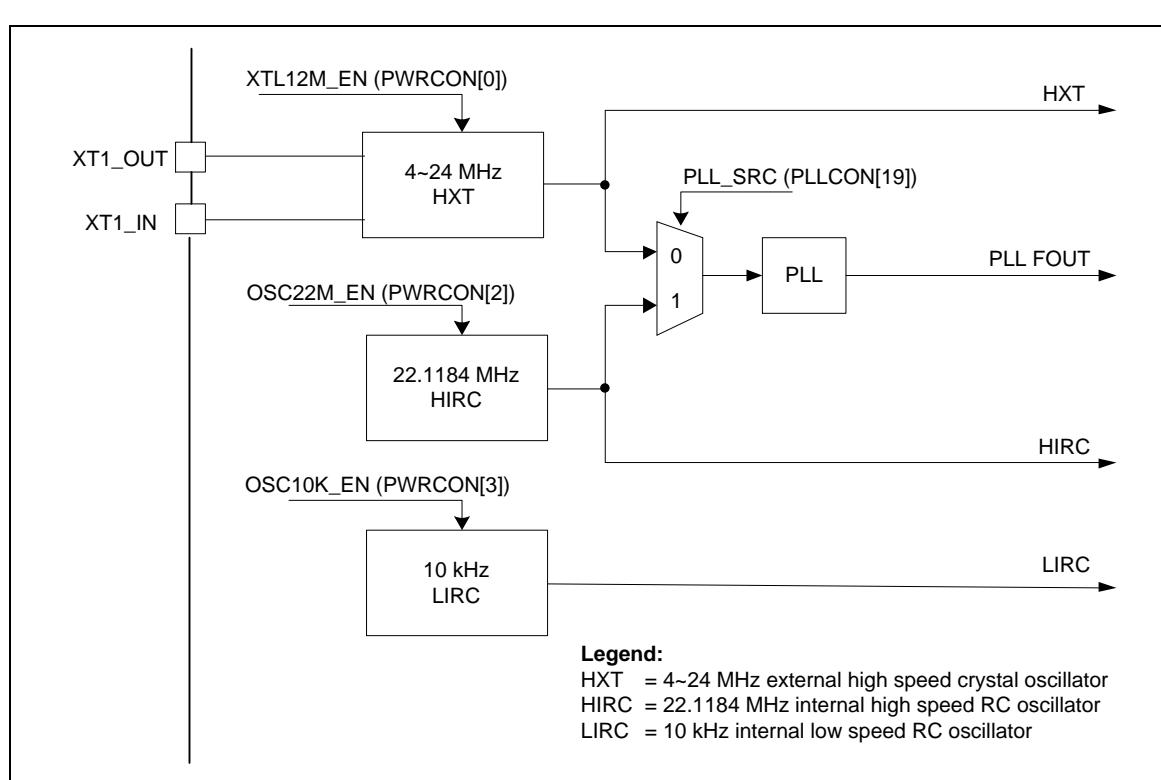


图 6-3 时钟发生器框图

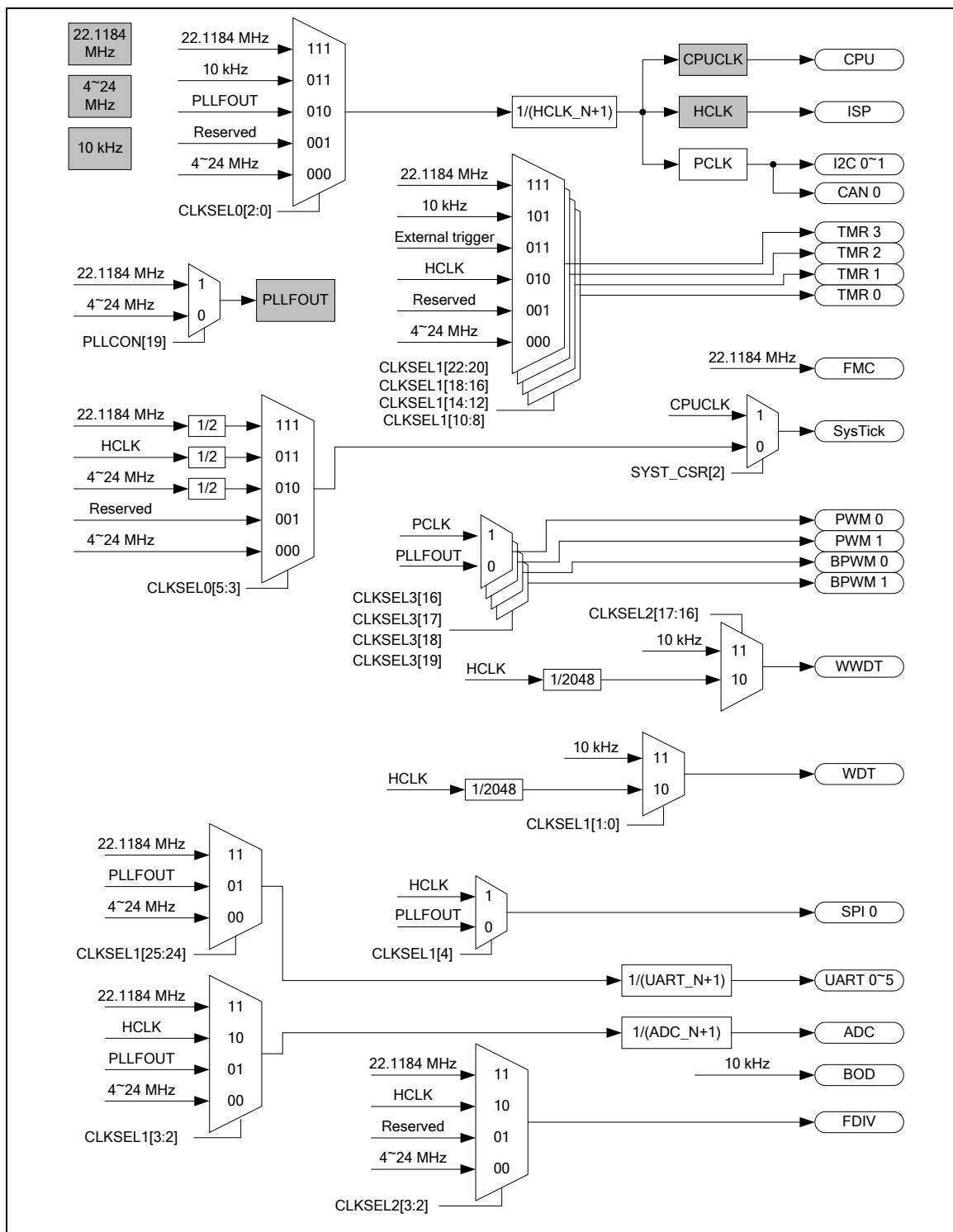


图 6-4 时钟发生器全局视图

6.3.2 系统时钟和SysTick 时钟

系统时钟有 4 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 6-5 所示

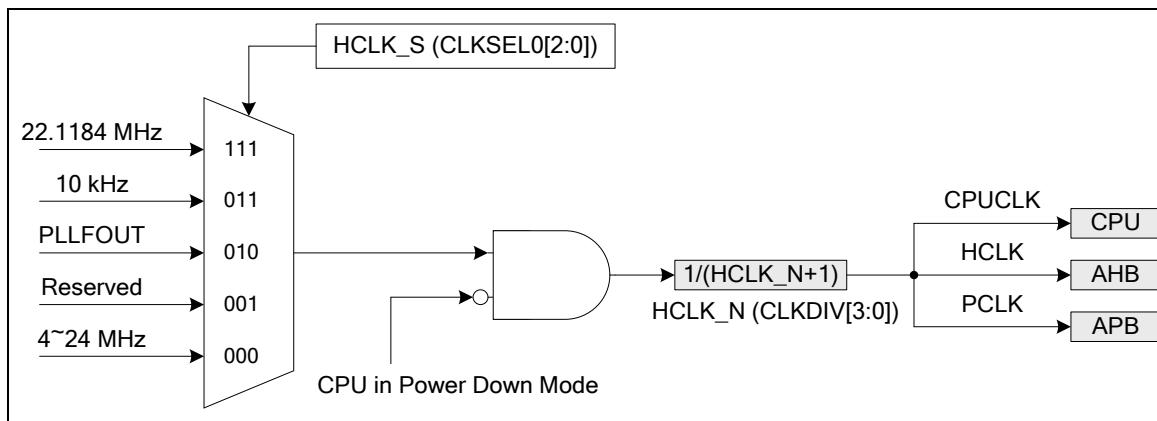


图 6-5 系统时钟框图

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 4 个时钟源。时钟源切换取决于寄存器 STCLK_S (CLKSEL0[5:3])。如图 6-6.

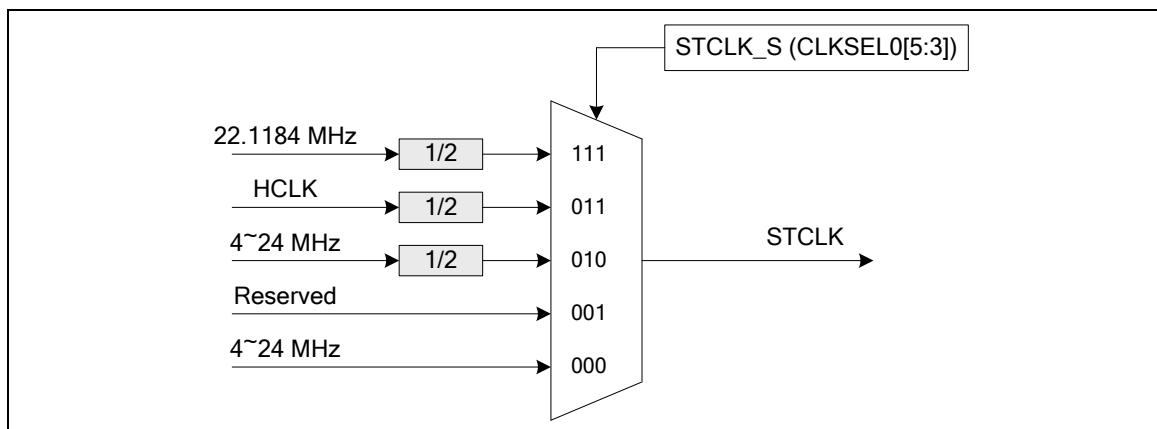


图 6-6 SysTick 时钟控制框图



6.3.3 掉电模式时钟

当芯片进入掉电模式，系统时钟，一些时钟源和一些外设时钟将被关闭。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
- WDT/Time 外设时钟(当时钟源使用10 kHz 内部低速RC振荡器)

6.3.4 分频器输出

该设备带有一个2的若干次幂的频率分频器，该分频器由16个链式的二分频器组成的移位寄存器。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。因此共有 2^N 种次幂的时钟分频选择，分频范围从 $F_{in}/2^1$ 到 $F_{in}/2^{16}$ ，此处 F_{in} 是到时钟分频器的时钟输入频率。

输出公式： $F_{out} = F_{in}/2^{(N+1)}$ ，其中 F_{in} 为输入时钟频率， F_{out} 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，分级计数器禁止。

如果DIVIDER1(FRQDIV[5])设置为1，分频器时钟(FRQDIV_CLK)将忽略2的若干次幂分频器。时钟频率将直接输出到CLKO。

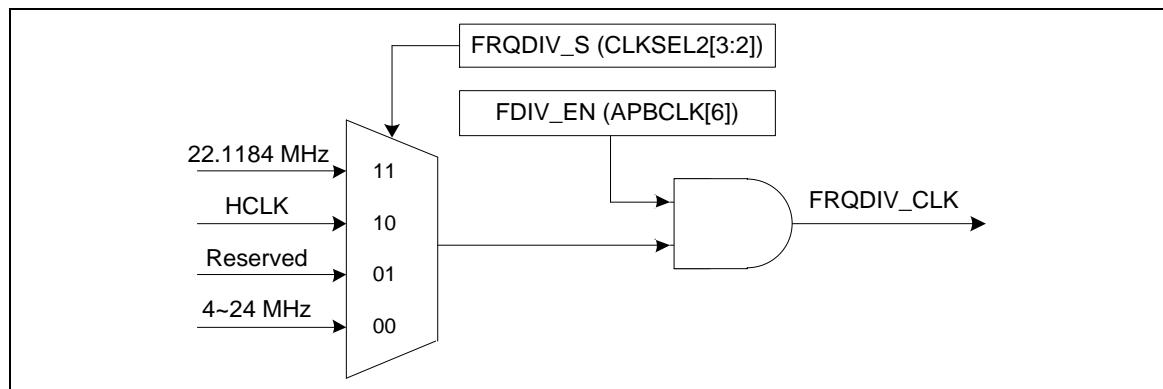


图 6-7 分频器的时钟源

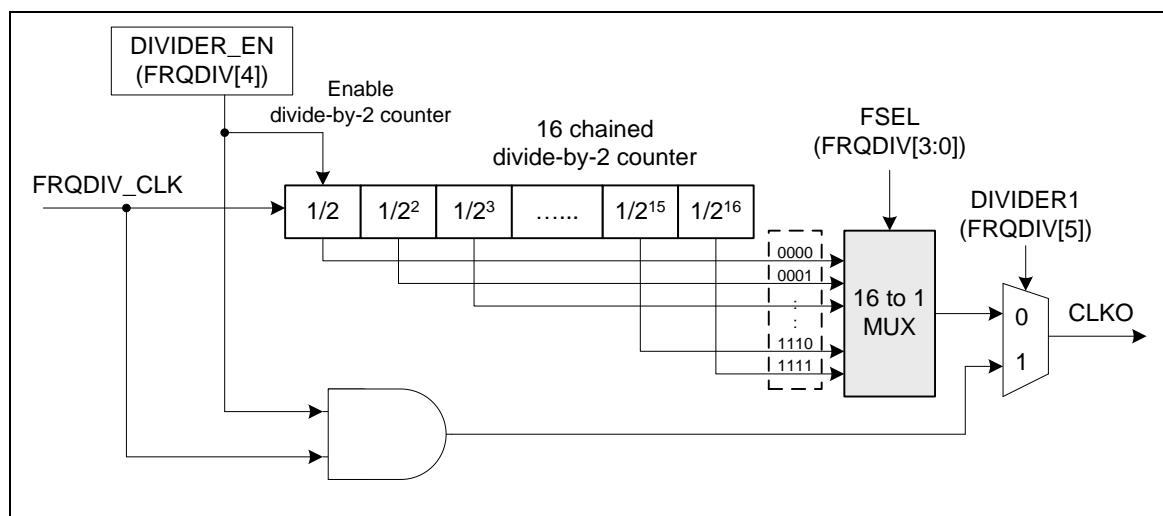


图 6-8 分频器框图



6.3.5 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
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
CLKDIV	CLK_BA+0x18	R/W	时钟分频数目寄存器	0x0000_0000
CLKSEL2	CLK_BA+0x1C	R/W	时钟源选择控制寄存器2	0x0002_00FF
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
APBCLK1	CLK_BA+0x30	R/W	APB 设备时钟使能控制寄存器 1	0x0000_0000
CLKSEL3	CLK_BA+0x34	R/W	时钟源选择控制寄存器 3	0x000F_003F
CLKDCTL	CLK_BA+0x70	R/W	时钟失败检测器控制寄存器	0x0000_0000
CLKDSTS	CLK_BA+0x74	R/W	时钟失败检测器状态寄存器	0x0000_0000
CDUPB	CLK_BA+0x78	R/W	时钟频率检测器上边界寄存器	0x0000_0000
CDLOWB	CLK_BA+0x7C	R/W	时钟频率检测器下边界寄存器	0x0000_0000



6.3.6 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	Reserved	XTL12M_EN

位	描述	
[31:9]	Reserved	保留.
[8]	PD_WAIT_CPU	<p>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。</p> <p>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)</p>
[7]	PWR_DOWN_EN	<p>系统掉电模式使能位(写保护) 当该位置被 ‘1’, 掉电模式使能, 掉电操作由PD_WAIT_CPU位来控制。</p> <p>(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)</p> <p>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。</p> <p>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, 但是内部 10 kHz 低速振荡器不受掉电模式的控制。</p> <p>在掉电模式下, PLL 与系统时钟将被禁用, 忽略时钟源选择。如果外设时钟源为内部 10 kHz 低速振荡器, 则外设的时钟不受掉电模式的控制。</p>

		<p>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)</p>
[6]	PD_WU_STS	<p>芯片掉电模式唤醒中断状态 该位由掉电唤醒事件被置位，表示从掉电模式恢复。。 如果GPIO, UART, WDT, I²C, TIMER, CAN, 或BOD 唤醒发生，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。</p>
[5]	PD_WU_INT_EN	<p>掉电模式唤醒中断使能位（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1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)</p>
[4]	PD_WU_DLY	<p>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。 当芯片需要工作在外部 4~24 MHz高速晶振条件下时，延迟时钟周期为 4096 个时钟周期； 当芯片需要工作在内部 22.1184 MHz 高速振荡器条件下时，延迟时钟周期为 256 个时钟周期。 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)</p>
[3]	OSC10K_EN	<p>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)</p>
[2]	OSC22M_EN	<p>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)</p>
[1]	Reserved	保留
[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。

		<p>0 = 禁用4 ~ 24 MHz 外部高速晶振(HXT)</p> <p>1 = 使能4~24 MHz 外部高速晶振(HXT)</p> <p>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)</p>
--	--	---

寄存器或指令模式	SLEEPDEEP (SCR[2])	PD_WAIT_CPU (PWRCON[8])	PWR_DOWN_EN (PWRCON[7])	CPU Run WiFi Instruction	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 及 选它为时钟源的WDT/Timer外设 时钟可运行。

表 6-5 芯片空闲/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， AHB总线设备时钟

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					ISP_EN	Reserved	

位	描述	
[31:3]	Reserved	保留
[2]	ISP_EN	Flash 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
[1:0]	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
Reserved			ADC_EN	Reserved			CAN0_EN
23	22	21	20	19	18	17	16
Reserved					UART2_EN	UART1_EN	UART0_EN
15	14	13	12	11	10	9	8
Reserved			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	Reserved	WDT_EN

位	描述	
[31:29]	Reserved	保留
[28]	ADC_EN	ADC 时钟使能位 0 = 禁用ADC 时钟 1 = 使能ADC 时钟
[27:25]	Reserved	保留
[24]	CAN0_EN	CAN 总线控制器时钟使能位 0 = 禁用CAN0 时钟 1 = 使能CAN0 时钟
[23:19]	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
[15:13]	Reserved	保留



[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	I²C1 时钟使能位 0 = 禁用 I ² C1 时钟 1 = 使能 I ² C1 时钟
[8]	I2C0_EN	I²C0 时钟使能位 0 = 禁用 I ² C0 时钟 1 = 使能 I ² C0 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]	Reserved	保留
[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)



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	Reserved	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 注意：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。新时钟稳定后，该位用于指示是否与用户要设定的系统时钟源相匹配。
[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]	Reserved 保留
[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	<p>Cortex™-M0 SysTick 时钟源选择(写保护)</p> <p>如果SYST_CSR[2] = 1, SysTick 时钟源来自 HCLK</p> <p>如果SYST_CSR[2] = 0, SysTick 时钟源由STCLK_S(CLKSEL0[5:3])定义.</p> <p>000 = 时钟源为外部 4~24 MHz 高速晶振时钟</p> <p>001 = 保留</p> <p>010 = 时钟源为外部 4~24 MHz 高速 晶振时钟/2分频</p> <p>011 = 时钟源为 HCLK/2分频</p> <p>111 = 时钟源为内部 22.1184 MHz 高速振荡器时钟/2分频</p> <p>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)。</p> <p>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</p>
[2:0]	HCLK_S	<p>HCLK时钟源选择(写保护)</p> <p>在时钟切换前, 相关时钟源 (预选和新选) 必须打开。</p> <p>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。</p> <p>000 = 时钟源为外部 4~24 MHz 高速 晶振时钟</p> <p>001 = 保留</p> <p>010 = 时钟源为 PLL 时钟</p> <p>011 = 时钟源为内部 10 kHz 低速振荡器时钟</p> <p>111=时钟源为内部 22.1184 MHz高速 振荡器时钟</p> <p>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)</p>



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
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
Reserved			SPI0_S	ADC_S		WDT_S	

位	描述
[31:26]	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 = 保留 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 = 保留 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	<p>TIMER1时钟源选择</p> <p>000 = 时钟源为外部 4~24 MHz 高速晶振时钟 001 = 保留 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</p>
[11]	Reserved	保留
[10:8]	TMR0_S	<p>TIMER0时钟源选择</p> <p>000 = 时钟源为外部 4~24 MHz 高速晶振时钟 001 = 保留 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</p>
[7:5]	Reserved	保留
[4]	SPI0_S	<p>SPI0时钟源选择</p> <p>0 = 时钟源来自 PLL 时钟 1 = 时钟源来自 HCLK</p>
[3:2]	ADC_S	<p>ADC时钟源选择</p> <p>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</p>
[1:0]	WDT_S	<p>看门狗定时器时钟源选择 (写保护位)</p> <p>00 = 保留 01 = 保留 10 = 时钟源为 HCLK /2048 clock 11 = 时钟源为内部 10 kHz 低速振荡器时钟</p> <p>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)。</p>

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
Reserved				HCLK_N			

位	描述	
[24:31]	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]	Reserved	保留
[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						WWDT_S	
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				FRQDIV_S		Reserved	

位	描述	
[31:18]	Reserved	保留
[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
[15:4]	Reserved	保留
[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 = 保留 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]	Reserved	保留



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 的工作模式。

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			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 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]	PLL OE (FOUT 使能) 管脚控制 0 = PLL FOUT 使能 1 = PLL FOUT 固定为低
[17]	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]	掉电模式 如果设置寄存器 PWRCON 的 PWR_DOWN_EN 位为 1, PLL 也将进入掉电模式。 0 = PLL 正常模式 1 = PLL 进入掉电模式 (默认)
[15:14]	PLL 输出分频控制位 参考下表公式
[13:9]	PLL 输入分频控制位 参考下表公式
[8:0]	PLL 反馈分频控制位 参考下表公式



输出时钟频率设置

$$F_{OUT} = F_{IN} \times \frac{NF}{NR} \times \frac{1}{NO}$$

约束条件:

1. $3.2MHz < F_{IN} < 150MHz$

2. $800KHz < \frac{F_{IN}}{2 * NR} < 7.5MHz$

$100MHz < F_{CO} = F_{IN} * \frac{NF}{NR} < 200MHz$

3. $120MHz < F_{CO}$ is preferred

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

$F_{IN} = 12 MHz$

$NR = (1+2) = 3$

$NF = (46+2) = 48$

$NO = 4$

$$F_{OUT} = 12/4 \times 48 \times 1/3 = 48 MHz$$

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		DIVIDER1	DIVIDER_EN	FSEL			

位	描述	
[31:6]	Reserved	保留
[5]	DIVIDER1	<p>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</p>
[4]	DIVIDER_EN	<p>频率分频器使能位 0 = 禁用频率分频器 1 = 使能频率分频器</p>
[3:0]	FSEL	<p>分频器输出频率选择位 输出频率的公式： $F_{out} = F_{in}/2^{(N+1)}$. 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。</p>



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				BPWM1_EN	BPWM0_EN	PWM1_EN	PWM0_EN
15	14	13	12	11	10	9	8
Reserved				UART5_EN	UART4_EN	UART3_EN	
7	6	5	4	3	2	1	0
Reserved							

位	描述	
[31:20]	Reserved	保留
[19]	BPWM1_EN	BPWM1 时钟使能位 0 = 禁止BPWM1 时钟 1 = 使能 BPWM1 时钟
[18]	BPWM0_EN	BPWM0 时钟使能位 0 = 禁止BPWM0 时钟 1 = 使能BPWM0 时钟
[17]	PWM1_EN	PWM1 时钟使能位 0 = 禁止PWM1时钟 1 = 使能 PWM1时钟
[16]	PWM0_EN	PWM0 时钟使能位 0 = 禁止PWM0时钟 1 = 使能 PWM0时钟
[15:11]	Reserved	保留
[10]	UART5_EN	UART5 时钟使能位 0 = 禁止UART5时钟 1 = 使能UART5时钟
[9]	UART4_EN	UART4 时钟使能位 0 = 禁止UART4时钟 1 = 使能UART4时钟

[8]	UART3_EN	UART3 时钟使能位 0 = 禁止UART3时钟 1 = 使能UART3时钟
[7:0]	Reserved	保留

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	0x000F_003F

31	30	29	28	27	26	25	24			
Reserved										
23	22	21	20	19	18	17	16			
Reserved			BPWM1_S		BPWM0_S		PWM1_S		PWM0_S	
15	14	13	12	11	10	9	8			
Reserved										
7	6	5	4	3	2	1	0			
Reserved										

位	描述
[31:20]	Reserved 保留
[19]	BPWM1_S BPWM1 时钟源选择 BPWM1的引擎时钟源由BPWM1_S定义 0 = 时钟源为 PLL. 1 =时钟源为 PCLK.
[18]	BPWM0_S BPWM0 时钟源选择 BPWM0的引擎时钟源由BPWM0_S定义 0 = 时钟源为 PLL. 1 =时钟源为 PCLK.
[17]	PWM1_S PWM1 时钟源选择 PWM1的引擎时钟源由PWM1_S定义 0 = 时钟源为 PLL. 1 =时钟源为 PCLK.
[16]	PWM0_S PWM0 时钟源选择 PWM0的引擎时钟源由PWM0_S定义 0 = 时钟源为 PLL. 1 =时钟源为 PCLK.
[15:0]	Reserved 保留



时钟失败检测器控制寄存器 (CLKDCTL)

寄存器	偏移地址	R/W	描述	复位值
CLKDCTL	CLK_BA+0x70	R/W	时钟失败检测器控制寄存器	0x0000_0000

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

位	描述	
[31:18]	Reserved	保留
[17]	HXTFQIEN	HXT 时钟频率监测中断使能位 0 = 禁止 HXT 时钟频率监测失败中断 1 = 使能 HXT 时钟频率监测失败中断
[16]	HXTFQDEN	HXT 时钟频率监测使能位 0 = 禁止 HXT 时钟频率监测. 1 = 使能 HXT 时钟频率监测
[15:6]	Reserved	保留
[5]	HXTFIEN	HXT 时钟失败中断使能位 0 = 禁止 HXT 时钟失败中断 1 = 使能 HXT 时钟失败中断
[4]	HXTFDEN	HXT 时钟失败检测器使能位 0 = 禁止 HXT 时钟失败检测器. 1 = 使能 HXT 时钟失败检测器
[3:0]	Reserved	保留



时钟失败检测器状态寄存器(CLKDSTS)

寄存器	偏移地址	R/W	描述	复位值
CLKDSTS	CLK_BA+0x74	R/W	时钟失败检测器状态寄存器	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							

位	描述	
[31:9]	Reserved	保留
[8]	HXTFQIF	HXT 时钟频率监测中断标志位 0 = HXT 时钟正常 1 = HXT 时钟异常(写1清除)
[7:1]	Reserved	保留
[0]	HXTFIF	HXT 时钟失败中断标志位 0 = HXT 时钟正常 1 = HXT 时钟停止(写1清除)



时钟频率检测器上边界寄存器 (CDUPB)

寄存器	偏移地址	R/W	描述	复位值
CDUPB	CLK_BA+0x78	R/W	时钟频率检测器上边界寄存器	0x0000_0000

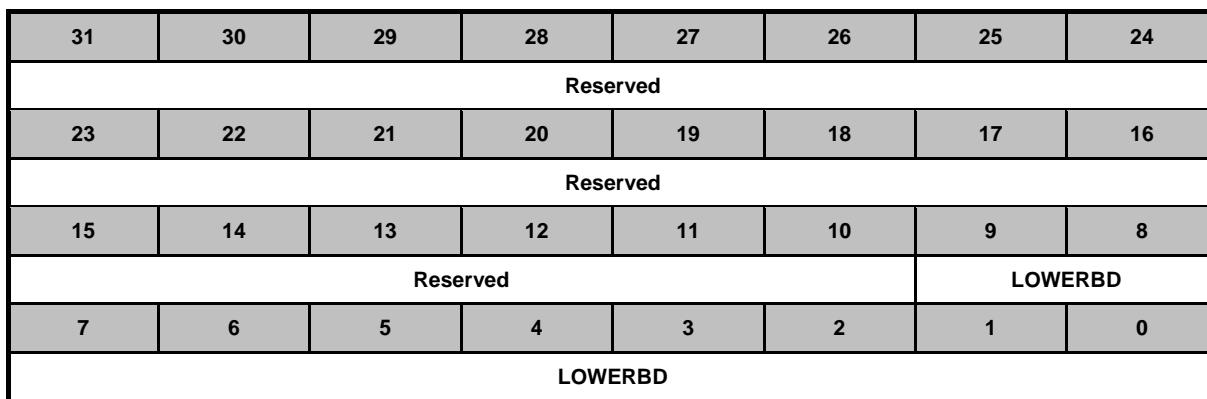
31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						UPERBD	
7	6	5	4	3	2	1	0
UPERBD							

位	描述	
[31:10]	Reserved	保留
[9:0]	UPERBD	HXT 时钟频率检测器上边界 该位定义频率监测窗口的最大值 当HXT频率监测值高于该寄存器的值时，HXT频率检测失败中断标志将设置为 1。



时钟频率检测器下边界寄存器(CDLOWB)

寄存器	偏移地址	R/W	描述	复位值
CDLOWB	CLK_BA+0x7C	R/W	时钟频率检测器上边界寄存器	0x0000_0000



位	描述	
[31:10]	Reserved	保留
[9:0]	LOWERBD	HXT 时钟频率检测器下边界 该位定义频率监测窗口的最小值 当HXT频率监测值低于该寄存器的值时，HXT频率检测失败中断标志将设置为 1.



6.4 Flash存储控制器 (FMC)

6.4.1 概述

NuMicro™ NUC131系列 具有68/36K 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) 用户可以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™ NUC131系列也提供了数据 flash, 用来存储用户想要的数据。

NuMicro™ NUC131的数据flash是由config0,config1寄存器灵活配置的, Config0中的 DFVSEN位决定了数据flash的大小是否可以由用户配置, DFEN位使能数据flash, Config1中的DFBADR域决定了数据flash的起始地址。当DFVSEN为1时, 数据flash的大小固定4K, 起始地址为: 0x0001_F000, APROM 空间为64/32K。当DFVSEN为0且DFEN为1时, 数据flash空间为0, APROM空间为68/36K字节。当DFVSEN为0且DFEN为0时, APROM和数据flash共享68/36K连续地址空间, 数据flash的起始地址由Config1中的DFBADR域决定

6.4.2 特性

- 连续地址读访问零等待状态时, 最高可达50 MHz
- 所有嵌入Flash支持512字节页擦除
- 68/36 KB应用程序存储空间(APROM)
- 4KB在系统编程 (ISP) 加载程序空间(LDROM)
- 可配置数据FLASH空间
- 支持在应用编程(IAP) 在APROM 和LDROM之间切换代码, 不用复位
- 支持在系统编程(ISP)更新片上Flash

6.4.3 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:

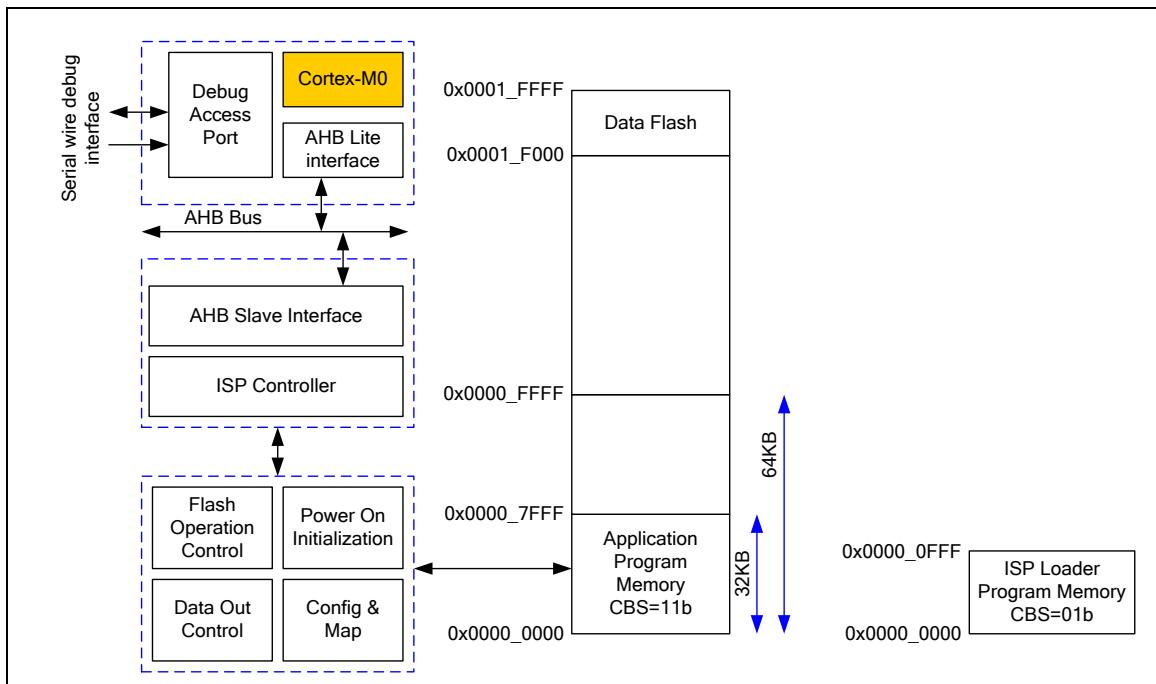


图 6-9 Flash 存储器控制框图 (DFVSEN = 1)

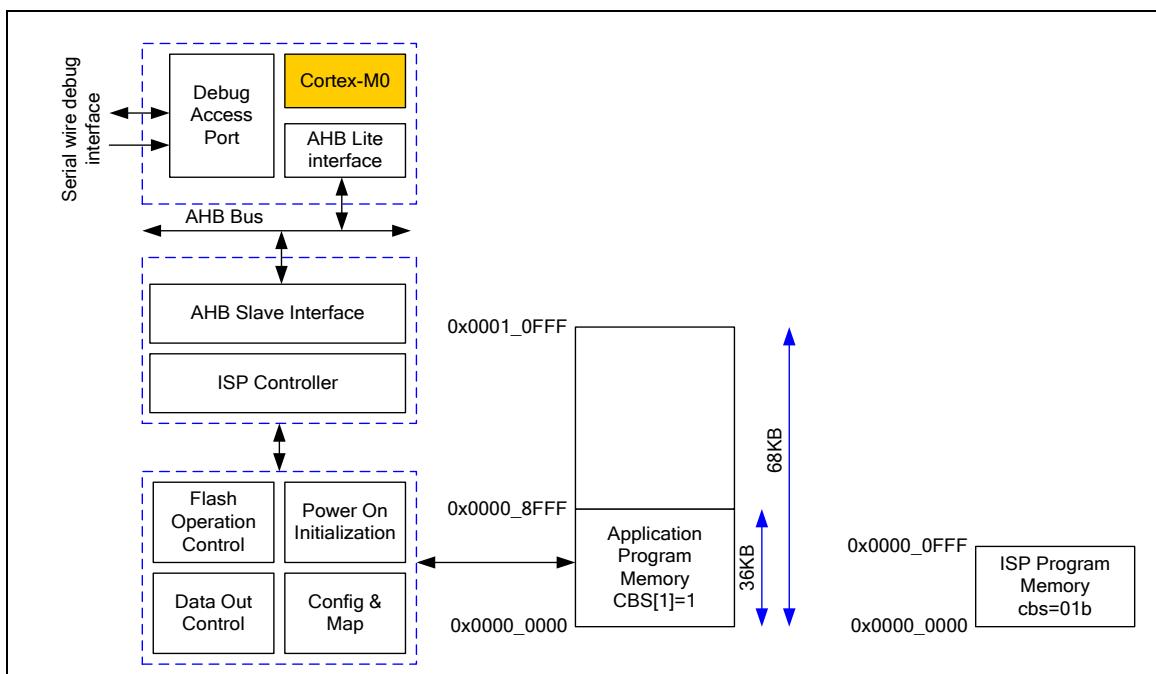


图 6-10 Flash 存储器控制框图 (DFVSEN = 0)



6.4.4 功能描述

6.4.4.1 Flash 存储器结构

NuMicro™ NUC131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。当DFVSEN为1时, 数据flash的大小固定4K, 起始地址为: 0x0001_F000, 当DFVSEN为0且DFEN为1时, 数据flash空间为0, APROM空间为68/36K字节。当DFVSEN为0且DFEN为0时, APROM和数据flash共享68/36K连续地址空间, 数据flash的起始地址由Config1中的DFBADR域决定

用户配置提供了几个字节来控制系统逻辑, 如FLASH安全锁, 启动选择, 欠压电平, 数据FLASH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 ~ 0xFFFF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_0FFF去读。

表 6-6 和图 6-11 显示了36/68KB设备的APROM, LDROM, 数据Flash和用户配置 的地址映像信息

模块名称	类型	大小	起始地址	结束地址
APROM	36 KB	32 KB	0x0000_0000	0x0000_7FFF
	68 KB	64 KB	0x0000_0000	0x0000_FFFF
Data Flash	36 KB	4 KB	0x0001_F000	0x0001_FFFF
	68 KB	4 KB	0x0001_F000	
LDROM	36/68 KB	4 KB	0x0010_0000	0x0010_0FFF
User Configuration	36/68 KB	2 words	0x0030_0000	0x0030_0004

表 6-6 存储器地址映像 (DFVSEN = 1)

模块名称	类型	大小	起始地址	结束地址
APROM	36 KB	(36-0.5*N) KB	0x0000_0000	0x0000_8FFF (36KB, if DFEN=1) DFBADR-1 (if DFEN=0)
	68 KB	(68-0.5*N) KB	0x0000_0000	0x0001_0FFF (68KB, if DFEN=1) DFBADR-1 (if DFEN=0)
Data Flash	36 KB	4 KB	0x0001_F000	0x0000_8FFF (36KB, if DFEN=0)
	68 KB	4 KB	0x0001_F000	0x0001_0FFF (68KB, if DFEN=0)
LDROM	36/68 KB	4 KB	0x0010_0000	
User Configuration	36/68 KB	2 words	0x0030_0000	0x0030_0004

表 6-7 存储器地址映像 (DFVSEN = 0)

Flash存储器组织结构如图 6-11:

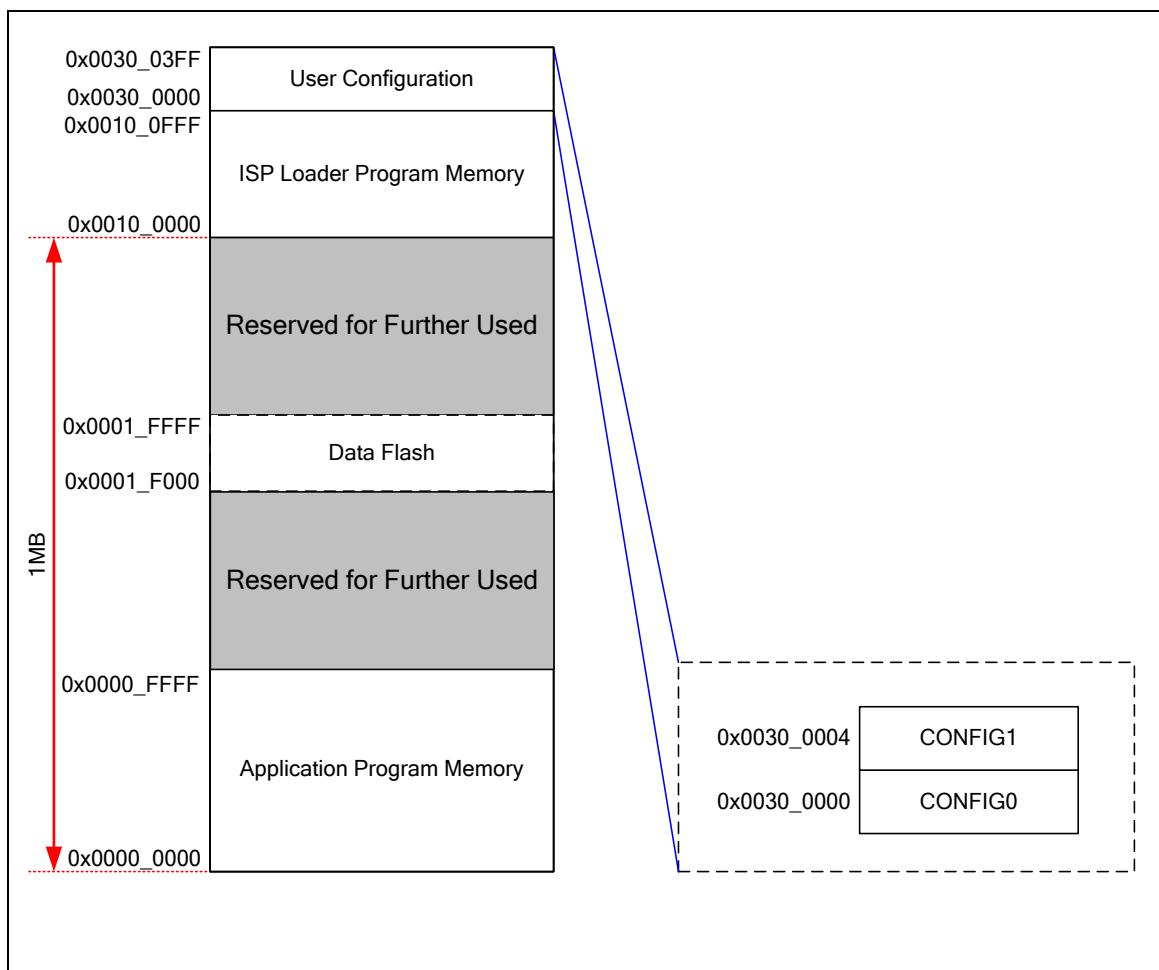


图 6-11 Flash存储器组织结构 (DFVSEN = 1)

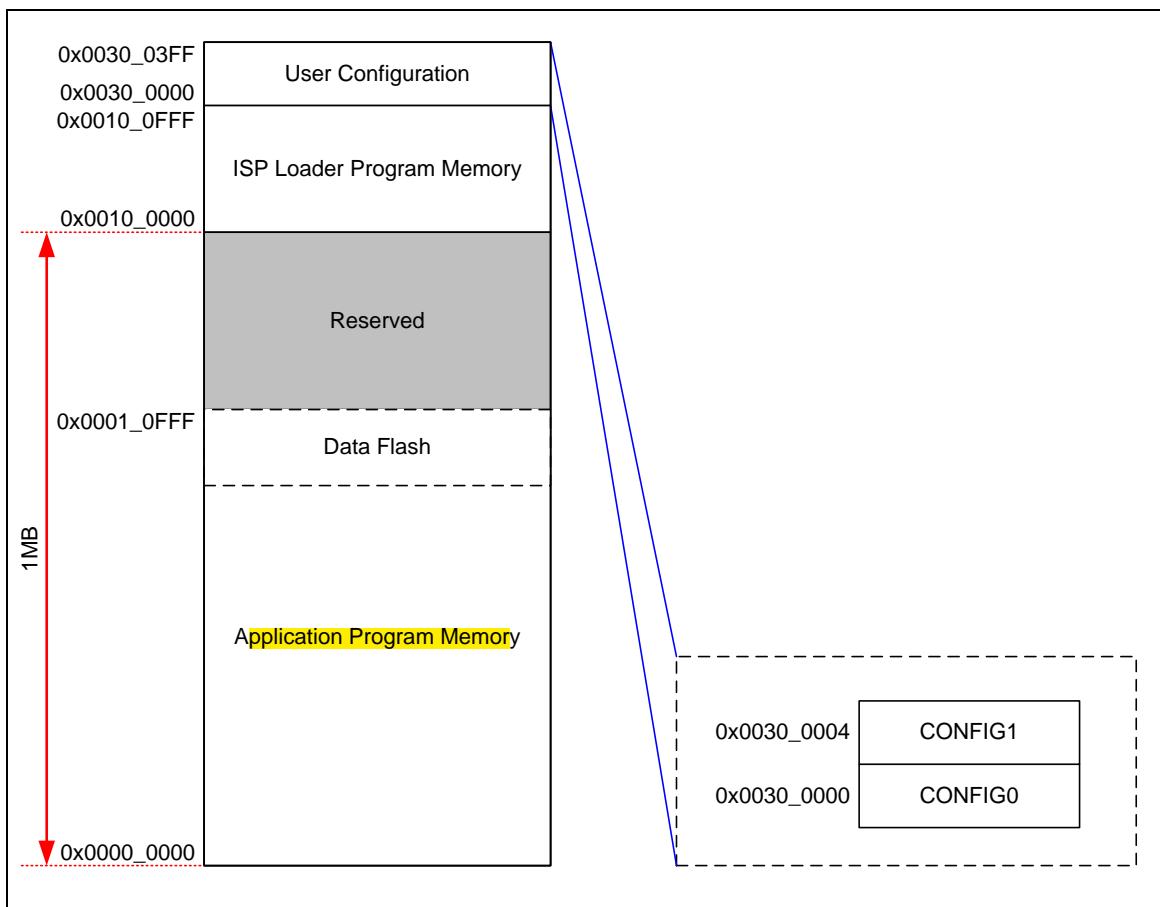


图 6-12 Flash存储器组织结构 (DFVSEN = 0)



6.4.4.2 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[2]	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	CWDTE[1:0]		DFVSEN	LOCK	DFEN

CONFIG0	地址 = 0x0030_0000	
位	描述	
[31]	CWDTEN[2]	看门狗定时器硬件使能位 如果开启看门狗定时器硬件使能功能，上电后看门狗使能位WDTEN(WDT_CTL[7])及看门狗复位使能位RSTEN (WDT_CTL[1])被自动置位，看门狗定时器时钟源强制为LIRC并且LIRC不能被关掉 CWDTEN[2:0] 即是 CONFIG0[31][4][3], 011=WDT硬件使能功能开启，WDT时钟源除芯片在掉电模式时一直是开启的，当芯片进入掉电模式，如果CWDTPDEN为0则WDT时钟源一直开启；如果CWDTPDEN为1，WDT时钟源由LIRCEN (CLK_PWRCTL[3])控制，请参考CWDTPDEN描述 111 = WDT硬件使能功能关闭 其他 = WDT硬件使能功能开启， WDT时钟一直开启。
[30]	CWDTPDEN	看门狗时钟掉电使能位 0 = OSC10K 看门狗定时器时钟源一直强制使能 1 = OSC10K 看门狗定时器时钟源在掉电模式下由OSC10K_EN (PWRCON[3]) 控制 注：此位只有在CWDTEN 为0时有效。
[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：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 注: 对PF.0, PF.1, 当CGPFMFP (CONFIG0[27]) 为0时该位有效
[9:8]	Reserved	Reserved
[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]	Reserved	保留

[4:3]	CWDTEN[1:0]	<p>看门狗定时器硬件使能位</p> <p>如果开启看门狗定时器硬件使能功能，上电后看门狗使能位WDTEN(WDT_CTL[7])及看门狗复位使能位RSTEN (WDT_CTL[1])被自动置位，看门狗定时器时钟源强制为LIRC并且LIRC不能被关掉</p> <p>CWDTEN[2:0] 即是 CONFIG0[31][4][3].</p> <p>011=WDT硬件使能功能开启，WDT时钟源除芯片在掉电模式时一直是开启的，当芯片进入掉电模式，如果CWDTPDEN为0则WDT时钟源一直开启；如果CWDTPDEN为1，WDT时钟源由LIRCEN (CLK_PWRCTL[3])控制，请参考CWDTPDEN描述</p> <p>111 = WDT硬件使能功能关闭</p> <p>其他 = WDT硬件使能功能开启，WDT时钟一直开启。</p>
[2]	DFVSEN	<p>数据Flash可变空间使能</p> <p>0 = 数据 flash 空间可设置，起始地址由DFBADR (Config1)决定</p> <p>1 = 数据 flash 空间固定为4K字节</p>
[1]	LOCK	<p>安全加密</p> <p>0 = 加密FLASH数据</p> <p>1 = 解除Flash 数据加密</p> <p>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</p> <p>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</p>
[0]	DFEN	<p>数据 Flash 使能位</p> <p>0 = 数据Flash使能</p> <p>1 = 数据Flash禁用</p> <p>注:此位只有在DFVSEN为0时有效。当DFVSEN为0且DFEN为1时，数据flash空间为0，APROM空间为68/36K字节。当DFVSEN为0且DFEN为0时，APROM和数据flash共享68/36K连续地址空间，数据flash的起始地址由Config1中的DFBADR域决定</p>

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

Config	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 如果DFVSEN为0且DFEN为0, 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

6.4.4.3 启动选择

NuMicro™ NUC131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4kB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 6-13 表示当系统从APROM和LDROM启动时的储存器映射。

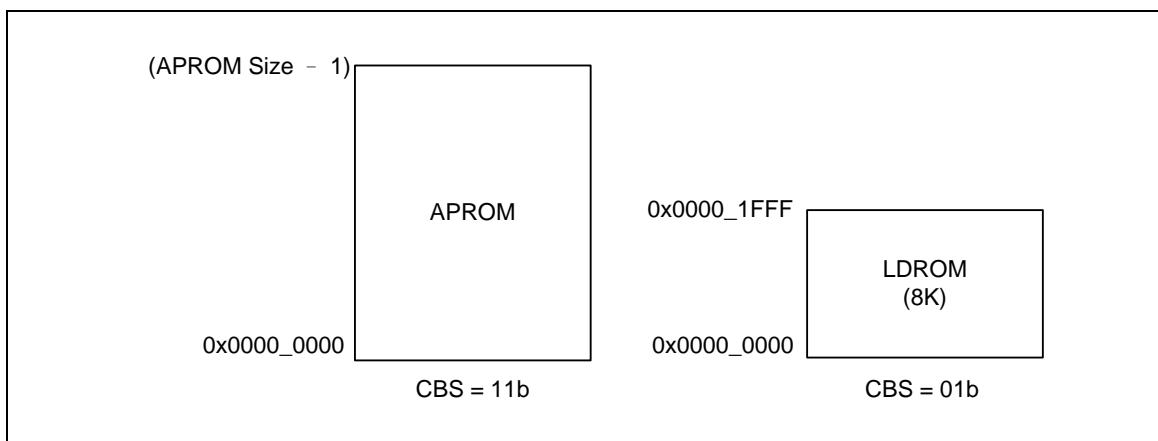


图 6-13 APROM 和 LDROM 启动的程序执行范围

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)。

6.4.4.4 在应用编程(IAP)

NuMicro NUC131系列提供了在应用编程（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4 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。也就是4 KB LDROM也被映射到0x0010_0000~0x0010_1FFF。

IAP功能使能时，地址空间映像请参考图 6-14

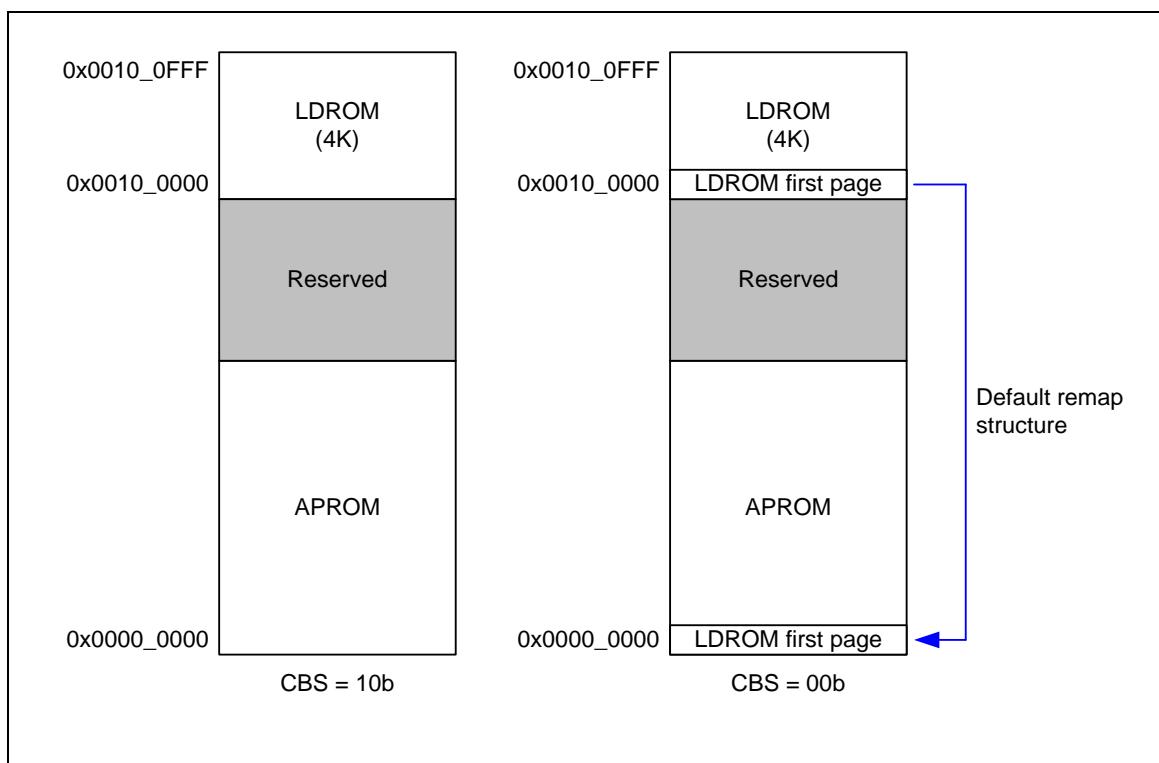


图 6-14 IAP使能时代码的执行范围

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像地址到ISPADR改变第一可执行页的重新映像地址，通过读ISPSTA 寄存器中VECMAP域，用户可以检查改变是否成功。

6.4.4.5 在系统编程(ISP)

NuMicro™ NUC131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。

6.4.4.6 ISP流程

NuMicro™ NUC131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：

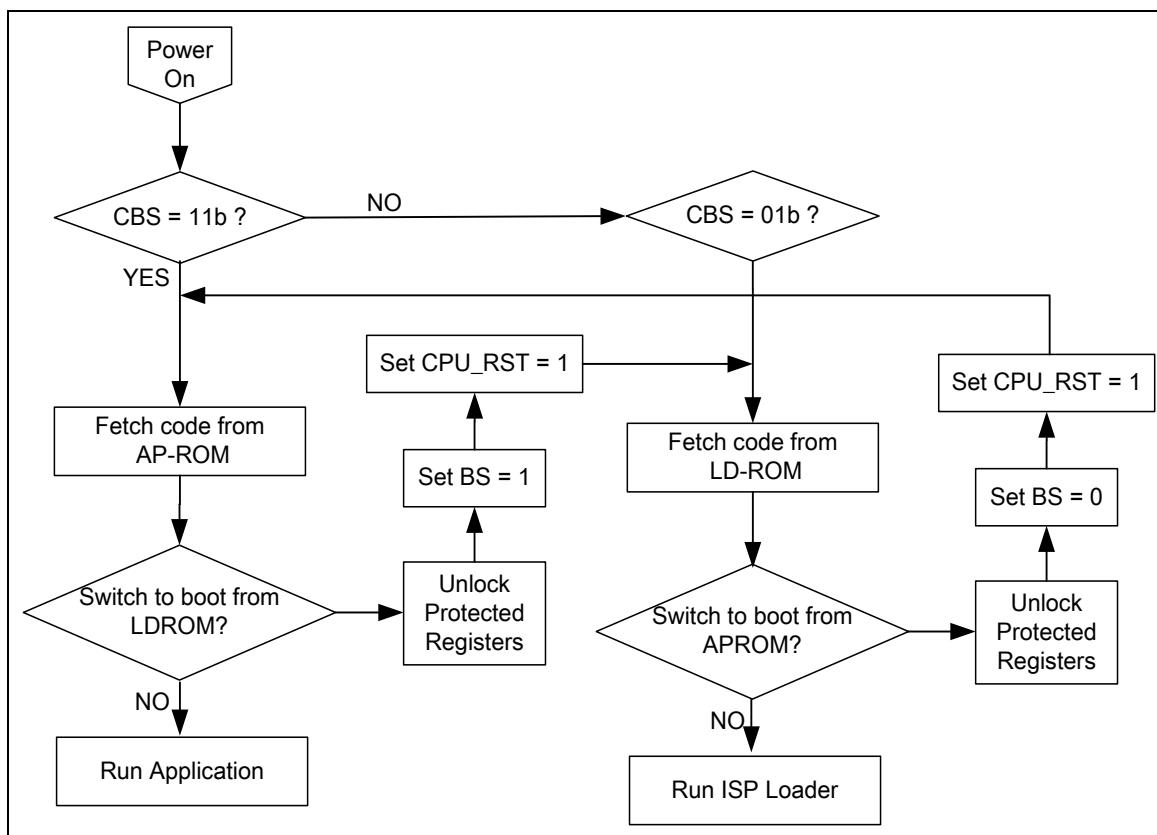


图 6-15 BS位启动选择流程示例

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ISPFF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。

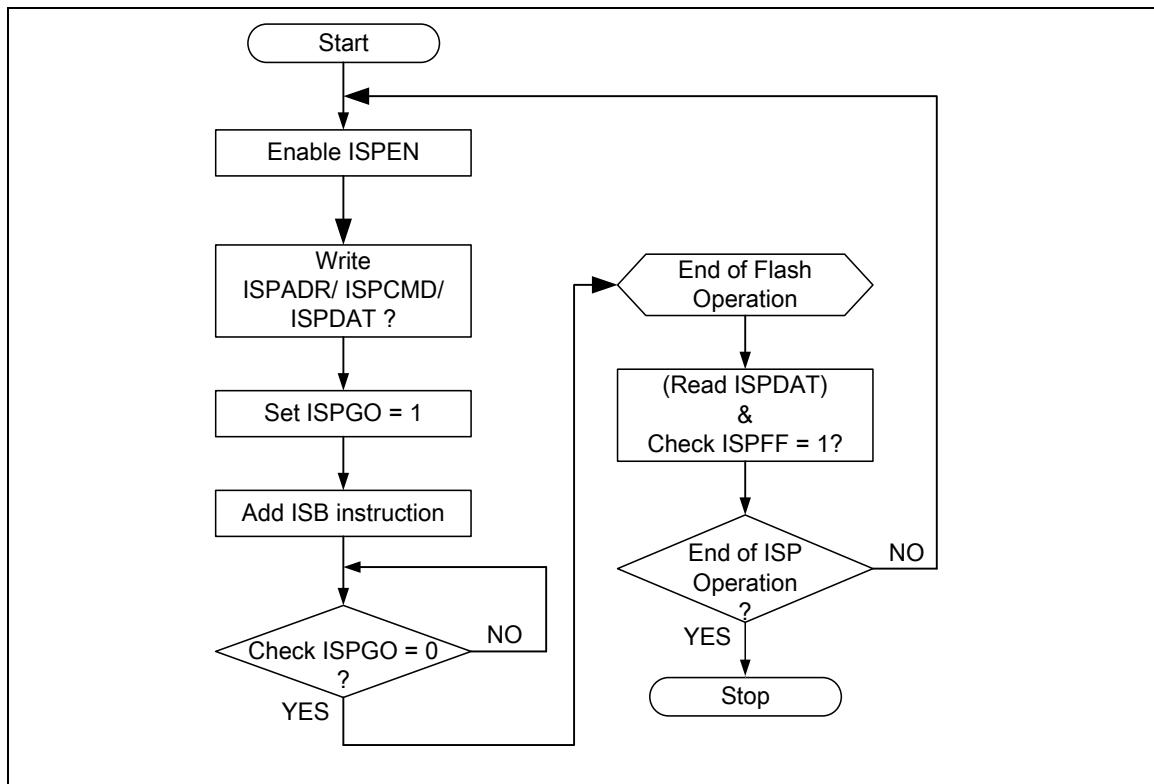


图 6-16 ISP 流程示例

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 6-8 ISP 命令表



6.4.5 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



6.4.6 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
Reserved							
23	22	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	ISPFF	LDUEN	CFGUEN	APUEN	Reserved	BS	ISPEN

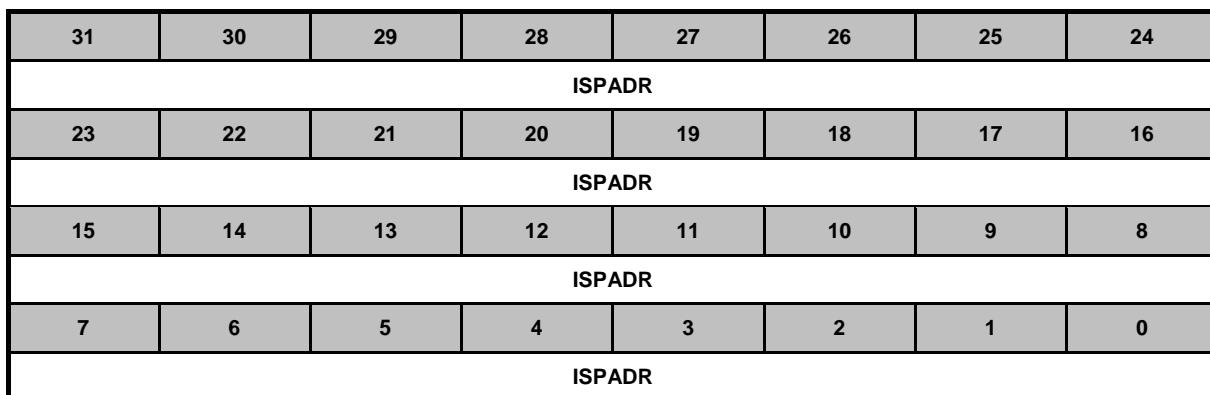
位	描述	
[31:7]	保留	保留.
[6]	ISPFF	ISP失败标志 (写保护位) 当ISP满足下列条件时，该位由硬件置位： (1) APUEN等于0时，APROM 写APROM. (2) LDUEN等于0时，LDROM 写LDROM. (3) CFGUEN等于0时，CONFIG 被擦除或编程. (4) 定义地址无效，如超过正常范围. 该位写 1 清除
[5]	LDUEN	LDROM更新使能 位(写保护位) 0 = 禁止LDROM更新 1 =当芯片在 APROM 中运行时， LDROM 可以更新。
[4]	CFGUEN	使能由 ISP 更新配置位 (写保护位) 0 = 禁止ISP更新配置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



位	描述	
[31:0]	ISPADR	ISP 地址 The NuMicro™ NUC131系列内嵌最大17Kx32 (68 KB) Flash, 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	偏移量	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							
23	22	21	20	19	18	17	16
ISPDAT							
15	14	13	12	11	10	9	8
ISPDAT							
7	6	5	4	3	2	1	0
ISPDAT							

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	偏移量	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
Reserved							
23	22	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		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)

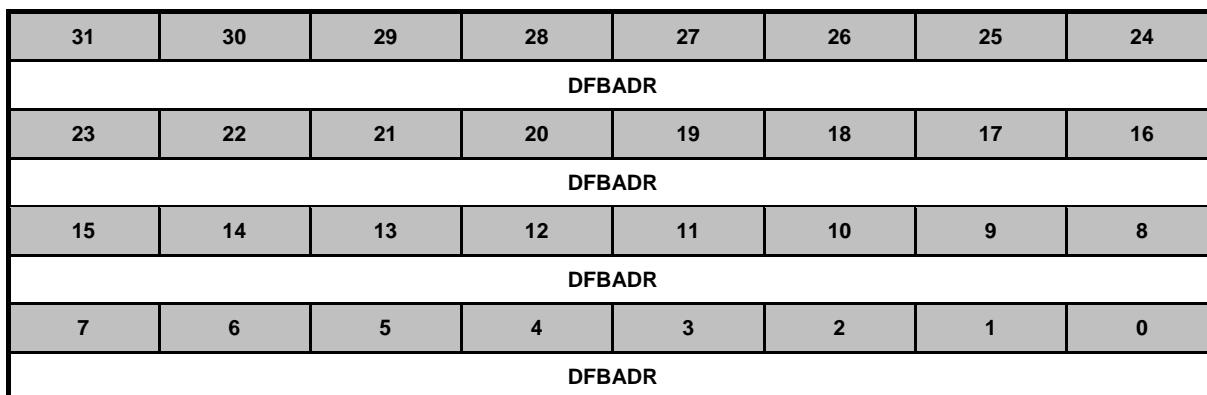
寄存器	偏移量	R/W	描述	复位值
ISPTRG	FMC_BA+0x10	R/W	ISP触发控制寄存器	0x0000_0000

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

位	描述	
[31:1]	保留	保留
[0]	ISPGO	<p>ISP开始触发 (写保护)</p> <p>写 1 开始ISP操作，当ISP操作结束后，该位由硬件自动清零。</p> <p>0 = ISP 操作结束 1 = ISP 正在执行</p> <p>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REGWRPROT，地址GCR_BA+0x100。</p>

数据FLASH基地址寄存器(DFBADR)

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



位	描述	
[31:0]	DFBADR	<p>数据FLASH基地址</p> <p>该寄存器为数据FLASH开始地址寄存器, 只读.</p> <p>当DFVSEN为0时, data flash与APROM共享flash空间, 数据flash大小由用户配置, 本寄存器的值从config1中获取</p> <p>当DFVSEN为1时, data flash的空间固定为4K, 该寄存器的值固定为0x0001_F000</p>



Flash访问时间控制寄存器(FATCON)

寄存器	偏移量	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
Reserved							
23	22	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	FOMSEL1	Reserved	FOMSEL0	Reserved			

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	<p>芯片频率优化模式选择0（写保护位） 当芯工作频率低于 25 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=保留 11=保留 其中00表示FOMSEL1 = 0, FOMSEL0 = 0; 01 表示 FOMSEL1 = 0, FOMSEL0 = 1, 等等</p>
[3:0]	保留	保留



ISP 状态寄存器(ISPSTA)

寄存器	偏移量	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				
15	14	13	12	11	10	9	8
VECMAP							
7	6	5	4	3	2	1	0
Reserved	ISPFF	Reserved			CBS		ISPGO

位	描述	
[31:21]	Reserved	保留.
[20:9]	VECMAP	<p>向量页映像地址 (写保护) 当前 flash 地址空间 0x0000_0000~0x0000_01FF 映像到地址 {VECMAP[11:0], 9'h000} ~ {VECMAP[11:0], 9'h1FF}</p>
[8:7]	Reserved	Reserved.
[6]	ISPFF	<p>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.</p>
[5:3]	Reserved	保留.
[2:1]	CBS	<p>启动选择状态 (只读) 这是CBS在CONFIG0中的镜像</p>
[0]	ISPGO	<p>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</p>



6.5 通用 I/O (GPIO)

6.5.1 概述

NuMicro™ NUC131 系列多达56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。56个管脚分配在GPIOA, GPIOB, GPIOC, GPIOD, GPIOE与GPIOF六个端口上。GPIOA/B最多有16个管脚，GPIOC最多有12个管脚，GPIOD最多有4个管脚，GPIOE最多有1个管脚，GPIOF最多7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。

6.5.2 特性

- 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。
- 使能管脚中断功能也将同时使能了唤醒功能。

6.5.3 基本配置

GPIO 管脚 通过 配置 GPA_MFP, GPB_MFP, GPC_MFP, GPD_MFP, GPE_MFP, GPF_MFP, ALT_MFP, ALT_MFP2, ALT_MFP3, 及 ALT_MFP4 寄存器来实现其功能

6.5.4 功能描述

6.5.4.1 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。

6.5.4.2 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

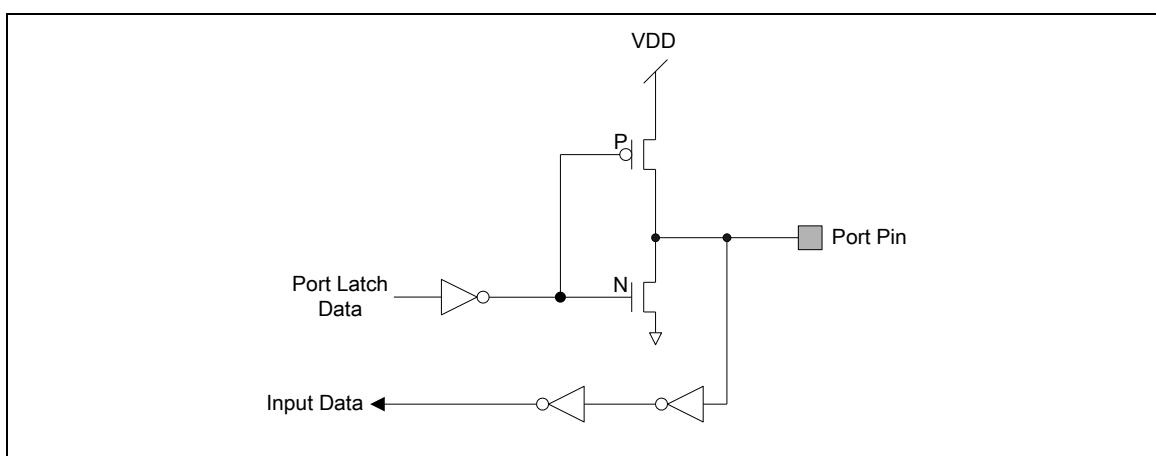


图 6-17 推挽输出

6.5.4.3 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

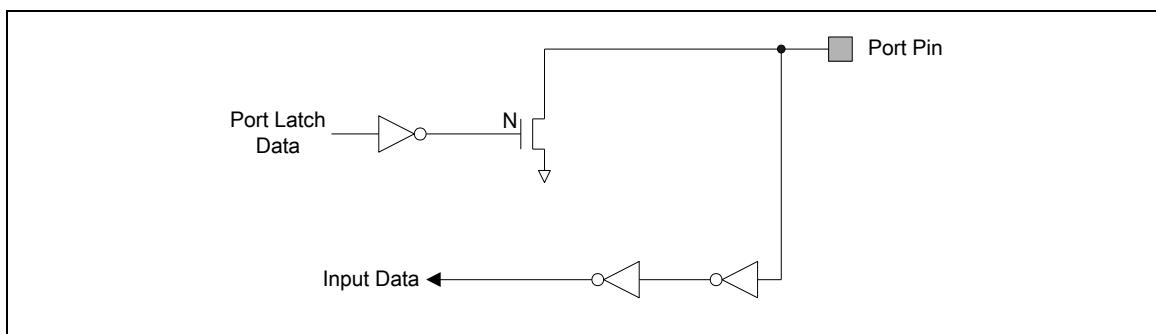


图 6-18 开漏输出

6.5.4.4 混双向模式说明

设置 GPIOx_PMD(PMDn[1:0]) 为 11b，GPIOxport[n] 为混双向模式，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。混双向输出是 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)

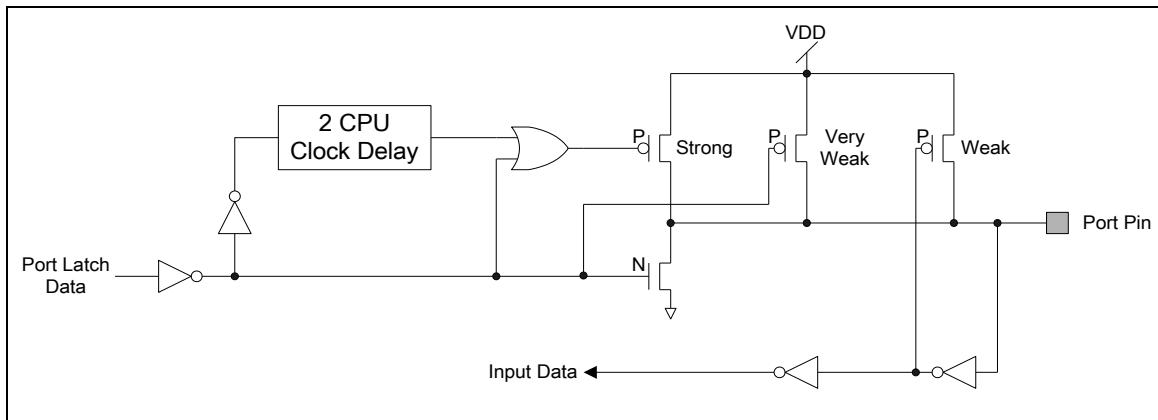


图 6-19 准双向I/O模式

6.5.4.5 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。



6.5.5 寄存器映射

R: 只读, **W:** 只写, **R/W:** 读/写

寄存器	偏移量	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
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	0xX0XX_X0XX
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_CFCF
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

寄存器	偏移量	R/W	描述	复位值
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	0xX000_X000
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_C0C0
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_X0X0
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	0x0000_0X00
GPIOE_OFFD	GPIO_BA+0x104	R/W	GPIO 端口 E 关闭数字通路寄存器	0x0000_0000
GPIOE_DOUT	GPIO_BA+0x108	R/W	GPIO 端口 E 数据输出寄存器	0x0000_0020
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_00X0
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	0x000X_XX0X
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_01F3
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_0XXX
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

寄存器	偏移量	R/W	描述	复位值
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=0~3, 6~11, 14, 15	GPIO_BA+0x280 + 0x04 * n	R/W	GPIO PC.n端口数据输入/输出	0x0000_000X
PDn_PDIO n=6, 7, 14, 15	GPIO_BA+0x2C0 + 0x04 * n	R/W	GPIO PD.n端口数据输入/输出	0x0000_000X
PEn_PDIO n=5	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, 4, 8	GPIO_BA+0x340 + 0x04 * n	R/W	GPIO PF.n端口数据输入/输出	0x0000_000X

6.5.6 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	偏移量	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	0xFFFF_FFFF
GPIOB_PMD	GPIO_BA+0x040	R/W	GPIO 端口 B 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOC_PMD	GPIO_BA+0x080	R/W	GPIO 端口 C 管脚 I/O 模式控制	0xFFXX_X0XX
GPIOD_PMD	GPIO_BA+0x0C0	R/W	GPIO 端口 D 管脚 I/O 模式控制	0xFF00_0X00
GPIOE_PMD	GPIO_BA+0x100	R/W	GPIO 端口 E 管脚 I/O 模式控制	0x0000_0X00
GPIOF_PMD	GPIO_BA+0x140	R/W	GPIO 端口 F 管脚 I/O 模式控制	0x000X_XX0X

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	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1: n = 0~15(GPIOA/GPIOB) n = 0~3, 6~11, 14, 15 (GPIOC) n = 6, 7, 14, 15(GPIOD) n = 5(GPIOE) n = 0, 1, 4~8 (GPIOF) 注2: 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	偏移量	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
Reserved							
7	6	5	4	3	2	1	0
Reserved							

位	描述	
[n+16] n=0,1..15	OFFD	<p>GPIOx Pin[n] 关闭数字输入通道使能</p> <p>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</p> <p>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)</p> <p>注：</p> <p>n = 0~15 (GPIOA/GPIOB) n = 0~3, 6~11, 14, 15 (GPIOC) n = 6, 7, 14, 15 (GPIOD) n = 5 (GPIOE) n = 0, 1, 4~8 (GPIOF)</p>
[15:0]	Reserved	保留

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

寄存器	偏移量	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_CFCF
GPIOD_DOUT	GPIO_BA+0x0C8	R/W	GPIO 端口 D 数据输出值	0x0000_C0C0
GPIOE_DOUT	GPIO_BA+0x108	R/W	GPIO 端口 E 数据输出值	0x0000_0020
GPIOF_DOUT	GPIO_BA+0x148	R/W	GPIO 端口 F 数据输出值	0x0000_01F3

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

位	描述	
[31:16]	Reserved	保留
[n] n = 0,1..15	DOUT[n]	<p>GPIOx Pin[n] 输出值</p> <p>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.</p> <p>1 = GPIO配置成输出, 开漏和准双向模式时, GPIO port [A/B/C/D/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/F] Pin[n] 为低</p> <p>注:</p> <p>n = 0~15 (GPIOA/GPIOB)</p> <p>n = 0~3, 6~11, 14, 15 (GPIOC)</p> <p>n = 6, 7, 14, 15(GPIOD)</p> <p>n = 5 (GPIOE)</p> <p>n = 0, 1, 4~8 (GPIOF)</p>

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

寄存器	偏移量	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
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
DMASK							
7	6	5	4	3	2	1	0
DMASK							

位	描述	
[31:16]	保留	保留
[n] n = 0,1..15	DMASK[n]	<p>端口 [A/B/C/D/F] 数据输出写屏蔽</p> <p>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。</p> <p>1 = 保护相应的GPIO_DOUT[n] 位 0 = 相应的GPIO_DOUT[n] 位可以被更新</p> <p>注1: 此功能只保护GPIOx_DOUT[n]相应位，不能保护相应位控制寄存器 (PAn_PDIO, PBn_PDIO, PCn_PDIO, PDn_PDIO, PEn_PDIO and PFn_PDIO).</p> <p>注2: n = 0~15 (GPIOA/GPIOB) n = 0~3, 6~11, 14, 15 (GPIOC) n = 6, 7, 14, 15(GPIOD) n = 5 (GPIOE) n = 0, 1, 4~8 (GPIOF)</p>

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

寄存器	偏移量	R/W	描述	复位值
GPIOA_PIN	GPIO_BA+0x010	R	GPIO Port A Pin Value	0x0000_XXXX
GPIOB_PIN	GPIO_BA+0x050	R	GPIO Port B Pin Value	0x0000_XXXX
GPIOC_PIN	GPIO_BA+0x090	R	GPIO Port C Pin Value	0x0000_XXXX
GPIOD_PIN	GPIO_BA+0x0D0	R	GPIO Port D Pin Value	0x0000_X0X0
GPIOE_PIN	GPIO_BA+0x110	R	GPIO Port E Pin Value	0x0000_00X0
GPIOF_PIN	GPIO_BA+0x150	R	GPIO Port F Pin Value	0x0000_0XXX

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

位	描述	
[31:16]	保留	保留
[n] n = 0,1..15	PIN[n]	<p>端口 [A/B/C/D/E/F]管脚数据</p> <p>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。</p> <p>注：</p> <p>n = 0~15 (GPIOA/GPIOB)</p> <p>n = 0~3, 6~11, 14, 15 (GPIOC)</p> <p>n = 6, 7, 14, 15 (GPIOD)</p> <p>n = 5 (GPIOE)</p> <p>n = 0, 1, 4~8 (GPIOF)</p>

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

寄存器	偏移量	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+0xD4	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
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
DBEN							
7	6	5	4	3	2	1	0
DBEN							

位	描述	
[31:16]	保留	保留
[n] n = 0,1..15	DBEN[n]	<p>端口 [A/B/C/D/E/F] 输入信号去抖使能</p> <p>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。</p> <p>1 = 使能 bit[n] 去抖动功能 0 = 禁用 bit[n] 去抖动功能</p> <p>注：</p> <p>n = 0~15 (GPIOA/GPIOB) n = 0~3, 6~11, 14, 15 (GPIOC) n = 6, 7, 14, 15 (GPIOD) n = 5 (GPIOE) n = 0, 1, 4~8 (GPIOF)</p>



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

寄存器	偏移量	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							
7	6	5	4	3	2	1	0
IMD							

位	描述	
[31:16]	保留	保留
[n] n = 0,1..15	IMD[n]	<p>端口 [A/B/C/D/E/F] 边沿或电平检测中断控制</p> <p>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。</p> <p>0 = 边沿触发中断 1 = 电平触发中断</p> <p>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</p> <p>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</p> <p>注：</p> <p>n = 0~15 (GPIOA/GPIOB) n = 0~3, 6~11, 14, 15 (GPIOC) n = 6, 7, 14, 15 (GPIOD) n = 5 (GPIOE) n = 0, 1, 4~8 (GPIOF)</p>

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

寄存器	偏移量	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							
23	22	21	20	19	18	17	16
IR_EN							
15	14	13	12	11	10	9	8
IF_EN							
7	6	5	4	3	2	1	0
IF_EN							

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： n = 0~15 (GPIOA/GPIOB) n = 0~3, 6~11, 14, 15 (GPIOC) n = 6, 7, 14, 15 (GPIOD) n = 5 (GPIOE) n = 0, 1, 4~8 (GPIOF)
[n] n = 0,1..15	IF_EN[n] 端口 [A/B/C/D/E/F] 输入下降沿或输入低电平中断使能 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’：

	<p>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。</p> <p>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。</p> <p>1 = 使能PIN[n]低电平或由高电平到低电平变化的中断</p> <p>0 = 禁用PIN[n]低电平或由高电平到低电平变化的中断</p> <p>注：</p> <p>n = 0~15 (GPIOA/GPIOB)</p> <p>n = 0~3, 6~11, 14, 15 (GPIOC)</p> <p>n = 6, 7, 14, 15 (GPIOD)</p> <p>n = 5 (GPIOE)</p> <p>n = 0, 1, 4~8 (GPIOF)</p>
--	--

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

寄存器	偏移量	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							
7	6	5	4	3	2	1	0
ISRC							

位	描述	
[31:16]	保留	保留
[n] n = 0,1..15	ISRC[n]	<p>端口 [A/B/C/D/E/F] 中断触发源标志</p> <p>读：</p> <p>1 = GPIOx[n]产生中断</p> <p>0 = GPIOx[n]没有中断</p> <p>写：</p> <p>1= 清相应的未处理中断标志</p> <p>0= 无动作</p> <p>.</p> <p>注:</p> <p>n = 0~15 (GPIOA/GPIOB)</p> <p>n = 0~3, 6~11, 14, 15 (GPIOC)</p> <p>n = 6, 7, 14, 15 (GPIOD)</p> <p>n = 5 (GPIOE)</p> <p>n = 0, 1, 4~8 (GPIOF)</p>



中断去抖动周期控制(DBNCECON)

寄存器	偏移量	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 <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>DBCLKSEL</th> <th>描述</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>采样中断输入信号, 每 1 clocks一次</td> </tr> <tr> <td>1</td> <td>采样中断输入信号, 每 2 clocks一次</td> </tr> <tr> <td>2</td> <td>采样中断输入信号, 每 4 clocks一次</td> </tr> <tr> <td>3</td> <td>采样中断输入信号, 每 8 clocks一次</td> </tr> <tr> <td>4</td> <td>采样中断输入信号, 每 16 clocks一次</td> </tr> <tr> <td>5</td> <td>采样中断输入信号, 每 32 clocks一次</td> </tr> <tr> <td>6</td> <td>采样中断输入信号, 每 64 clocks一次</td> </tr> <tr> <td>7</td> <td>采样中断输入信号, 每 128 clocks一次</td> </tr> <tr> <td>8</td> <td>采样中断输入信号, 每 256 clocks一次</td> </tr> <tr> <td>9</td> <td>采样中断输入信号, 每 2*256 clocks一次</td> </tr> <tr> <td>10</td> <td>采样中断输入信号, 每 4*256 clocks一次</td> </tr> </tbody> </table>	DBCLKSEL	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
DBCLKSEL	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	偏移量	R/W	描述	复位值
PA _n _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
PB _n _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
PC _n _PDIO $n = 0~3, 6~11,$ $14, 15$	GPIO_BA+0x280 + 0x04 * n	R/W	GPIO PC. _n 管脚数据输入/输出	0x0000_000X
PD _n _PDIO $n = 6, 7, 14, 15$	GPIO_BA+0x2C0 + 0x04 * n	R/W	GPIO PD. _n 管脚数据输入/输出	0x0000_000X
PE _n _PDIO $n = 5$	GPIO_BA+0x300 + 0x04 * n	R/W	GPIO PE. _n 管脚数据输入/输出	0x0000_000X
PF _n _PDIO $n = 0, 1, 4~8$	GPIO_BA+0x340 + 0x04 * n	R/W	GPIO PF. _n 管脚数据输入/输出	0x0000_000X

Note: x = A/B/C/D/E/F

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							Px _n _PDIO

位	描述	
[0]	Px _n _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



6.6 定时器控制器(TIMER)

6.6.1 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，时钟发生，外部输入管脚事件计数和外部捕捉管脚脉宽测量等。

6.6.2 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 四种计数操作模式
- 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 = $(1 / T \text{ MHz}) * (2^8) * (2^{24})$, T 是定时器周期
- 通过TDR (TDRx[23:0]) (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

6.6.3 框图

定时器控制器的框图和时钟控制如下所示：

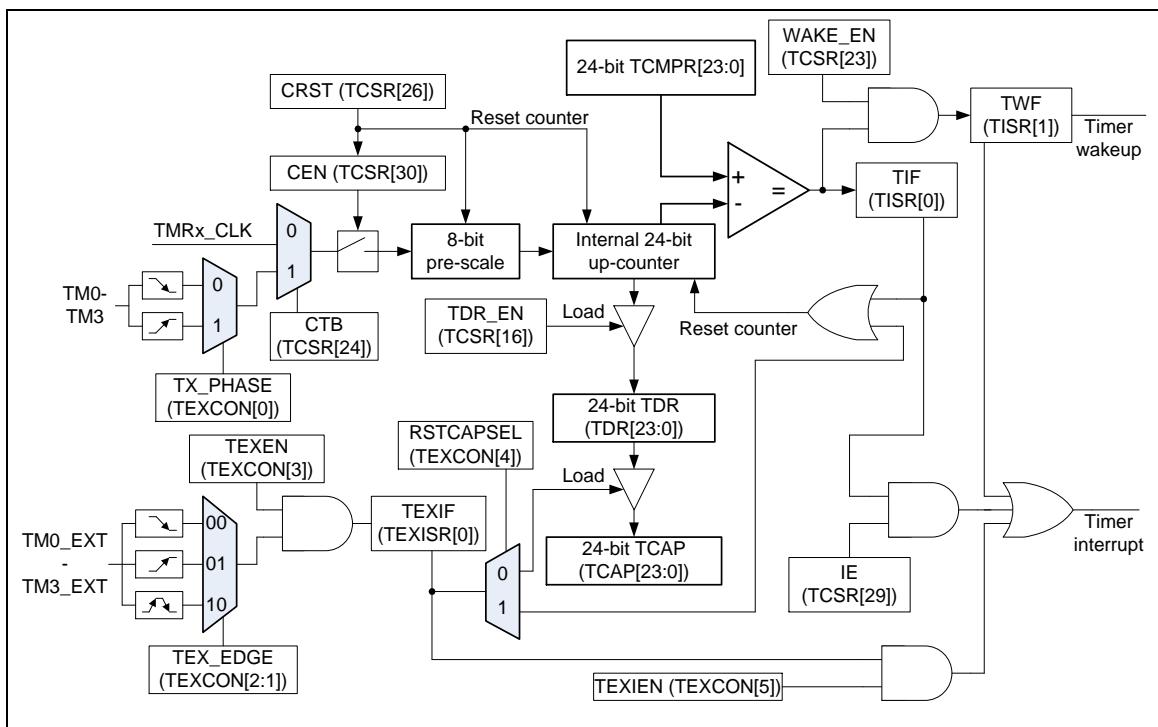


图 6-20 定时器控制器框图

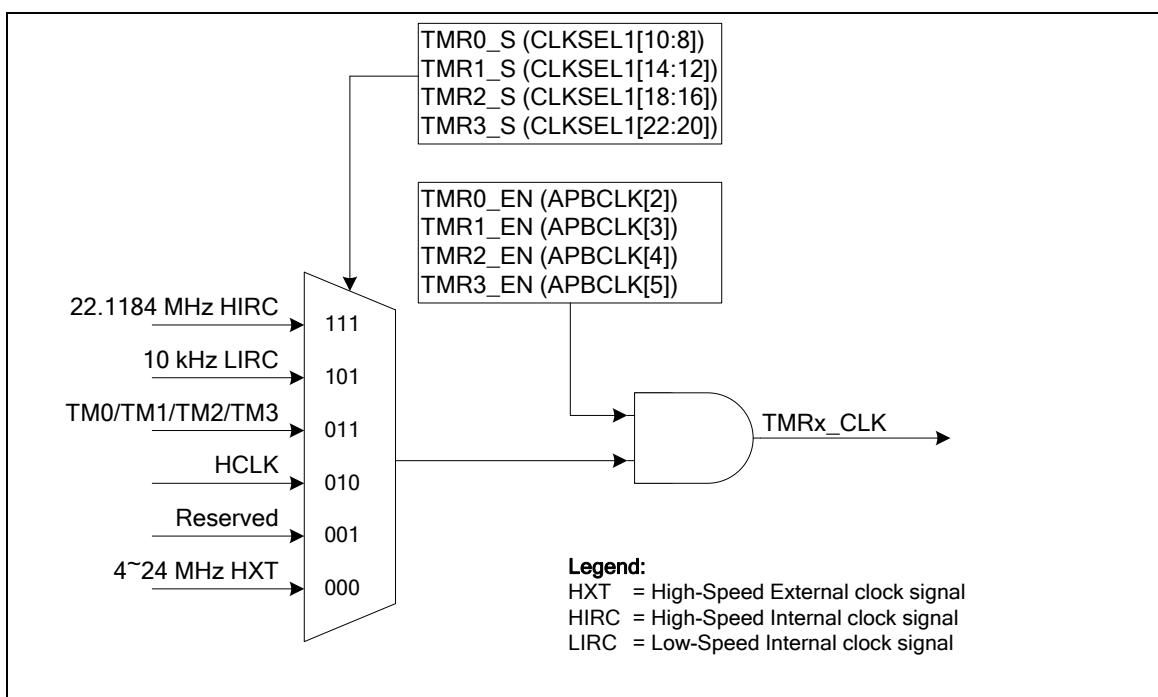


图 6-21 定时器控制器时钟源



6.6.4 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。

6.6.5 功能描述

6.6.5.1 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。

6.6.5.2 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。

6.6.5.3 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TCMP的值比较，直到 CEN位由软件清0。

6.6.5.4 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置1，则定时器的计数器开始计数。toggle-out 模式的计数操作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。

6.6.5.5 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， $81, 82, 83, \dots$ to $(2^{24}-1)$ ，然后再一次 $0, 1, 2, 3, \dots$ 到 $2^{24}-1$ ，如此往复。接下来，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。

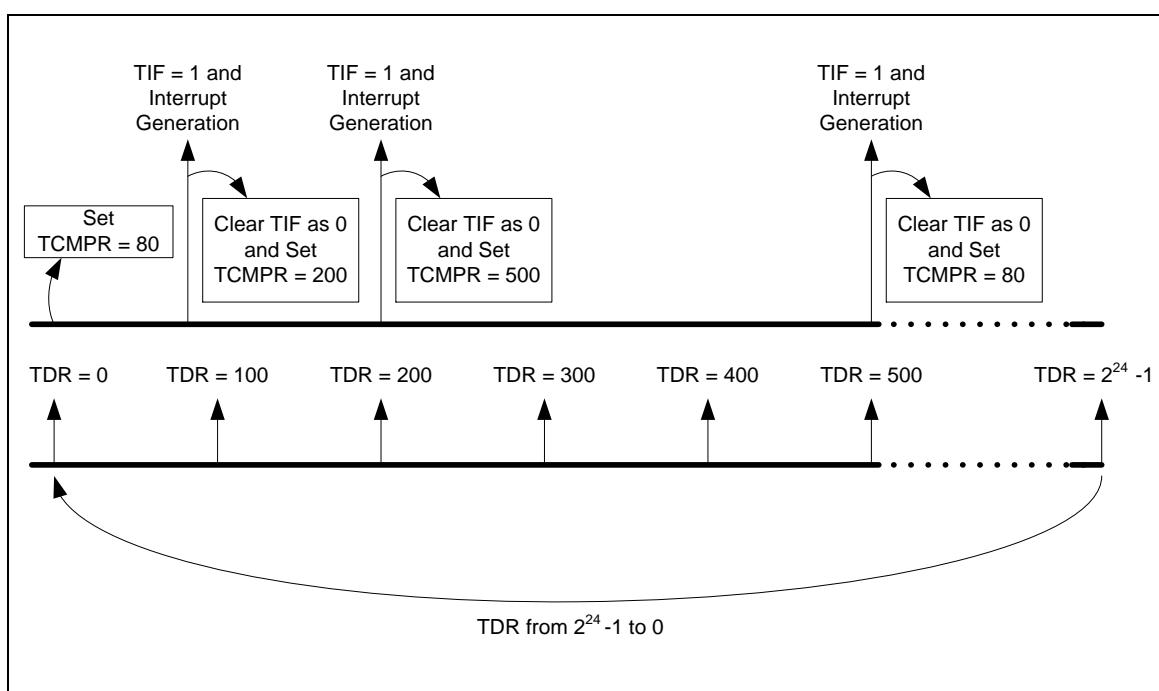


图 6-22 连续计数模式



6.6.5.6 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 TM_x 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。该功能下，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TM_x管脚消抖电路。如果 TM_x 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 TM_x 管脚的相位。

事件计数模式下，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TM_x管脚的输入事件TDR的值。

6.6.5.7 外部捕捉模式

事件捕捉功能是当检测到 TM_x_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)。在该模式下，需把 RSTCAPSEL(TEXCON[4]) 位设置为0，用来选择 TM_x_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 TM_x_EXT 管脚消抖电路。在 TM_x_EXT 的消抖电路关闭时，TM_x_EXT 管脚的转变频率必须少于 1/3 HCLK，在 TM_x_EXT 的消抖电路打开时，TM_x_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 TM_x_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 TM_x_EXT 管脚有边沿变化时。

6.6.5.8 事件复位计数模式

当检测到 TM_x_EXT 管脚(x=0~3) 有边沿变化时，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 TM_x_EXT 转变时用作为事件复位计数器。



6.6.6 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



6.6.7 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

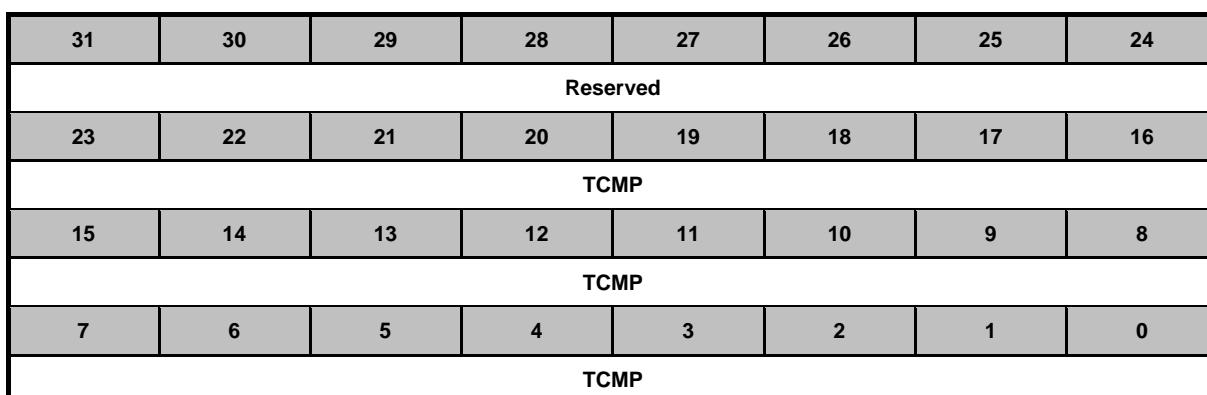
31	30	29	28	27	26	25	24
DBGACK_TMR	CEN	IE	MODE		CRST	CACT	CTB
23	22	21	20	19	18	17	16
WAKE_EN	Reserved			TRG_PWM_EN	TRG_SRC_SEL	Reserved	TDR_EN
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PRESCALE							

位	描述	
[31]	DBGACK_TMR	<p>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。</p>
[30]	CEN	<p>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。</p>
[29]	IE	<p>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TIF (TISR[0]) 置 1， 定时器中断信号产生并通知CPU。</p>

[28:27]	MODE	定时器操作模式 00 = 定时器工作在单触发模式 (one-shot mode)) 01 = 定时器工作在周期模式 (period mode)) 10 = 定时器工作在触发输出模式(Toggle-output mode) 11 = 定时器工作在连续计数模式(Continuous Counting mode)
[26]	CRST	定时器复位 0 = 该位写 0 无效 1 = 如果CACT为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
[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。具体描述请参考6.6.5.6章节 0 = 禁用计数器模式 1 = 使能计数器模式
[23]	WAKE_EN	唤醒使能位 0 = 唤醒触发事件禁止 1 = 唤醒触发事件使能
[22:20]	Reserved	保留.
[19]	TRG_PWM_EN	触发PWM 使能位 如果该位置为1，定时器溢出中断或捕捉中断可以用于触发PWM。 0 = 定时器中断触发PWM禁止。 1 = 如果TRG_SRC_SEL (TCSR[18]) = 0, 定时器溢出中断信号将触发PWM。 如果TRG_SRC_SEL (TCSR[18]) = 1, 捕捉中断信号将触发PWM。
[18]	TRG_SRC_SEL	触发源选择位 该位用于选择触发源是从定时溢出中断信号还是捕捉中断信号。 0 = 选择定时器溢出中断信号触发PWM。 1 = 选择捕捉中断信号触发PWM。
[17]	Reserved	保留.
[16]	TDR_EN	数据加载使能位 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



位	描述	
[31:24]	Reserved	保留
[23:0]	TCMP	<p>定时器比较值</p> <p>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标志将被置1。</p> <p>超时周期 = (定时器输入时钟周期) * (8-bit PRESCALE + 1) * (24-bit TCMP)</p> <p>注意1：不能向 TCMP 里写 0x0 或 0x1，否则内核将运行到未知状态。</p> <p>注意2：当定时器工作在 continuous counting 模式，即使软件写一个新的值到 TCMP，24位向上计数定时器将保持继续计数。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。</p>



定时器中断状态寄存器(TISR)

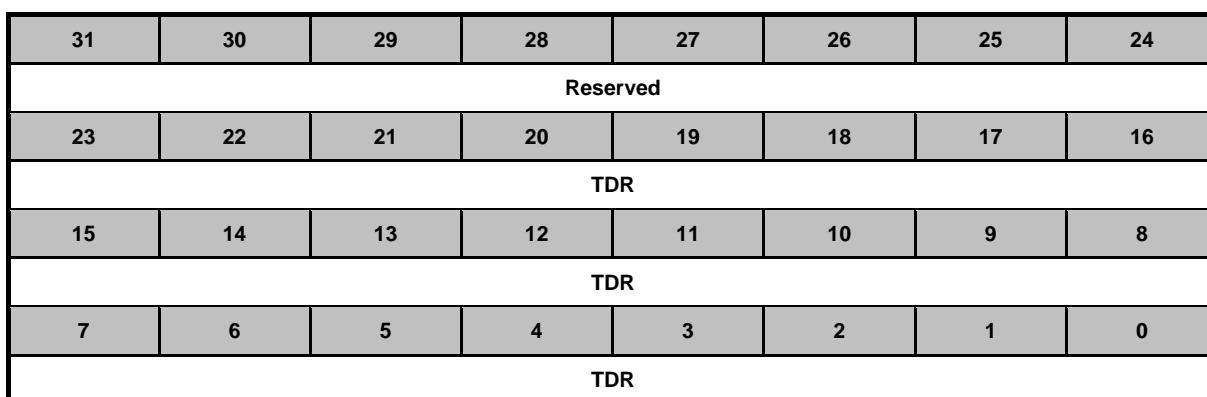
寄存器	偏移地址	R/W	描述	复位值
TISR0	TMR01_BA+0x08	R/W	Timer0中断状态寄存器	0x0000_0000
TISR1	TMR01_BA+0x28	R/W	Timer1中断状态寄存器	0x0000_0000
TISR2	TMR23_BA+0x08	R/W	Timer2中断状态寄存器	0x0000_0000
TISR3	TMR23_BA+0x28	R/W	Timer3中断状态寄存器	0x0000_0000

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	<p>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.</p>
[0]	TIF	<p>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。</p>

定时器数据寄存器(TDR)

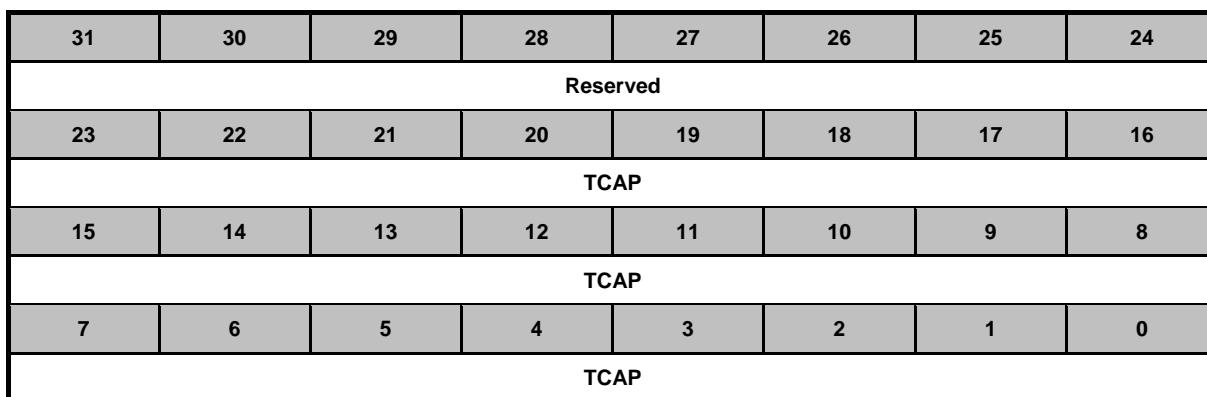
寄存器	偏移地址	R/W	描述	复位值
TDR0	TMR01_BA+0x0C	R	Timer0数据寄存器	0x0000_0000
TDR1	TMR01_BA+0x2C	R	Timer1数据寄存器	0x0000_0000
TDR2	TMR23_BA+0x0C	R	Timer2数据寄存器	0x0000_0000
TDR3	TMR23_BA+0x2C	R	Timer3数据寄存器	0x0000_0000



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) 会不断被更新来监视内部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



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	<p>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。</p>
[6]	TEXDB	<p>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。</p>
[5]	TEXIEN	<p>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。</p>
[4]	RSTCAPSEL	<p>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)</p>



[3]	TEXEN	定时器外部管脚使能位 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 = Tx_EXT (x= 0~3)管脚上 下降沿将被检测。 01 = Tx_EXT (x= 0~3)管脚上 上升沿将被检测。 10 = Tx_EXT (x= 0~3)管脚上 上升沿或下降沿将被检测。 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	<p>定时器外部中断标志位 该位表示外部捕捉中断标志状态。.</p> <p>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，该标志将被硬件置位。</p> <p>0 = TMx_EXT管脚中断没有发生。 1 = TMx_EXT管脚中断发生。</p> <p>注意：该位写 1 清零。</p>



6.7 PWM发生器和捕捉时钟(PWM)

6.7.1 概述

NUC131提供了两路PWM发生器-PWM0和PWM1图 6-23，每路PWM支持6通道PWM输出或输入捕捉。12位的预分频器可以为16位比较器和计数器灵活配置时钟。PWM计数器支持向上，向下，上下计数方式。PWM用比较器和计数器的比较来产生事件，这些事件用来产生PWM脉冲，中断，ADC转换的触发信号。

PWM发生器支持两个标准PWM输出模式：独立模式和互补模式，它们是不同的架构。互补模式，两个比较器产生各种带12位死区时间的PWM脉冲。PWM输出控制单元，支持极性输出，独立管脚屏蔽，三态输出使能和刹车功能。

PWM也支持输入捕捉功能，当输入信道有向上跳变、向下跳变、或者两者都有的跳变时，捕捉到的PWM计数值将锁存到相应的寄存器中。

6.7.2 特性

6.7.2.1 PWM功能特性

- 支持时钟频率最高达100MHz。
- 支持两个PWM模块，每个模块提供6个输出通道。
- 支持独立模式的PWM输出/输入捕捉。
- 支持3组互补信道的互补模式。
 - 12位分辨率的死区插入
 - 每个周期两个比较值。
- 支持12位从1到4096的预分频
- 支持16位分辨率的PWM计数器，每个模块提供3个PWM计数器。
 - 向上，向下和上下计数操作类型。
- 支持PWM管脚屏蔽功能和三态使能。
- 支持刹车功能
 - 刹车源来自管脚和系统安全事件（时钟故障，欠压监测和CPU锁住）
 - 刹车源管脚噪声滤波器。
 - 通过边缘检测刹车源来控制刹车状态直到刹车中断清除。
 - 刹车条件解除后电平检测刹车源自动恢复功能。
- 支持下列事件中断：
 - PWM计数器值为0、周期值或比较值。
 - 发生刹车条件
- 支持下列事件触发ADC
 - PWM计数器值为0、周期值或比较值。



6.7.2.2 捕捉功能特性

- 支持12个16位分辨率的输入捕捉信道。
- 支持上升/下降沿捕捉条件。
- 支持输入上升/下降沿 捕捉中断。
- 支持计数器重载选项的上升/下降沿 捕捉

6.7.2.3 PWM & BPWM 特性比较表

特性	PWM	BPWM
计数器数目	2个通道共享1个定时器,一共6个定时器	6个通道共享1个定时器,一共1个定时器
互补模式	V	X
停滞时间功能	V	X
刹车功能	V	X
捕获加载	2个通道加载一个定时器	6个通道加载一个定时器

表 6-9 PWM & BPWM 特性比较表

6.7.3 框图

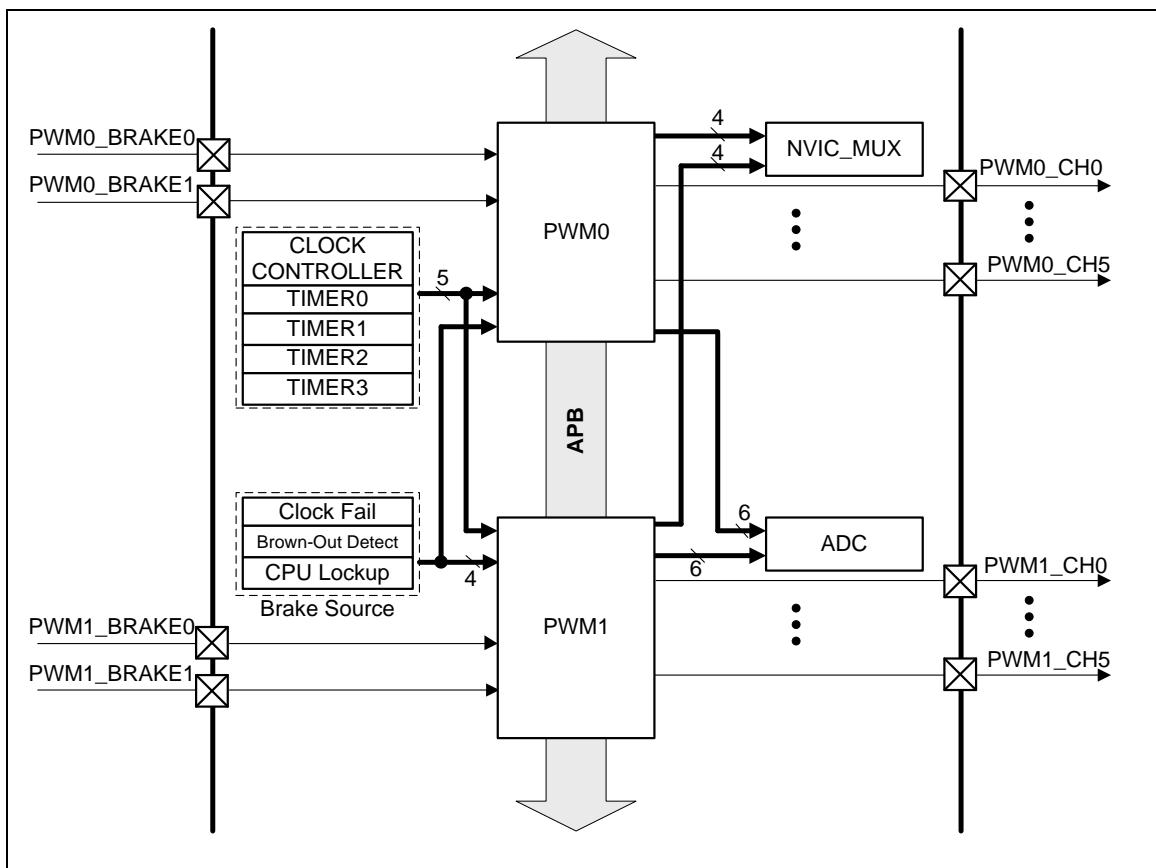


图 6-23 PWM 发生器概述框图

PWM系统时钟可以设为等于或两倍的HCLK频率，如图 6-24，寄存器设置细节请参考表 6-10。

每个PWM发生器有三个输入时钟源，每个时钟源可以从系统时钟或者4组定时器触发PWM输出，如图 6-25 PWM_CLK0 设置 ECLKSRC0 (PWM_CLKSRC[2:0])， PWM_CLK2 设置 ECLKSRC2 (PWM_CLKSRC[10:8])， PWM_CLK4设置ECLKSRC4 (PWM_CLKSRC[18:16])。

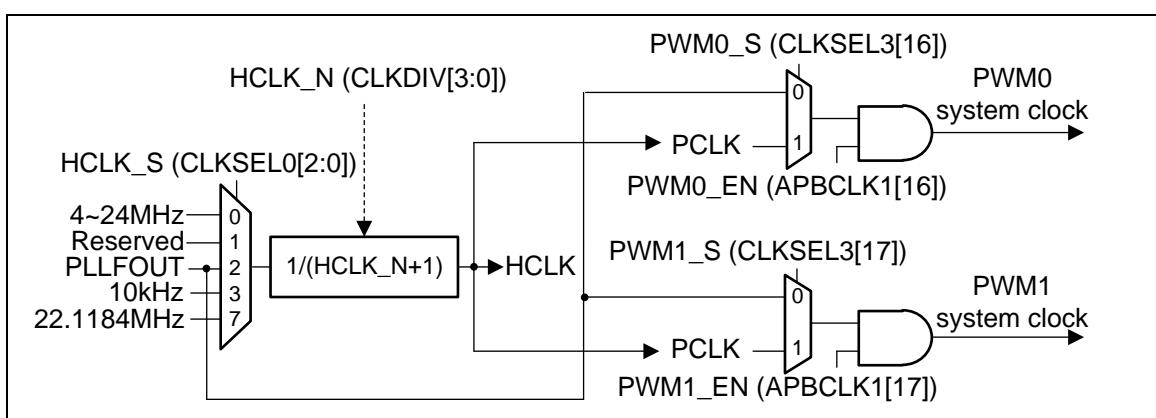


图 6-24 PWM 系统时钟源控制

PWM 系统时钟/HCLK 频率 比	HCLK_S (CLKSEL0[2:0])	HCLK_N (CLKDIV[3:0])	PWMn_S (CLKSEL3[X]), (N, X) 表示 (0, 16) 或 (1, 17)
1/1	无效	无效	1
2/1	2	1	0

表 6-10 PWM系统时钟源控制寄存器设置表

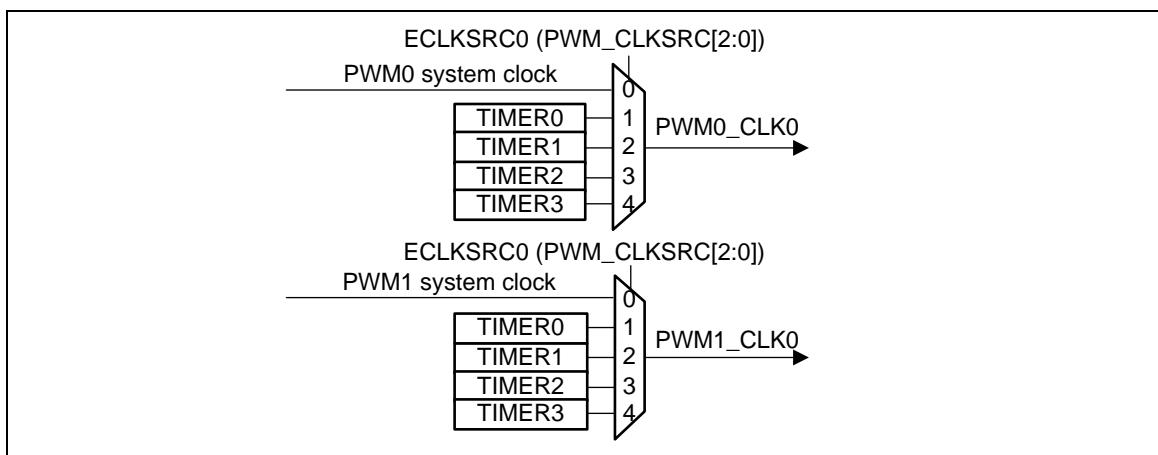


图 6-25 PWM 时钟源控制

图 6-26和图 6-27表示PWM独立模式和互补模式的架构。不管独立模式还是比较模式，相配信道组(PWM_CH0 和 PWM_CH1, PWM_CH2 和 PWM_CH3, PWM_CH4 和 PWM_CH5)共享相同计数器。当计数器的值等于0, PERIOD (PWM_PERIODn[15:0])或比较器, 将产生事件。这些事件通过相应的发生器来产生PWM脉冲、中断信号、ADC的触发信号。输出控制是用来改变PWM脉冲输出状态的。输出控制的刹车功能也能产生中断事件。在互补模式偶数信道用奇数信道比较器产生事件。

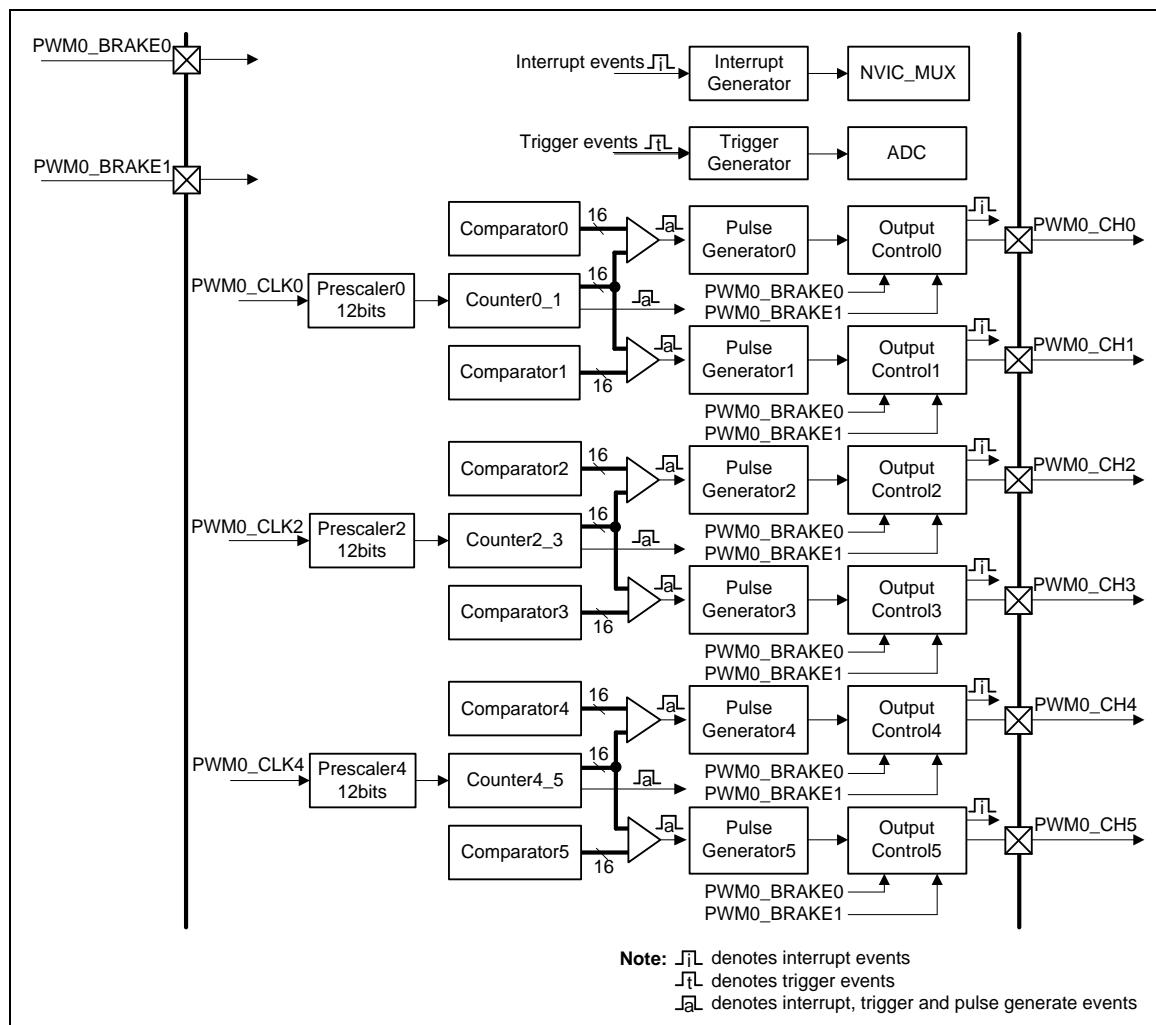


图 6-26 PWM 独立模式架构图

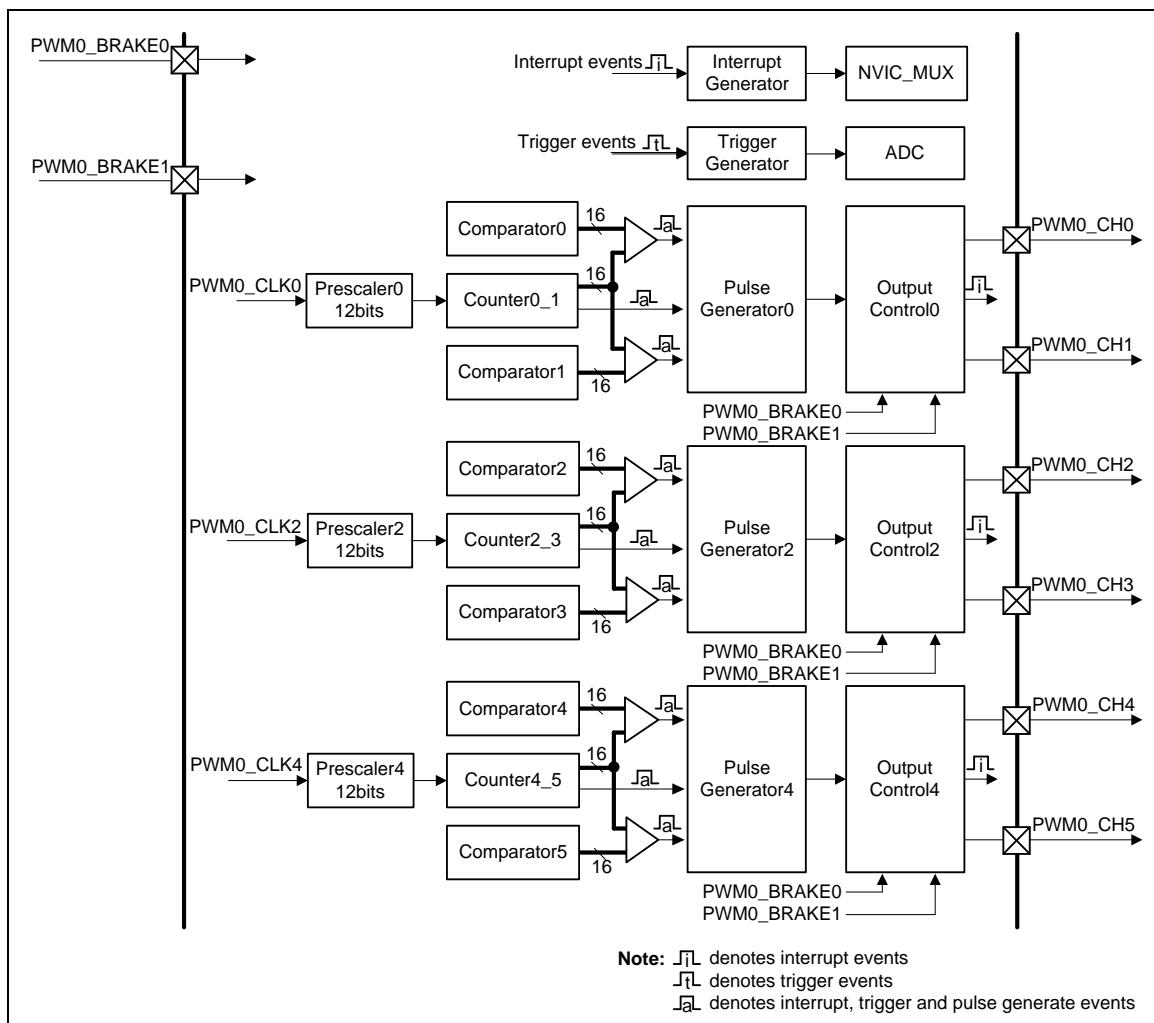


图 6-27 PWM 比较模式架构图

6.7.4 基本配置

PWM 管脚功能通过 GPA_MFP 寄存器配置， PWM_BRAKE0 和 PWM_BRAKE1 的管脚功能通过 GPB_MFP 和 GPC_MFP 寄存器配置。

PWM 时钟通过 APBCLK1[17:16] 使能。 PWM 时钟源通过 CLKSEL3[17:16] 选择。

6.7.5 功能描述

6.7.5.1 PWM 预分频器

PWM 预分频器是用于时钟源除频，预分频器计数 CLK_PSC +1 次，PWM 计数器只加一次。 CLK_PSC（预分频时钟寄存器）通过 CLK_PSC (PWM_CLK_PSCn[11:0], n 表示 0, 2, 4) 来设置。图 6-28 为 PWM 通道 0 CLK_PSC 波形图。

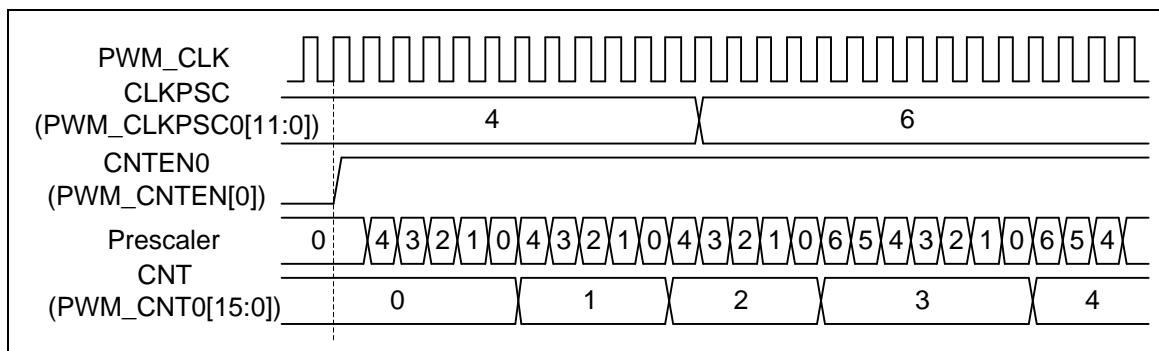


图 6-28 PWM_CH0 CLKPSC 波形

6.7.5.2 PWM计数器

PWM支持3种计数方式操作：向上计数，向下计数，上下计数方式。

6.7.5.3 向上计数方式

在向上计数操作中，16位PWM计数器是一个向上计数器并从0开始向上计数到PERIOD (PWM_PERIODn[15:0]，其中n表示通道数)来完成一个PWM周期。当前计数值可以从CNT (PWM_CNTn[15:0])读出。当计数器计数到0将产生零点事件，当计数到PERIOD将产生周期点事件。如下图显示一个向上计数器例子，PWM周期时间= (PERIOD+1) * PWM 时钟。

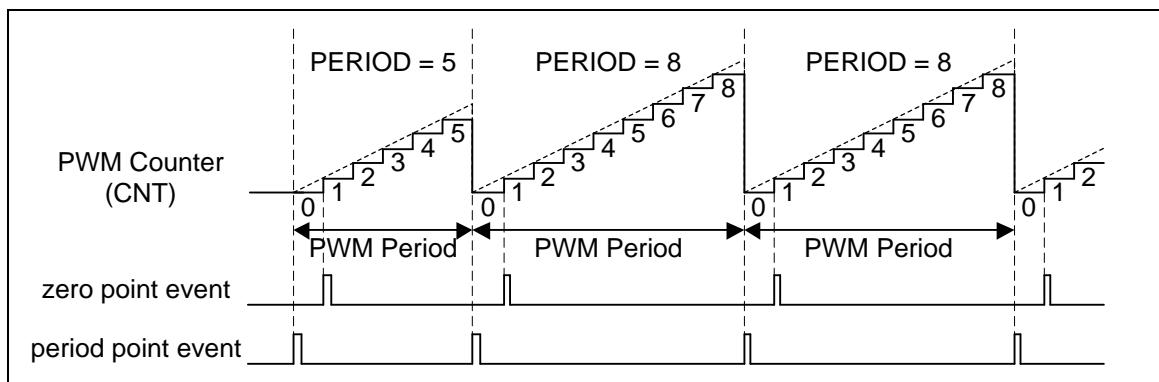


图 6-29 PWM 向上计数方式

6.7.5.4 向下计数方式

在向下计数方式中，16位PWM计数器是一个向下计数器并从PERIOD开始向下计数到0来完成一个PWM周期。当前计数值可以从CNT读出。当计数器计数到0将产生零点事件，当计数到PERIOD将产生周期事件，如下图显示一个向下计数器例子：PWM周期时间= (PERIOD+1) * PWM 时钟。

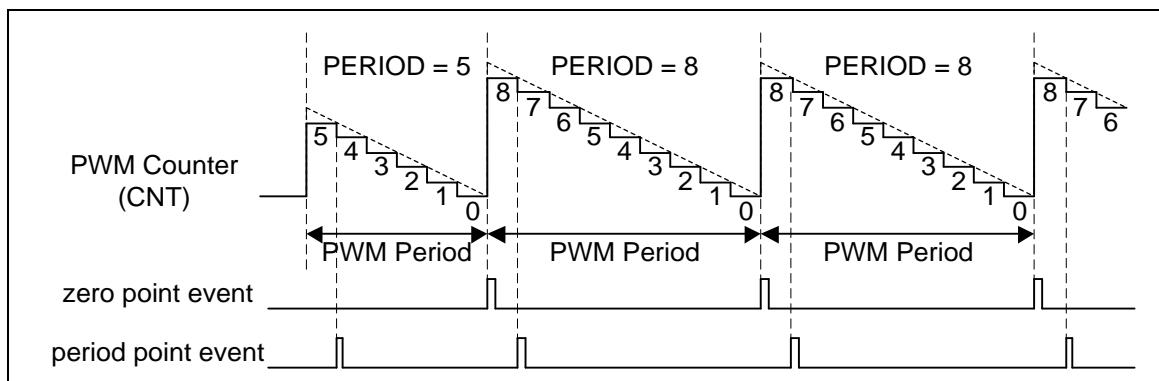


图 6-30 PWM 向下计数方式

6.7.5.5 上下计数方式

在上下计数操作中，16位PWM计数器是一个上下计数器，并开始向上计数0到PERIOD并开始向下计数到0完成一个PWM周期。当前计数值可以从CNT读出。当计数器计数到0将产生零点事件，当计数到PERIOD将产生中心点事件，如下图显示一个上下计数器例子：PWM周期时间= $(2 \times \text{PERIOD}) \times \text{PWM 时钟}$ 。DIRF (PWM_CNTn[16])是计数器方向标志，高是向上计数，低是向下计数。

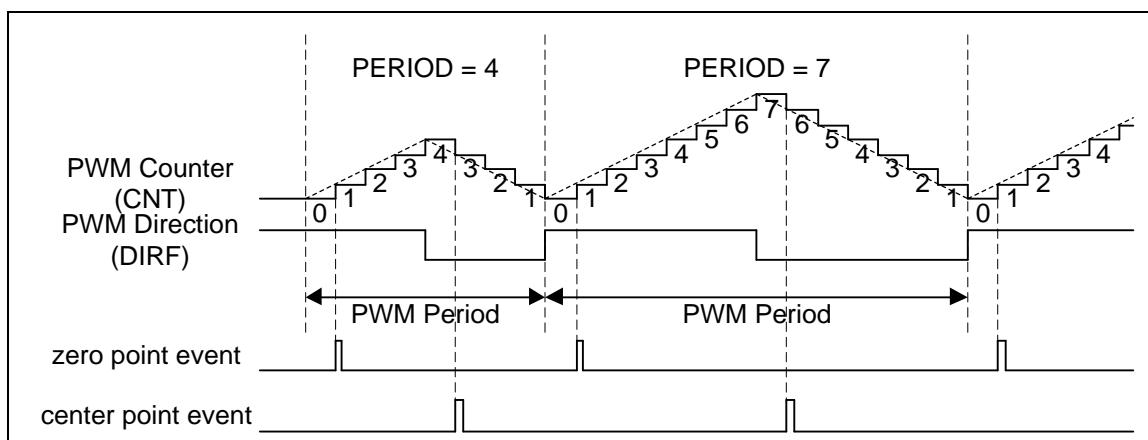


图 6-31 PWM 上下计数方式

6.7.5.6 PWM 比较器

CMPDAT (PWM_CMPDATn[15:0])是PWM信道n的基本比较器寄存器。每个通道只有一个CMPDAT。CMPDAT值一直与相应互补通道的计数器值作比较。当计数器值等于比较寄存器值，PWM产生一个事件，并用事件产生一个PWM脉冲，中断，或者触发ADC。在上下计数方式中，一个PWM周期将产生两个事件，如图 6-32所示。

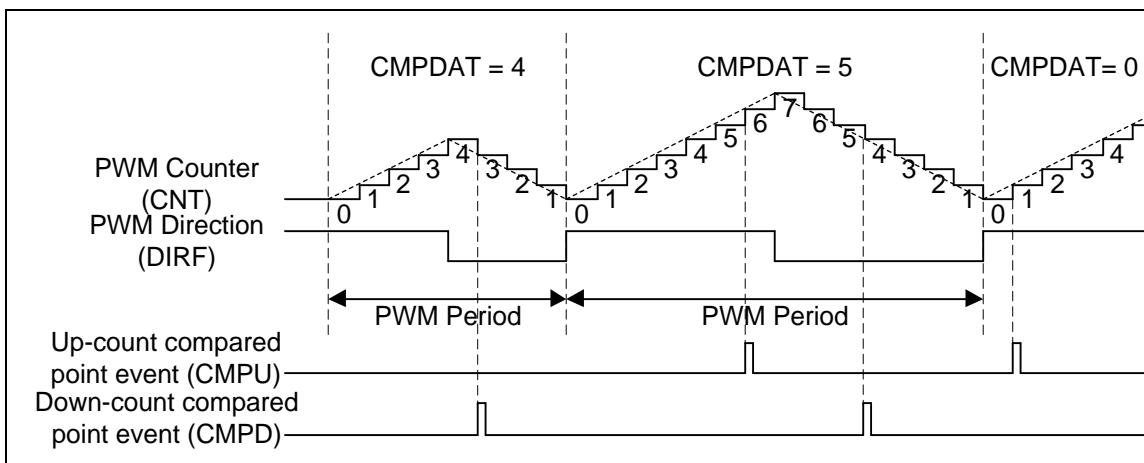


图 6-32 在上下计数方式中的 PWM CMPDAT 事件

6.7.5.7 PWM 双缓存

双缓存是用两个缓存器来分开软件写和硬件操作时序。软件设置好寄存器后，硬件将按照装载模式时序将寄存器值装载到缓存寄存器中。硬件的操作是基于缓存寄存器的值。这样可以避免软硬件不同步时的操作问题。

PWM 的 PERIOD 和 CMPDAT 都具有双缓存功能。在装载模式上应用双缓存的概念如下面描述。例如，图 6-33 所示，在周期装载模式，软件写 PERIOD 和 CMPDAT，在开始下个周期时 PWM 将载入新值到他们的缓存 PBUF (PWM_PBUFn[15:0]) 和 CMPBUF (PWM_CMPBUFn[15:0]) 不影响当前计数操作。载入值到缓存有三种装载模式：周期装载模式，立即装载模式，中心装载模式。

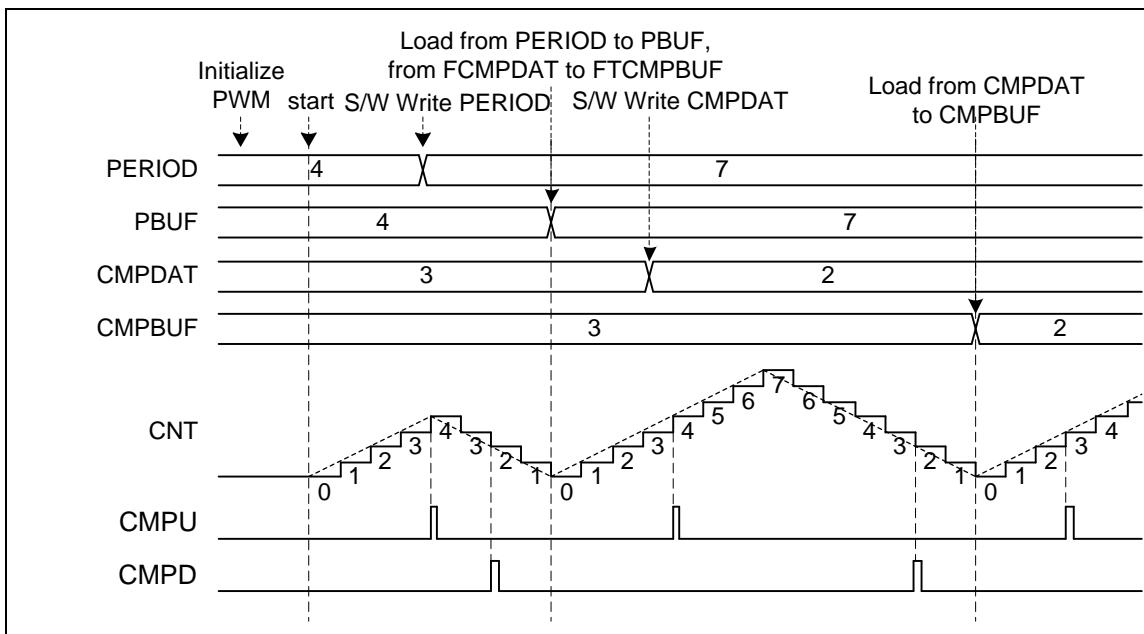


图 6-33 PWM 双缓存图解

6.7.5.8 周期装载模式

周期装载模式是默认模式，在各个装载模式中的优先级最低。当一个周期完成后PERIOD 和 CMPDAT 都装载到他们的缓存。例如：在向上计数操作PWM计数器从零计数到PERIOD后或在向下计数操作从PERIOD计数到零或在上下计数操作向上计数从零到PERIOD然后向下计数到零。

图 6-34表示向上计数操作时的周期装载时序，PERIOD DATA0表示PERIOD初始数据，PERIOD DATA1表示通过软件等所更新的第一个PERIOD 数据。CMPDAT同样遵照这规则。以下是图 6-34的步骤描述。用户可以通过观察PWM周期和CMPU事件知道PERIOD 和 CMPDAT 的更新条件。

1. 点 1，软件写 CMPDAT DATA1 到 CMPDAT
2. 点 2，PWM 周期结束，周期装载 CMPDAT DATA1 到 CMPBUF
3. 点 3，软件写 PERIOD DATA1 到 PERIOD
4. 点 4，PWM 周期结束，周期装载 CMPDAT DATA1 到 PBUF
5. 点 5，软件写 DATA2 到 PERIOD
6. 点 6，PWM 周期结束，周期装载 DATA2 到 PBUF

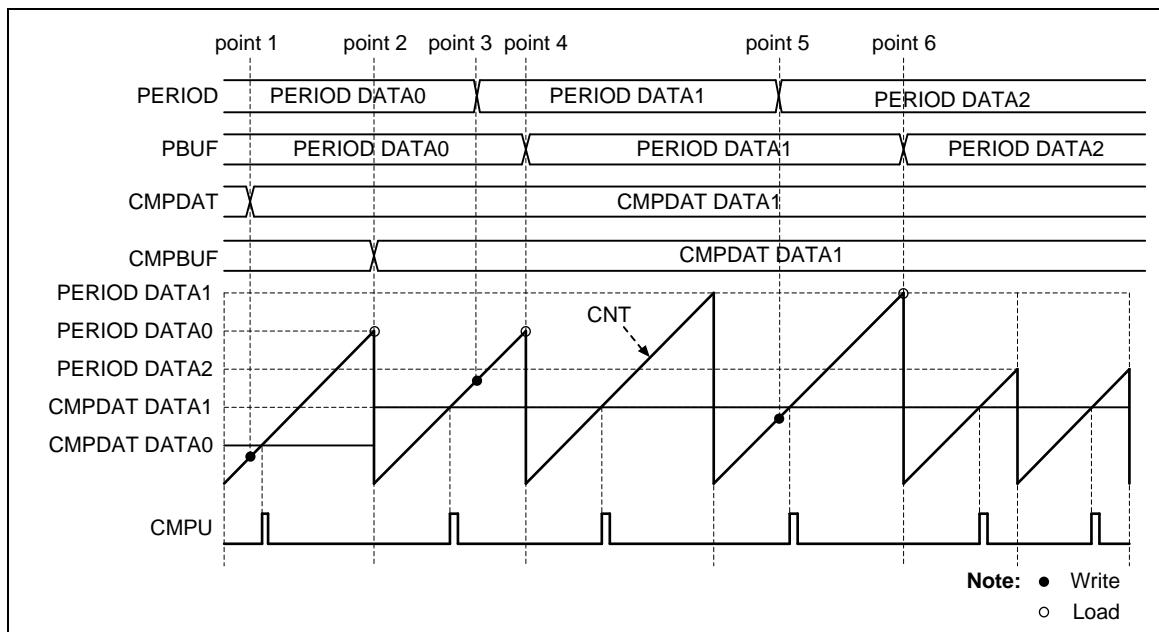


图 6-34 向上计数方式的周期装载模式

6.7.5.9 立即装载模式

如果IMMLDENn (PWM_CTL0[21:16]) 中与 PWM 通道 n 相应的位被置 1 时，当软件更新 PERIOD 或 CMPDAT 后，软件立即将 PERIOD 和 CMPDAT 值装载到缓存。如果 PERIOD 更新值小于当前计数值，计数器将循环计数。立即装载模式有最高优先级，如果设置 IMMLDENn，通道 n 的其他装载模式将失效。图 6-35，例子和步骤描述如下：

1. 软件写 CMPDAT DATA1 并在点 1 硬件立即将 CMPDAT DATA1 装载到 CMPBUF
2. 点 2，软件写入 PERIOD DATA1 值大于当前值，计数器将继续计数到 PERIOD DATA1 来完成装载。
3. 点 3，软件写入 PERIOD DATA2 值小于当前值，计数器将继续计数到最大值 0xFFFF 并循环从 0 开始到 PERIOD DATA2 来完成这个周期装载。

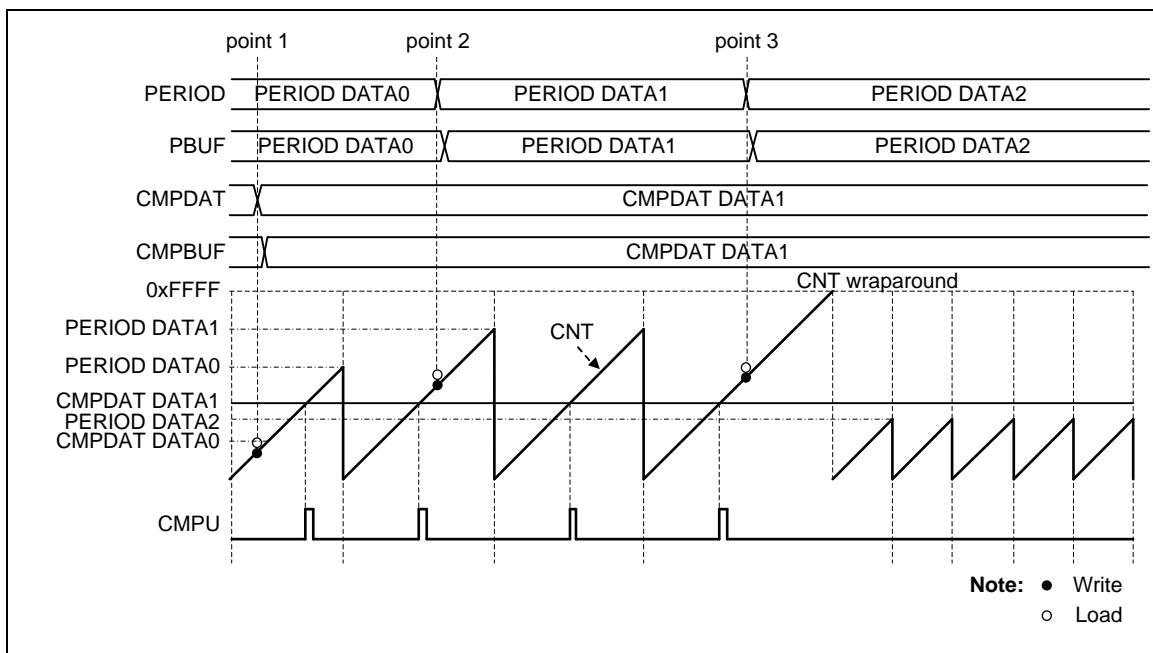


图 6-35 向上计数方式的立即装载模式

6.7.5.10 中心装载模式

如果 **CTRLDn** (PWM_CTL0[5:0]) 的 PWM 通道 n 相应位被置 1 并处于上下计数模式，在周期的中点 CMPDAT 值将装载到 CMPBUFn，也就是说计数器计数到 PERIOD 点。PERIOD 装载时序与周期装载模式一样。如图 6-36，例子和步骤描述如下：

1. 点 1，软件写 CMPDAT DATA1
2. 点 2，PWM 周期中点硬件装载 CMPDAT DATA1 到 CMPBUF
3. 点 3，软件写 PERIOD DATA1
4. 点 4，PWM 周期终点硬件装载 PERIOD DATA1 到 PBUF
5. 点 5，软件写 CMPDAT DATA2
6. 点 6，在 PWM 周期中点硬件装载 CMPDAT DATA2 到 CMPBUF
7. 点 7，软件写 PERIOD DATA2
8. 点 8，在 PWM 周期终点硬件装载 PERIOD DATA2 到 PBUF

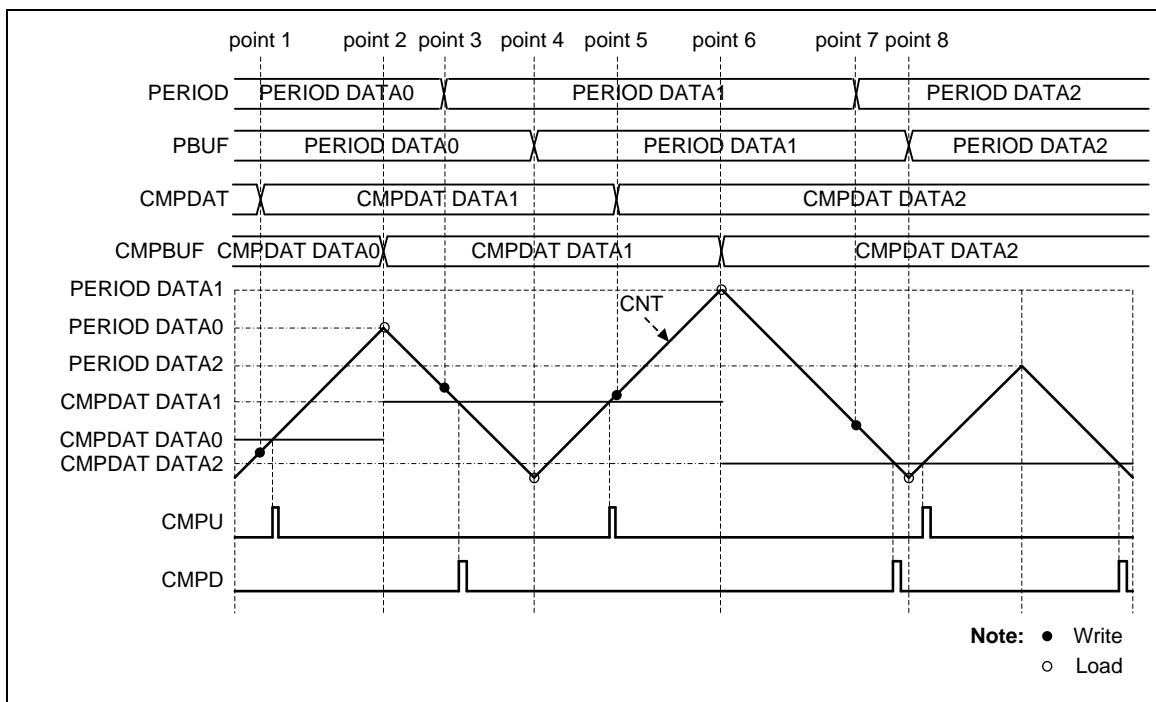


图 6-36 上下计数方式的中心装载模式

6.7.5.11 PWM 脉冲发生器

PWM脉冲发生器是用计数器和比较器事件来产生PWM脉冲。这些事件包括：向上和向下计数方式的零点或周期点，上下计数方式的中点，以及三种方式中的计数器等于比较值。作为上下计数方式，计数器有两个等于比较器的点一个在向上计数，一个在向下计数。此外，互补模式有两个比较器值与计数器比较，因此在上下计数方式中有四个相等值，向上或向下计数方式各有两个。

通过设定 **PWM_WGCTL0** 和 **PWM_WGCTL1**，每个事件点可以决定 PWM 波形：不变 (X)，为低 (L)，为高 (H) 或切换 (T)。用这些点可以很容易产生一个不规则的 PWM 脉冲或可变波形，如图 6-37. 图中，PWM 是互补模式，两个比较器 n 和 m 来产生 PWM 脉冲。n 表示偶数通道 0, 2, 4；m 表示奇数通道 1, 3, 5。n 和 m 通道是互补组。互补模式用两个信道 (CH0 和 CH1, CH2 和 CH3, CH4 和 CH5) 作为一组 PWM 输出一个互补波形。CMPU 表示当向上计数时 CNT 等于 CMPDAT.CMPD 表示当向下计数时 CNT 等于 CMPDAT。

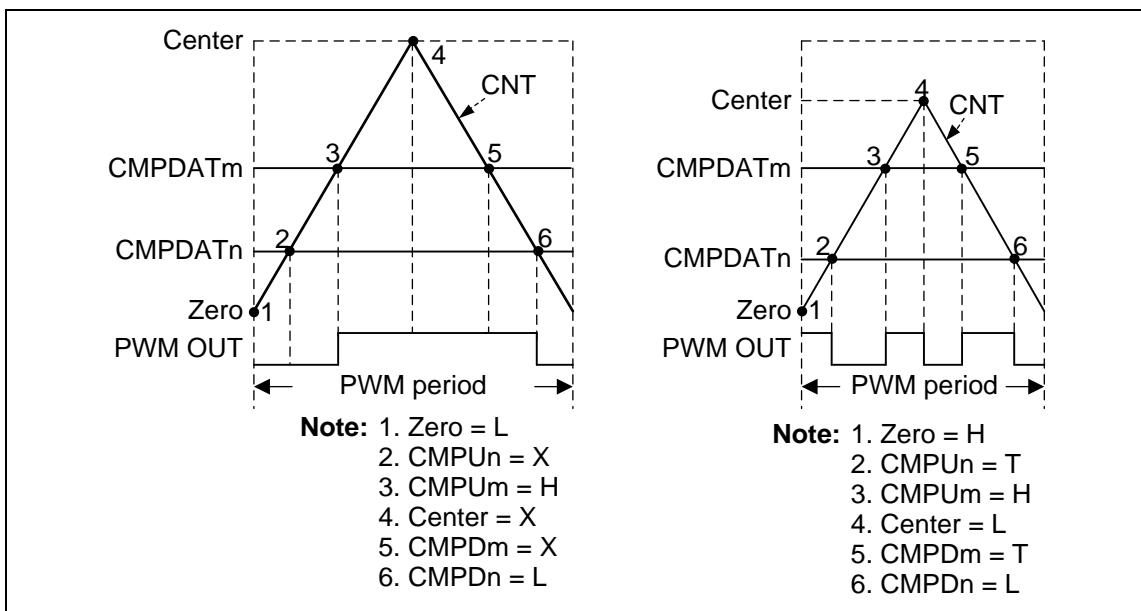


图 6-37 PWM脉冲产生

有时事件会被同时产生，因此，不同计数方式的事件优先级列表如下：向上计数方式（表 6-11），向下计数方式（表 6-12），上下计数方式（表 6-13）。通过事件优先级，使用者可以很容易产生0% 到100%占空比的脉冲，如图 6-38所示。

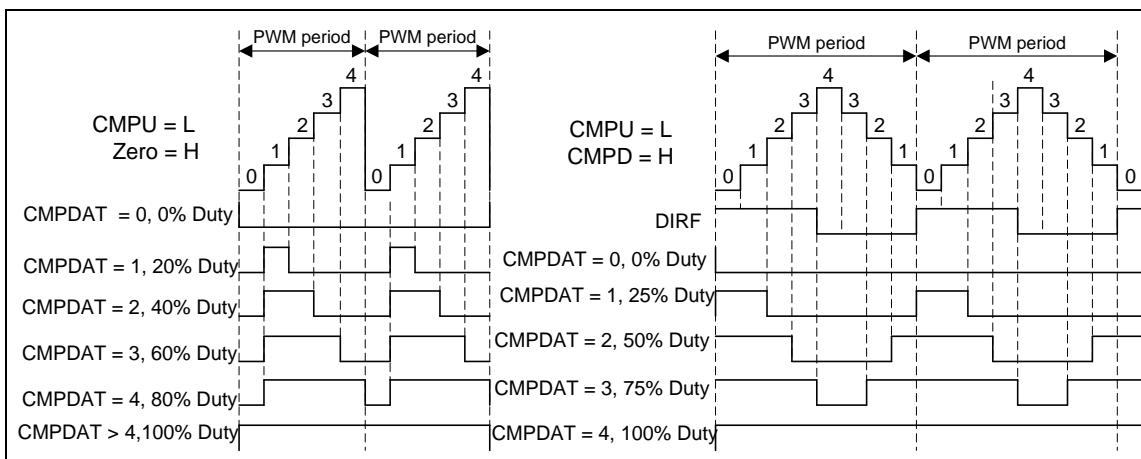


图 6-38 PWM 0% 到 100% 占空比脉冲发生

优先级	向上事件
1 (最高)	CNT = 周期 (PERIOD)
2	CNT = CMPUm
3	CNT = CMPUn
4 (最低)	CNT = 零点

表 6-11 向上计数器PWM脉冲发生事件优先级

优先级	向下事件
1 (最高)	CNT = 零点
2	CNT = CMPDm
3	CNT = CMPDn
4 (最低)	CNT = 周期 (PERIOD)

表 6-12 向下计数器 PWM 脉冲发生优先级

优先级	向上事件	向下事件
1 (最高)	CNT = CMPUm	CNT = CMPDm
2	CNT= CMPUn	CNT = CMPDn
3	CNT = 零点	CNT = 中心点 (PERIOD)
4	CNT = CMPDm	CNT = CMPUm
5 (最低)	周期 = CMPDn	CNT = CMPUn

表 6-13 上下计数器 PWM 脉冲产生事件优先级

6.7.5.12 PWM输出模式

PWM支持两个输出模式：独立模式，可以应用于直流电机系统；带死区插入互补模式，可以由于交流感应电机和永磁同步电机。

6.7.5.13 独立模式

默认PWM在独立模式，当信道n相应位PWMMODEn (PWM_CTL1[26:24])置0 独立模式被使能。独立模式中6个PWM信道：PWM_CH0, PWM_CH1, PWM_CH2, PWM_CH3, PWM_CH4 和 PWM_CH5 都运行在各自的周期占空比，如图 6-39所示：

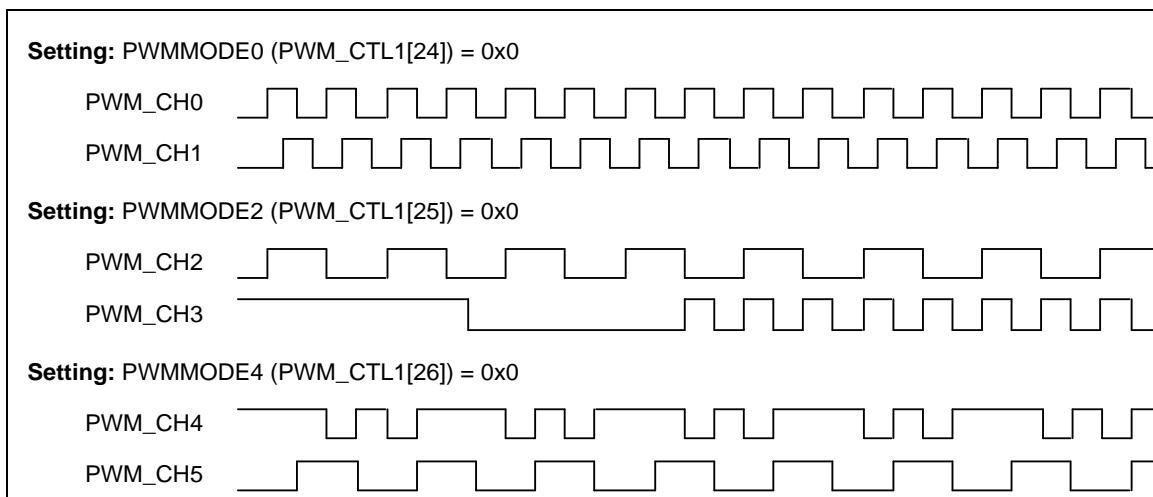


图 6-39 PWM 独立模式波形

6.7.5.14 互补模式

当互补通道与PWMMODEn (PWM_CTL1[26:24])相应位被置1，互补模式被使能。互补模式有3个PWM发生器，模块中总共3组PWM输出管脚。互补模式中内部奇数PWM信号必须与相应偶数通道的PWM信号互补。PWM_CH1将互补于PWM_CH0，PWM_CH3将互补于PWM_CH2，PWM_CH5将互补于PWM_CH4，如图 6-40所示：

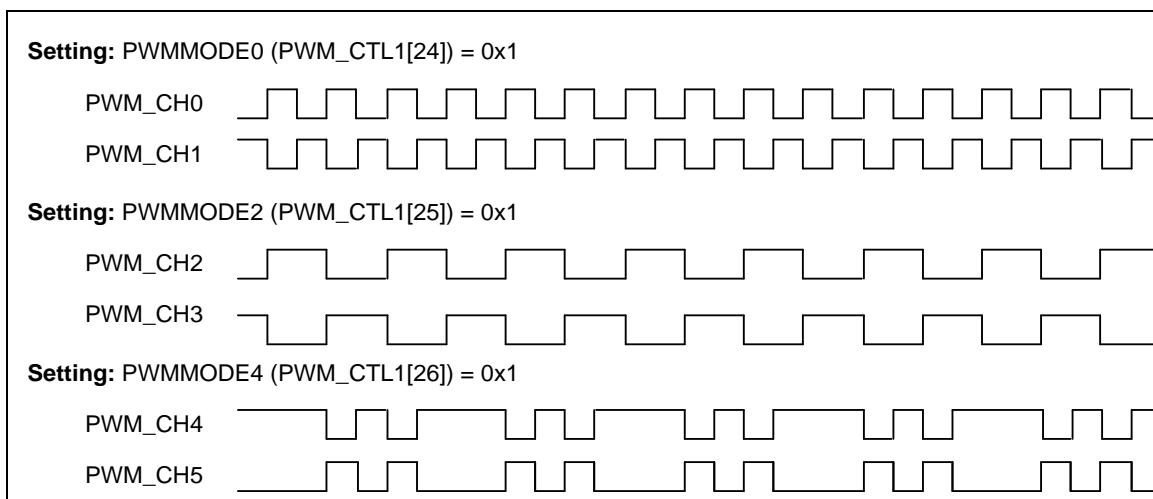


图 6-40 PWM 互补模式波形

6.7.5.15 PWM 输出控制

PWM脉冲产生后，有4到6步来控制PWM通道输出。独立模式中，屏蔽，刹车，管脚极性和输出使能4步如图 6-41所示。互补模式中，在以上4步前增加2步：互补通道和死区插入，如图 6-42所示。

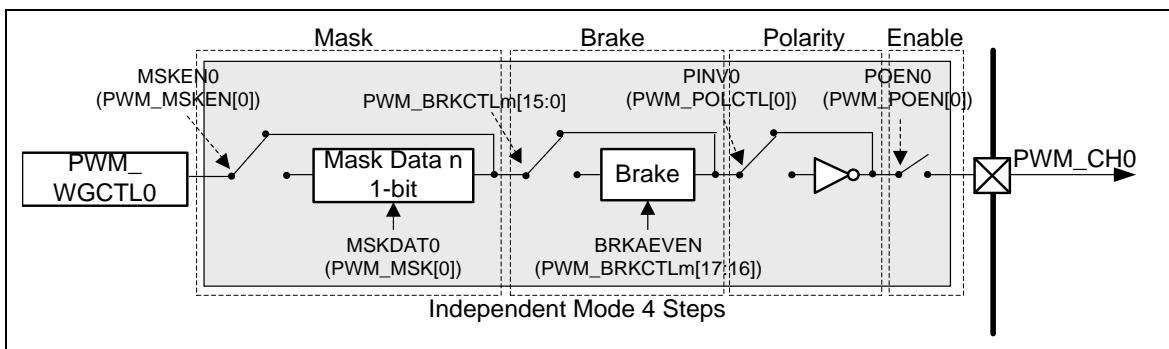


图 6-41 独立模式 PWM_CH0 输出控制

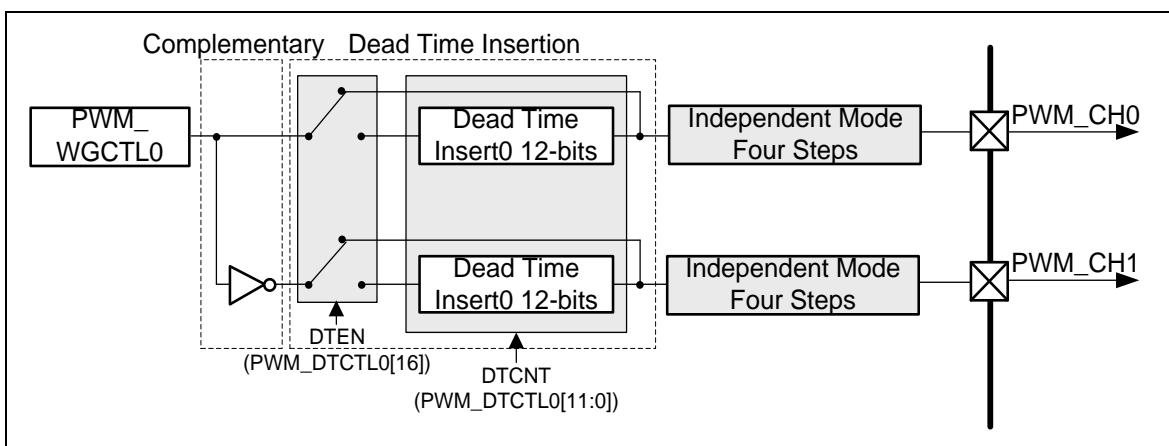


图 6-42 互补模式 PWM_CH0 和 PWM_CH1 输出控制

6.7.5.16 死区插入

互补应用中，互补通道也许用来驱动外部设备例如开关电源，在互补输出通道间死区发生器插入一个低电平周期叫（死区时间）来安全驱动设备，防止系统或设备烧坏。因此死区控制是正确操作互补系统的重要机制。通过设置通道 n DTEN (PWM_DTCTLn[16]) 的相应位来使能死区功能，设置 DTCNT (PWM_DTCTLn[11:0]) 控制死区时间周期，死区时间可以通过以下公式计算：

$$\text{死区时间} = (\text{DTCNT}[11:0]+1) * \text{PWMx_CLK 周期}$$

图 6-43 表示一个死区插入到一组 PWM 信号。

死区时钟源可以通过 DTCKSEL (PWM_DTCTLn[24]) 置 1 选择来自预分频输出。默认时钟源是来自预分频输入的 PWM_CLK。请注意：PWM_DTCTLn 是写保护寄存器。

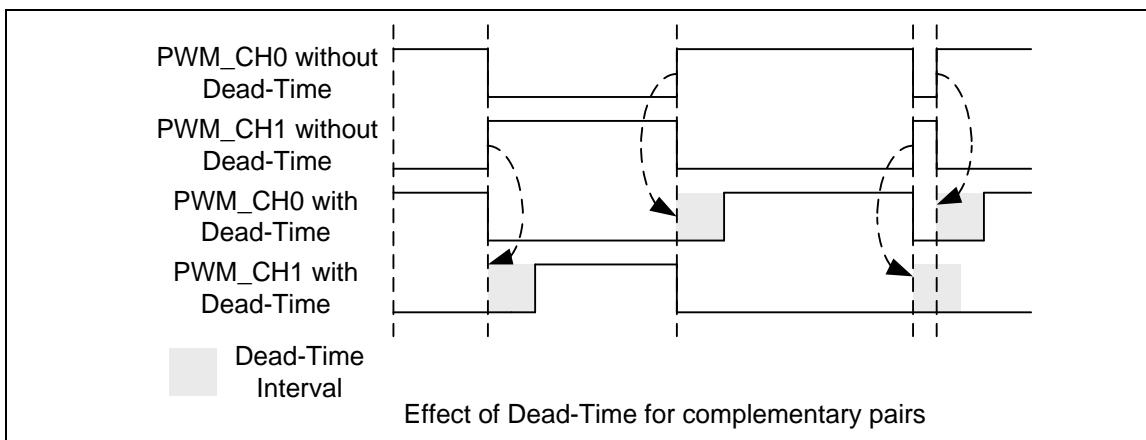


图 6-43 死区插入

6.7.5.17 PWM屏蔽输出功能

通过设置PWM屏蔽使能控制寄存器(PWM_MSKEN)和PWM屏蔽数据寄存器(PWM_MSK)可以对每个PWM的输出手动覆盖。通过这些设置，PWM通道的输出可以被强制设定为特定的逻辑状态，而与比较单元无关。PWM屏蔽位用于控制各类电子整流电机(ECM)，如BLDC电机。PWM_MSKEN寄存器有六个位，MSKEN_n (PWM_MSKEN[5:0])。如果MSKEN_n置高，通道n输出将被覆盖。PWM_MSK寄存器有六个位，MSKDAT_n (PWM_MSK[5:0])。当通道被覆盖，MSKDAT_n位值决定PWM通道n的输出状态值。图 6-44显示如何将PWM屏蔽控制器用于覆盖特性的例子。

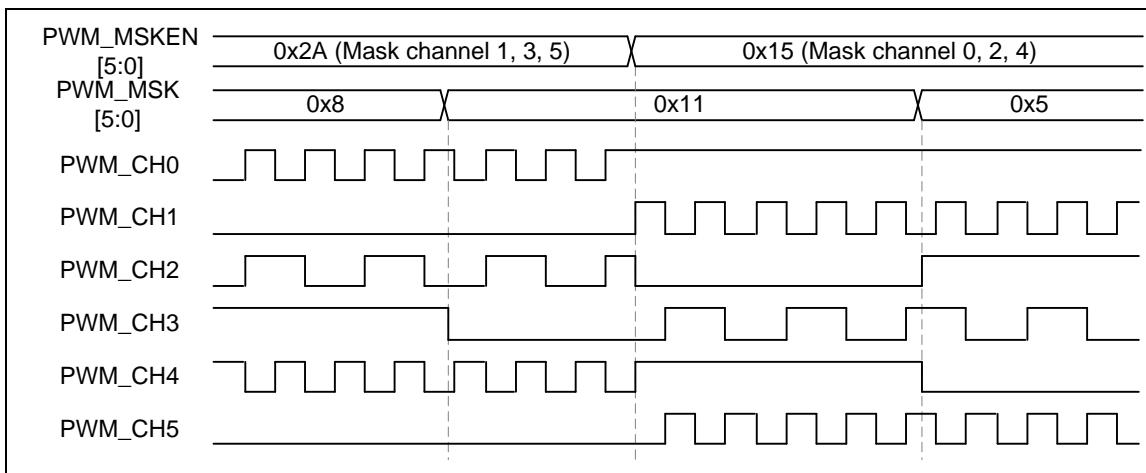


图 6-44 屏蔽控制波形图解

6.7.5.18 PWM刹车

每个PWM模块有两个外部输入刹车控制信号。外部信号将被一个3位噪音滤波器滤波。另外，外部信号可以通过设置BRK_xPINV (PWM_BNF[15, 7]，(x表示输入管脚0或1)被翻转，来实现刹车控制型号的极性设置。通过设置BRK_xFCS (PWM_BNF[11:9, 3:1])可以根据不同的噪音选择合适的噪音滤波器的采样时钟。另外，通过设置BRK_xFCNT (PWM_BNF[14:12, 6:4])用户可以定义滤波器需要多少次采样来确认有效刹车信号。配置BRK_xFEN (PWM_BNF[8, 0])将使能噪音滤波器，默认是关闭的。

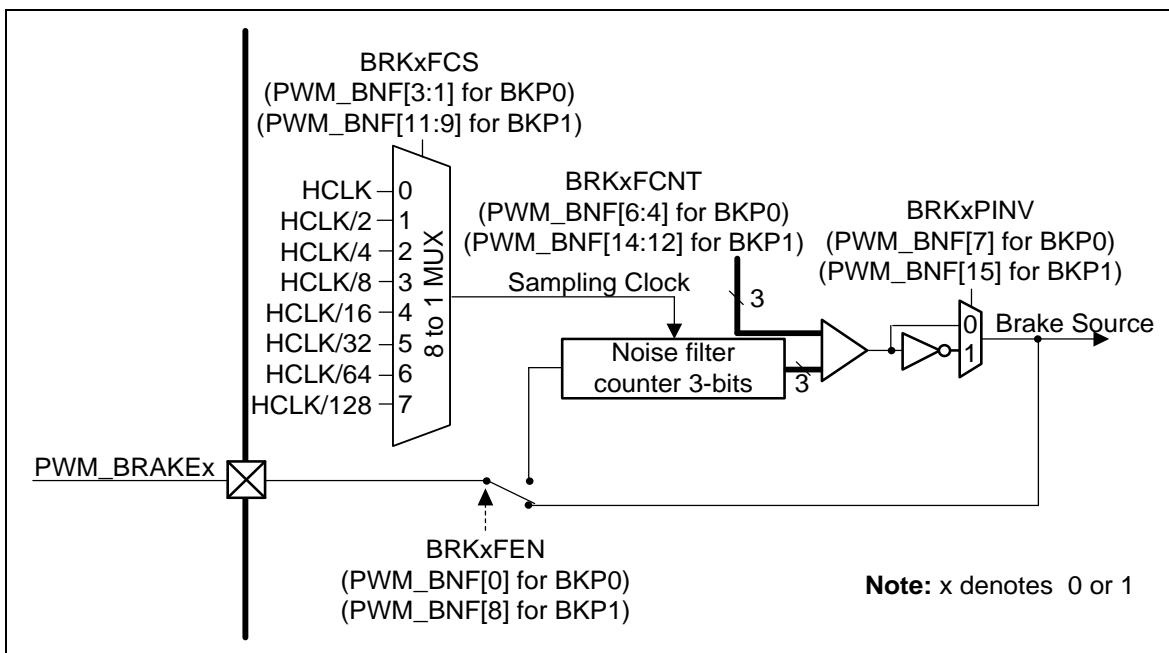


图 6-45 刹车噪音滤波器框图

每组互补通道共享一个刹车功能，如图 6-46。当刹车事件发生时，用户可以设置偶数通道的 BRKAEVEN (PWM_BRKCTL0_1[17:16]) 和奇数通道的 BRKAODD (PWM_BRKCTL0_1[19:18])，确保通道输出安全状态。有两个刹车检测：边缘检测和电平检测。当边缘检测发现刹车信号并且 BRKEIEN_{n_m} (PWM_INTEN1[2:0]) 被使能，刹车功能产生 BRK_INT。这个中断需要用软件清除，且在中断清除后 BRKESTS_n (PWM_INTSTS1[21:16]) 刹车状态将保持直到下一个 PWM 周期开始。通过电平检测，刹车功能也可以用另一种方式操作。一旦电平检测发现刹车信号变低并且 BRKLIE_{n_m} (PWM_INTEN1[10:8]) 被使能，刹车功能将产生 BRK_INT，但是当刹车信号回到高电平，当前 PWM 周期结束后，PWM 输出将变为正常，BRKLSTS_n (PWM_INTSTS1[29:24]) 位被自动清除。

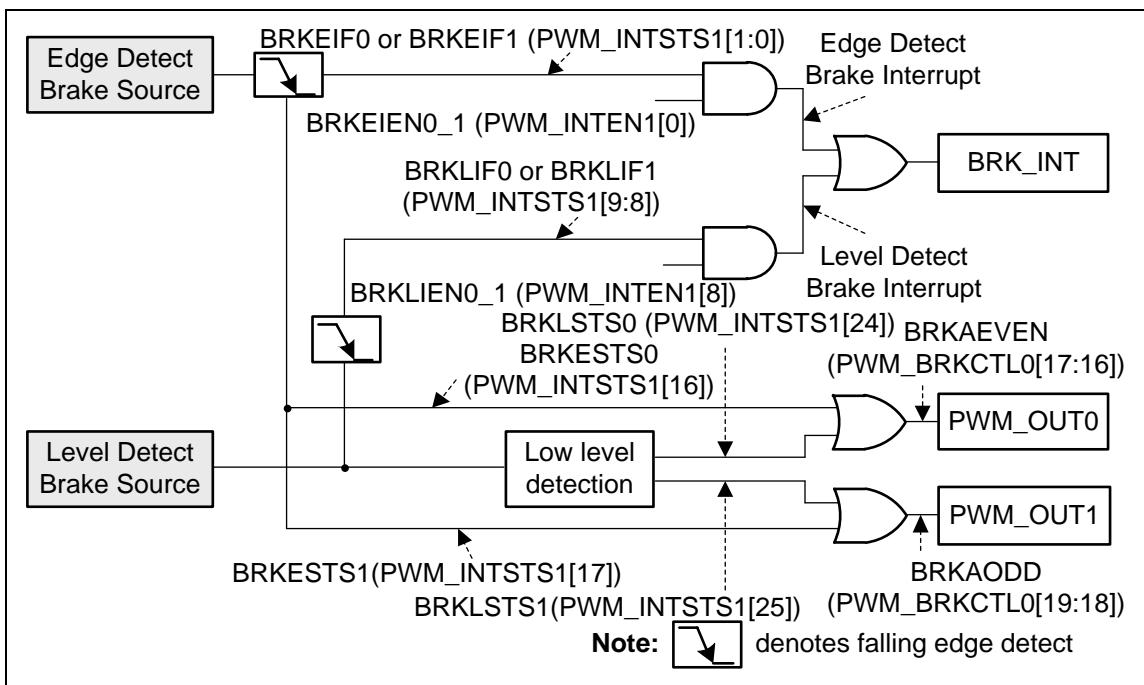


图 6-46 PWM_CH0 和 PWM_CH1 通道组 刹车框图

图 6-47, 表示 PWM_CH0 和 PWM_CH1 通道组的边缘检测波形。这种情况下, 边缘检测到两次刹车事件。当刹车事件发生 BRKEIF0 和 BRKEIF1 都被置 1 并且 BRKESTS0 和 BRKESTS1 也被置 1 表明 PWM_CH0 和 PWM_CH1 处于刹车状态。当第一个事件发生, 软件写 1 清 BRKEIF0。然后在下一个 PWM 周期开始时硬件清除 BRKESTS0。此时即使刹车事件依旧存在, PWM_CH0 仍然输出正常波形。第二个事件也触发相同标志, 但此时软件写 1 去清 BRKEIF1, 此后在下个 PWM 周期开始 PWM_CH1 正常输出。

相对于边缘检测例子, 图 6-48 表示 PWM_CH0 和 PWM_CH1 通道组的电平检测波形。这种情况, BRKLIF0 和 BRKLIF1 只可以表示刹车事件已发生。在下一个 PWM 周期开始时不管 BRKLIF0 和 BRKLIF1 是什么状态只要刹车事件消失, BRKLSTS0 和 BRKLSTS1 的刹车状态都会被自动恢复。

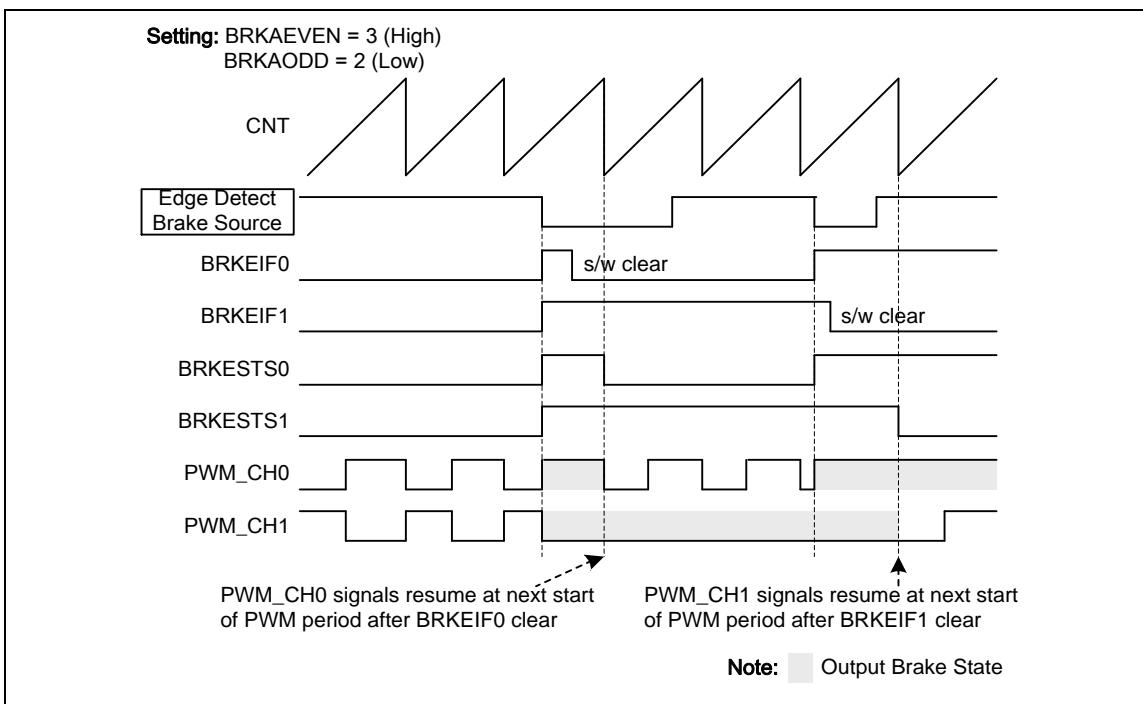


图 6-47 PWM_CH0 和 PWM_CH1 通道组的边缘检测波形

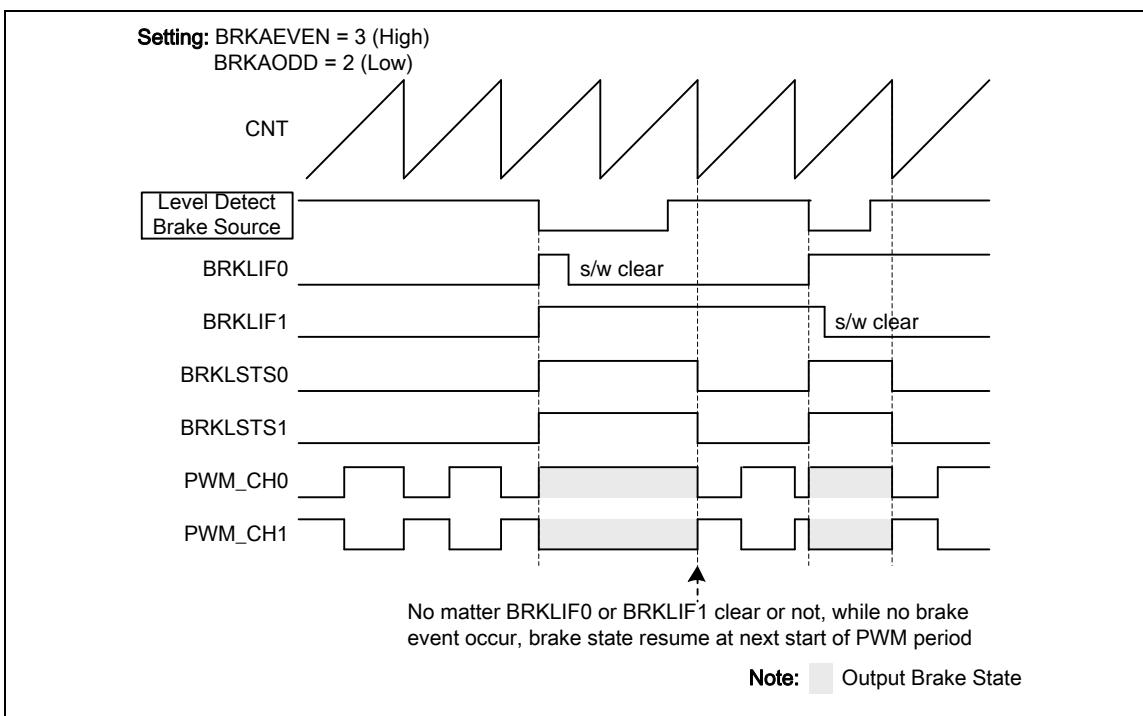


图 6-48 PWM_CH0 和 PWM_CH1 通道组的电平检测波形

两种检测器检测三个同样的刹车源：两个从外部输入的信号和一个系统故障，系统故障具有单独的刹车源使能控制。除了这三个刹车源，两个检测器还有一个软件触发刹车条件。如图 6-49 所示。

在以上描述的几个刹车源中，来自系统故障刹车源可以被指定为几种不同系统故障条件，这些故障条件，包括时钟故障，掉电保护和Cortex-M0 锁死。图 6-50 显示通过设置相应使能位，系统故障条件可以作为一个系统故障刹车源发送到 PWM 刹车。

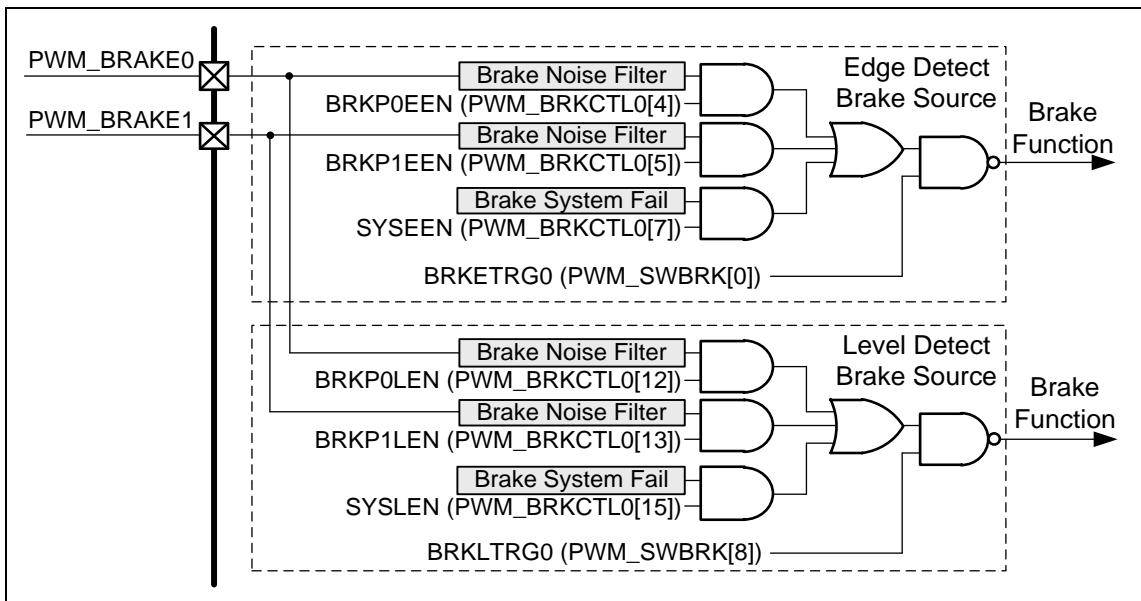


图 6-49 刹车源框图

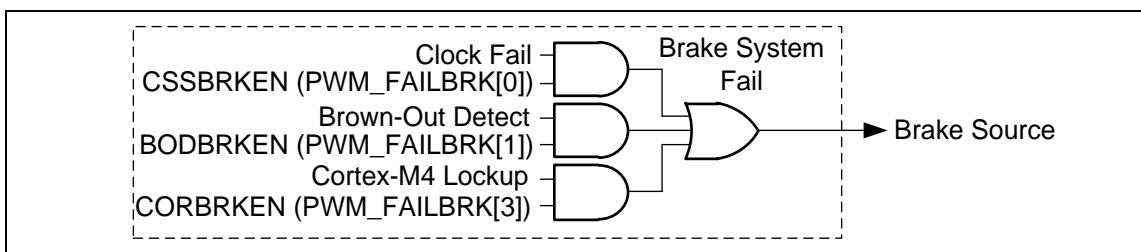


图 6-50 系统故障刹车框图

6.7.5.19 极性控制

每个 PWM 端口从 PWM_CH0 到 PWM_CH5 都有一个独立极性控制模块来配置 PWM 输出的极性状态。默认 PWM 输出时高，意味着 PWM OFF 状态时低 ON 状态时高。PWM 的极性可以通过 PWM 负极控制寄存器(PWM_POLCTL)来设置。图 6-51 所示，PWM 不同极性设置开始前的初始状态。

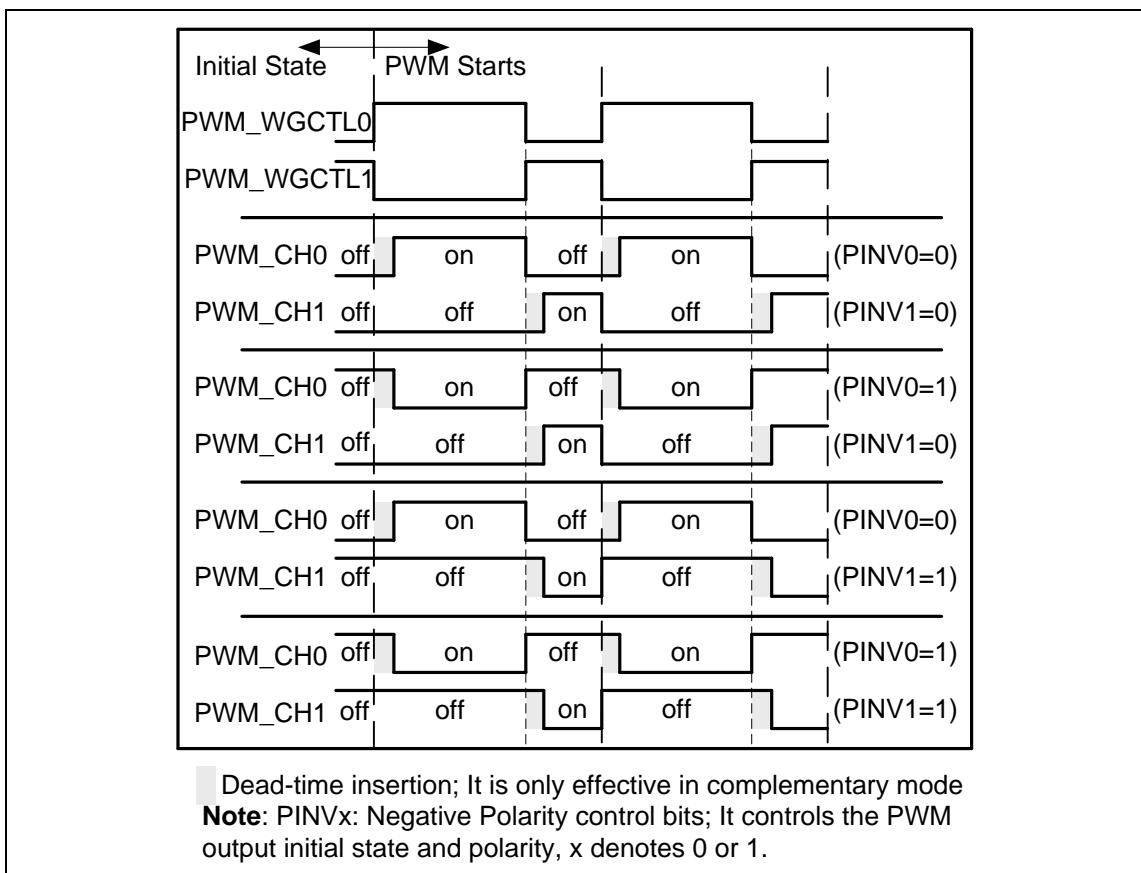


图 6-51 带上升沿死区插入初始状态和极性控制

6.7.5.20 PWM 中断发生器

每个 PWM 有三个独立中断，如图 6-52。

第一个 PWM 中断 (PWM_INT) 来自 PWM 互补组事件。计数器可以产生零点中断标志 ZIFn (PWM_INTSTS0[5:0]) 和周期点中断标志 PIFn (PWM_INTSTS0[13:8])。当 PWM 通道n 计数器值等于存在 PWM_CMPDATn 比较器值的时候，根据计数方向将触发不同的中断标志。如发生在向上计数则向上中断标志 CMPDIFn (PWM_INTSTS0[29:24]) 被置 1，如果相反方向则向下计数中断标志 CMPDIFn (PWM_INTSTS0[29:24]) 被置 1。信道n 的互补信道m 的比较器以同样的方法产生 CMPUIFm 和 CMPDIFm 中断。如果相应的中断使能位置 1，事件触发将产生中断信号。

第二个中断是捕捉中断 (CAP_INT)，在 NVIC 中与 PWM_INT 共享中断入口。当 CAPRIFn (PWM_CAPIF[5:0]) 被触发并且捕捉上升沿中断使能位 CAPRIENn (PWM_CAPIEN[5:0]) 置 1 可以产生 CAP_INT 中断。或者在下降沿条件，当捕捉下降沿中断使能位 CAPFIENn (PWM_CAPIEN[13:8]) 置 1 CAPIFI Fn (PWM_CAPIF[13:8]) 可以被触发。

最后是刹车中断(BRK_INT)，在 PWM 刹车部分有 BRK_INT 详细描述。

下图表示 PWM 中断架构。

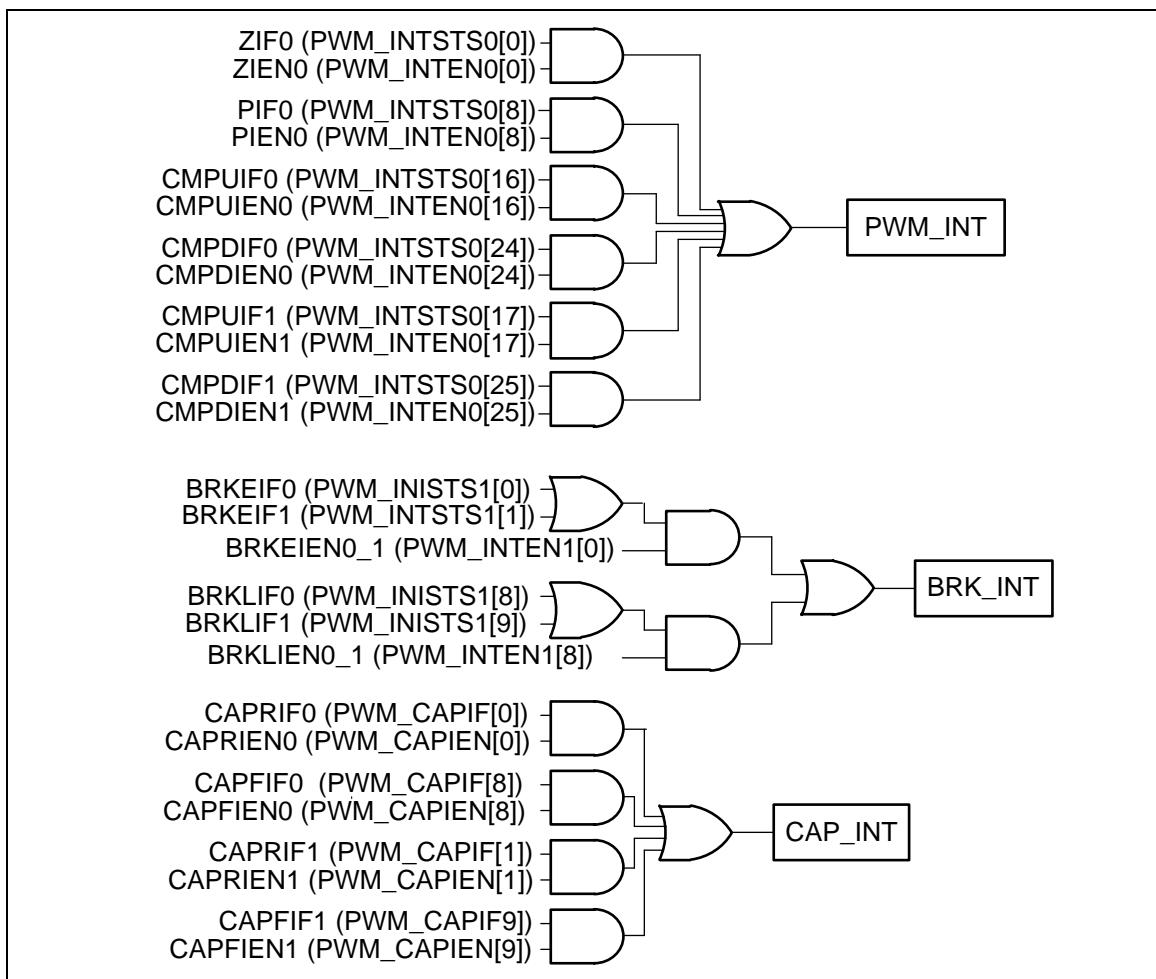


图 6-52 PWM_CH0 和 PWM_CH1 组中断架构图

6.7.5.21 PWM触发ADC发生器

PWM可以作为一个ADC转换触发源。每一个PWM通道组都共享同样的触发源。设置TRGSEL n 来选择触发源，TRGSEL n 有TRGSEL0, TRGSEL1...TRGSEL5，它们分别位于PWM_ADCTS0[3:0], PWM_ADCTS0[11:8], PWM_ADCTS0[19:16], PWM_ADCTS0[27:24], PWM_ADCTS1[3:0]和PWM_EADTS1[11:8]。设置TRGEN n 来使能触发输出到ADC, TRGEN n 是TRGEN0, TRGEN1, ..., TRGEN5, 他们分别位于PWM_ADCTS0[7], PWM_ADCTS0[15], PWM_ADCTS0[23], PWM_ADCTS0[31], PWM_ADCTS1[7]和PWM_ADCTS1[15], 数字 $n(n = 0, 1, \dots, 5)$ 表示PWM通道数。

一对通道可以有7个PWM事件触发源可选择。图 6-53是PWM_CH0 和 PWM_CH1的一个例子。PWM通过设置PERIOD 和 CMPDAT在不同时序可以触发ADC开始转换。图 6-54是在上下计数方式触发ADC的时序波形。

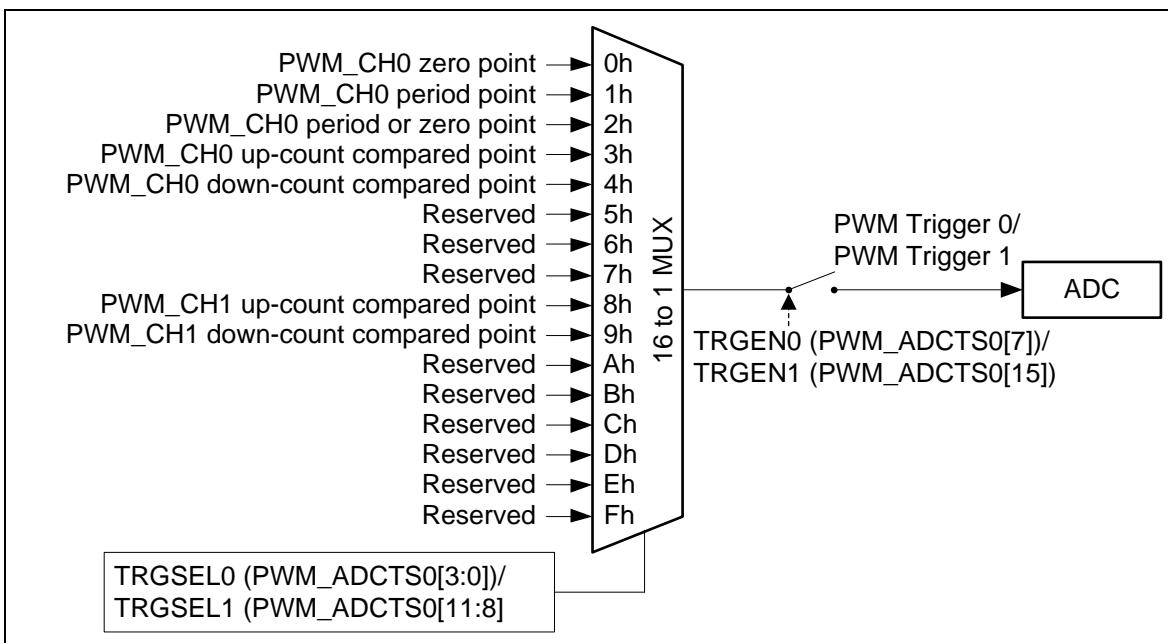


图 6-53 PWM_CH0 和 PWM_CH1 组触发ADC框图

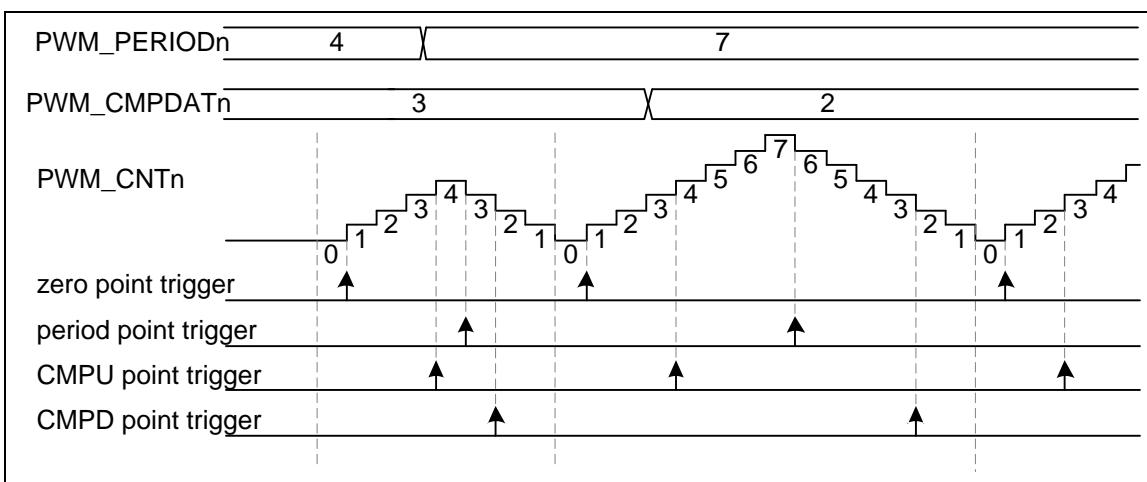


图 6-54 上下计数方式 PWM 触发 ADC 时序波形

6.7.5.22 捕捉操作

捕捉输入信道和PWM输出信道共享管脚和计数器。计数器可以是上或下计数方式。如果输入通道有上升沿或下降沿跳变时捕捉功能将 PWM 计数器值分别锁存到寄存器 RCAPDATn (PWM_RCAPDATn[15:0]) 和 FCAPDATn (PWM_FCAPDATn[15:0])。如果上升沿或下降沿锁存发生并且相应通道n的上升沿或下降沿中断使能位被设置，捕捉功能也将产生一个中断CAP_INT (使用 PWM_INT 向量)，CAPRIENn (PWM_CAPIEN[5:0]) 设置上升沿中断使能，CAPFIENn (PWM_CAPIEN[13:8]) 设置下降沿中断使能。当上升沿或下降沿锁存发生，相应的PWM计数器是否将重载PWM_PERIODn值，取决于设置RCRLDENn 或 FCRLDENn (RCRLDENn 和 FCRLDENn 分别位于 PWM_CAPCTL[21:16] 和 PWM_CAPCTL[29:24])。注：相应的GPIO管脚必须通过使能CAPINENn (PWM_CAPINEN[5:0]) 相应捕捉信道n来配置为捕捉功能。图 6-55 是通道0的捕捉框图。

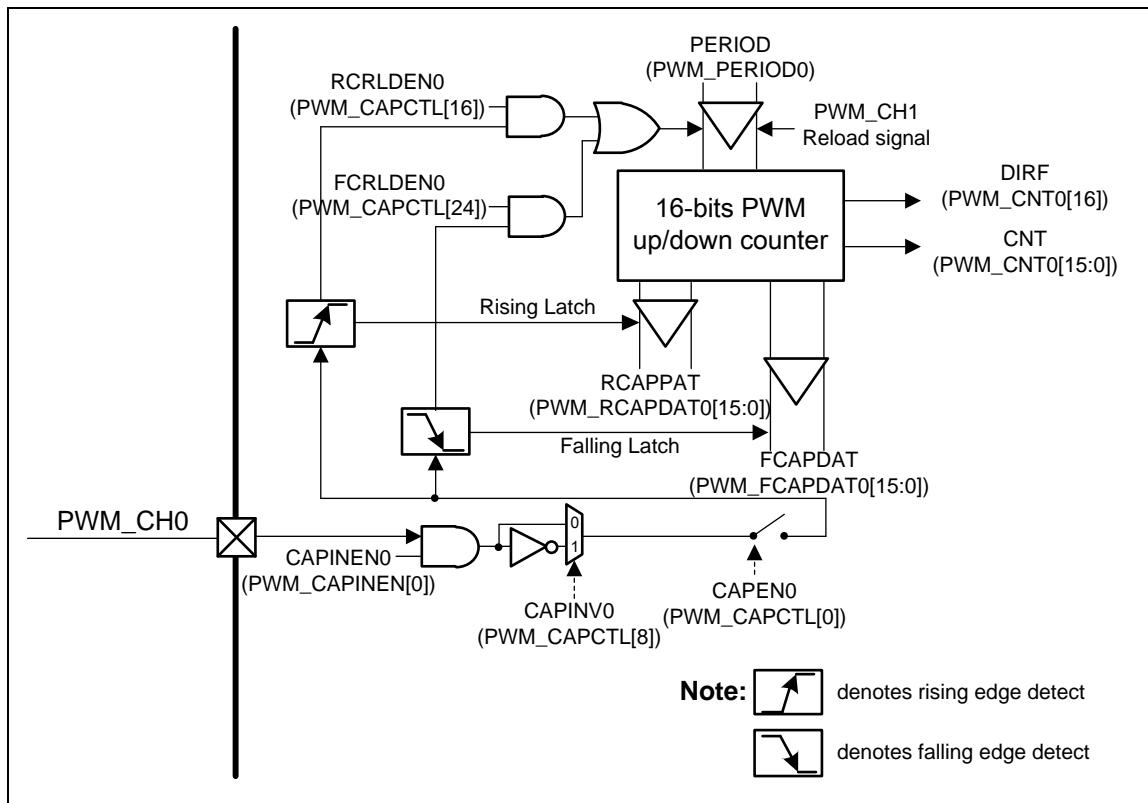


图 6-55 PWM_CH0 捕捉框图

图 6-55 表示捕捉功能时序。这种情况捕捉计数器设置为 PWM 向下计数方式并且 PERIOD 设置为 8，因此计数器从 8 到 0 向下计数。当捕捉管脚检测到一个下降沿，捕捉功能锁存计数器值到 PWM_FCAPDATn。当检测到上升沿将计数器值将锁存到 PWM_RCAPDATn。时序图中，当下降沿第一次被测到，捕捉功能将从 PERIOD 设置中重载计数值，因为 FCRLDENn 被使能。但是第二次下降沿没有导致重载，因为 FCRLDENn 被禁能了。这个例子中，在捕捉输入的上升沿计数器也被重载因为 RCRLDENn 被使能。

另外，这种情况如果是设为向上计数方式，计数器将重载零并向上计数到 PERIOD 值。重点是计数器是两个互补通道共享，因此计数器重载时序也被另一个通道的重载信号控制。

图 6-56 也表示中断和中断标志产生的时序例子。当上升沿在通道 n 被检测到，相应位 CAPRIFn (PWM_CAPIF[5:0]) 将被硬件设置。同样，通道 n 检测到下降沿 相应位 CAPEIFn (PWM_CAPIF[13:8]) 被硬件设置。CAPRIFn 和 CAPEIFn 可以通过软件写 1 清除。如果 CAPRIFn 被设置并且 CAPRIENn 被使能，捕捉功能将产生一个中断。如果 CAPEIFn 被设置并且 CAPFIENn 被使能，中断也会产生。

有一个情况图中没显示：当 CAPRIF 已经被设置如果上升沿锁存再次发生，溢出标志 CRIFOVn (PWM_CAPSTS[5:0]) 将被硬件置 1，表示 CAPRIF 溢出了。同样如果下降沿锁存再次发生，硬件同样操作中断标志 CAPEIF 和溢出标志 CFIFOVn (PWM_CAPSTS[13:8])。

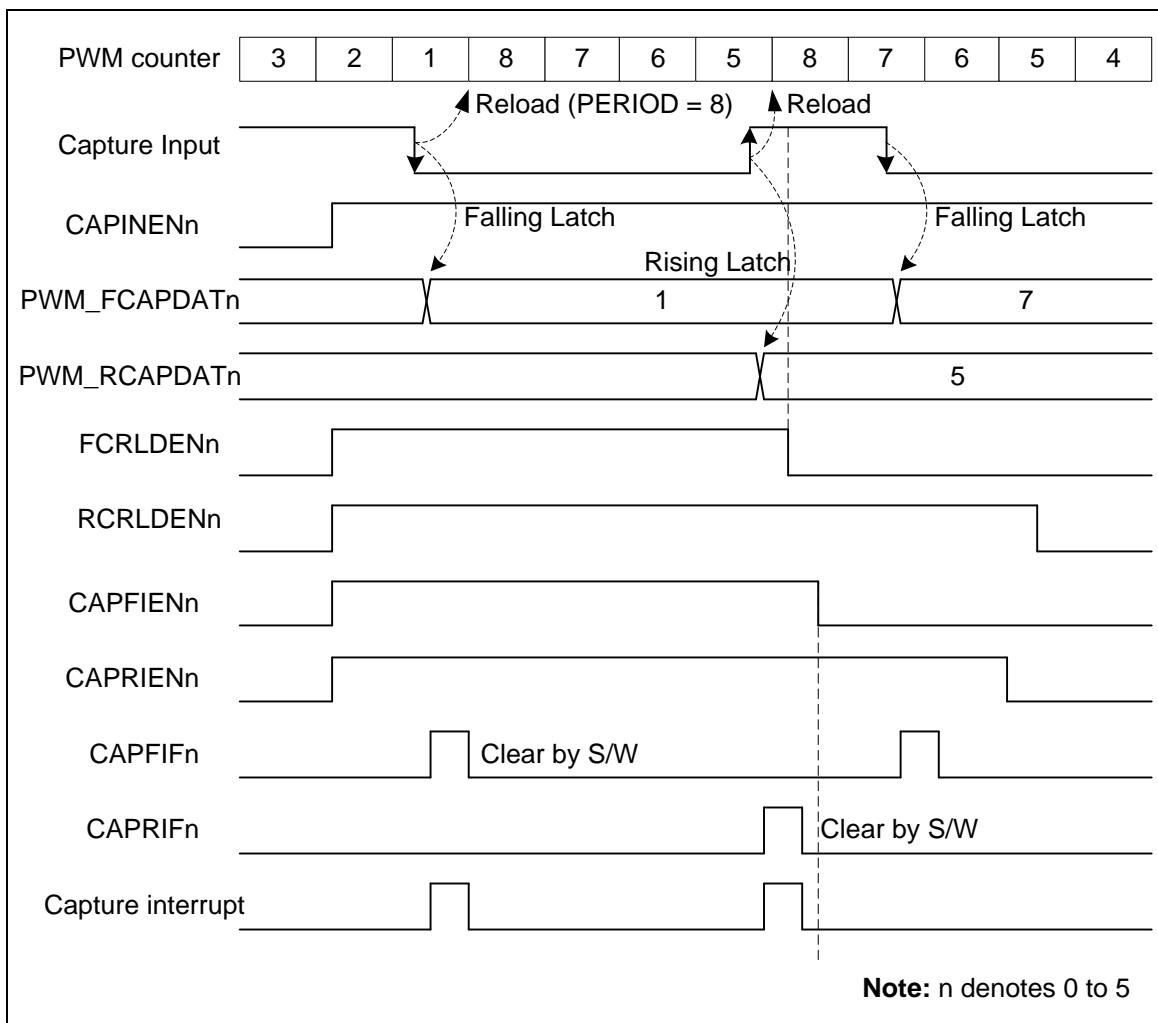


图 6-56 捕捉操作波形

捕捉脉冲宽度可以根据以下公式计算：

对于负脉冲，通道低脉冲宽度被计算为($PWM_PERIODn + 1 - PWM_RCAPDATn$). 在图 6-56，低脉冲宽度是 $8+1-5 = 4$ 。

对于正脉冲，通道高脉冲宽度计算为($PWM_PERIODn + 1 - PWM_FCAPDATn$). 在图 6-56,高脉冲宽度是 $8+1-7 = 2$ 。



6.7.6 寄存器映射

R: 只读, **W:** 只写, **R/W:** 读/写

寄存器	偏移量	R/W	描述	复位后值
PWM 基地址:				
PWM0_BA = 0x4004_0000				
PWM1_BA = 0x4014_0000				
PWM_CTL0 x=0, 1	PWMx_BA+0x00	R/W	PWM 控制寄存器0	0x0000_0000
PWM_CTL1 x=0, 1	PWMx_BA+0x04	R/W	PWM 控制寄存器 1	0x0000_0000
PWM_CLKSRC x=0, 1	PWMx_BA+0x10	R/W	PWM 时钟源寄存器	0x0000_0000
PWM_CLKPSC0 _1 x=0, 1	PWMx_BA+0x14	R/W	PWM 时钟预分频寄存器0	0x0000_0000
PWM_CLKPSC2 _3 x=0, 1	PWMx_BA+0x18	R/W	PWM时钟预分频寄存器2	0x0000_0000
PWM_CLKPSC4 _5 x=0, 1	PWMx_BA+0x1C	R/W	PWM时钟预分频寄存器4	0x0000_0000
PWM_CNTEN x=0, 1	PWMx_BA+0x20	R/W	PWM计数器使能寄存器	0x0000_0000
PWM_CNTCLR x=0, 1	PWMx_BA+0x24	R/W	PWM 计数器清除寄存器	0x0000_0000
PWM_PERIOD0 x=0, 1	PWMx_BA+0x30	R/W	PWM 周期寄存器 0	0x0000_0000
PWM_PERIOD2 x=0, 1	PWMx_BA+0x38	R/W	PWM 周期寄存器 2	0x0000_0000
PWM_PERIOD4 x=0, 1	PWMx_BA+0x40	R/W	PWM 周期寄存器 4	0x0000_0000
PWM_CMPDAT0 x=0, 1	PWMx_BA+0x50	R/W	PWM 比较器寄存器 0	0x0000_0000
PWM_CMPDAT1 x=0, 1	PWMx_BA+0x54	R/W	PWM比较器寄存器1	0x0000_0000
PWM_CMPDAT2 x=0, 1	PWMx_BA+0x58	R/W	PWM比较器寄存器2	0x0000_0000
PWM_CMPDAT3 x=0, 1	PWMx_BA+0x5C	R/W	PWM比较器寄存器3	0x0000_0000
PWM_CMPDAT4	PWMx_BA+0x60	R/W	PWM比较器寄存器4	0x0000_0000

x=0, 1				
PWM_CMPDAT5 x=0, 1	PWMx_BA+0x64	R/W	PWM 比较器寄存器5	0x0000_0000
PWM_DTCTL0_1 x=0, 1	PWMx_BA+0x70	R/W	PWM 死区控制寄存器0_1	0x0000_0000
PWM_DTCTL2_3 x=0, 1	PWMx_BA+0x74	R/W	PWM死区控制寄存器2_3	0x0000_0000
PWM_DTCTL4_5 x=0, 1	PWMx_BA+0x78	R/W	PWM死区控制寄存器4_5	0x0000_0000
PWM_CNT0 x=0, 1	PWMx_BA+0x90	R	PWM 计数器寄存器 0	0x0000_0000
PWM_CNT2 x=0, 1	PWMx_BA+0x98	R	PWM 计数器寄存器 2	0x0000_0000
PWM_CNT4 x=0, 1	PWMx_BA+0xA0	R	PWM 计数器寄存器 4	0x0000_0000
PWM_WGCTL0 x=0, 1	PWMx_BA+0xB0	R/W	PWM 发生寄存器 0	0x0000_0000
PWM_WGCTL1 x=0, 1	PWMx_BA+0xB4	R/W	PWM发生寄存器1	0x0000_0000
PWM_MSKEN x=0, 1	PWMx_BA+0xB8	R/W	PWM 屏蔽使能寄存器	0x0000_0000
PWM_MSK x=0, 1	PWMx_BA+0xBC	R/W	PWM 屏蔽数据寄存器	0x0000_0000
PWM_BNF x=0, 1	PWMx_BA+0xC0	R/W	PWM 刹车噪音滤波寄存器	0x0000_0000
PWM_FAILBRK x=0, 1	PWMx_BA+0xC4	R/W	PWM 系统故障刹车寄存器	0x0000_0000
PWM_BRKCTL0_1 x=0, 1	PWMx_BA+0xC8	R/W	PWM 刹车边缘检测控制寄存器0_1	0x0000_0000
PWM_BRKCTL2_3 x=0, 1	PWMx_BA+0xCC	R/W	PWM刹车边缘检测控制寄存器2_3	0x0000_0000
PWM_BRKCTL4_5 x=0, 1	PWMx_BA+0xD0	R/W	PWM刹车边缘检测控制寄存器4_5	0x0000_0000
PWM_POLCTL x=0, 1	PWMx_BA+0xD4	R/W	PWM 管脚极性转换寄存器	0x0000_0000
PWM_POEN x=0, 1	PWMx_BA+0xD8	R/W	PWM 输出使能寄存器	0x0000_0000
PWM_SWBRK	PWMx_BA+0xDC	W	PWM 软件刹车寄存器	0x0000_0000

x=0, 1				
PWM_INTENO x=0, 1	PWMx_BA+0xE0	R/W	PWM 中断使能寄存器 0	0x0000_0000
PWM_INTEN1 x=0, 1	PWMx_BA+0xE4	R/W	PWM 中断使能寄存器1	0x0000_0000
PWM_INTSTS0 x=0, 1	PWMx_BA+0xE8	R/W	PWM 中断标志寄存器 0	0x0000_0000
PWM_INTSTS1 x=0, 1	PWMx_BA+0xEC	R/W	PWM 中断标志寄存器1	0x0000_0000
PWM_ADCTS0 x=0, 1	PWMx_BA+0xF8	R/W	PWM触发 ADC源选择寄存器 0	0x0000_0000
PWM_ADCTS1 x=0, 1	PWMx_BA+0xFC	R/W	PWM 触发 ADC源选择寄存器 1	0x0000_0000
PWM_SSCTL x=0, 1	PWMx_BA+0x110	R/W	PWM 同步开始控制寄存器	0x0000_0000
PWM_SSTRG x=0, 1	PWMx_BA+0x114	W	PWM 同步开始触发寄存器	0x0000_0000
PWM_STATUS x=0, 1	PWMx_BA+0x120	R/W	PWM 状态寄存器	0x0000_0000
PWM_CAPINEN x=0, 1	PWMx_BA+0x200	R/W	PWM捕捉输入使能寄存器	0x0000_0000
PWM_CAPCTL x=0, 1	PWMx_BA+0x204	R/W	PWM捕捉控制寄存器	0x0000_0000
PWM_CAPSTS x=0, 1	PWMx_BA+0x208	R	PWM 捕捉状态寄存器	0x0000_0000
PWM_RCAPDAT0 x=0, 1	PWMx_BA+0x20C	R	PWM上升沿捕捉数字寄存器0	0x0000_0000
PWM_FCAPDAT0 x=0, 1	PWMx_BA+0x210	R	PWM下降沿捕捉数字寄存器0	0x0000_0000
PWM_RCAPDAT1 x=0, 1	PWMx_BA+0x214	R	PWM上升沿捕捉数字寄存器1	0x0000_0000
PWM_FCAPDAT1 x=0, 1	PWMx_BA+0x218	R	PWM 下降沿捕捉数字寄存器 1	0x0000_0000
PWM_RCAPDAT2 x=0, 1	PWMx_BA+0x21C	R	PWM 上升沿捕捉数字寄存器2	0x0000_0000
PWM_FCAPDAT2	PWMx_BA+0x220	R	PWM 下降沿捕捉数字寄存器 2	0x0000_0000

x=0, 1				
PWM_RCAPDAT 3 x=0, 1	PWMx_BA+0x224	R	PWM上升沿捕捉数字寄存器3	0x0000_0000
PWM_FCAPDAT 3 x=0, 1	PWMx_BA+0x228	R	PWM下降沿捕捉数字寄存器 3	0x0000_0000
PWM_RCAPDAT 4 x=0, 1	PWMx_BA+0x22C	R	PWM 上升沿捕捉数字寄存器4	0x0000_0000
PWM_FCAPDAT 4 x=0, 1	PWMx_BA+0x230	R	PWM 下降沿捕捉数字寄存器 4	0x0000_0000
PWM_RCAPDAT 5 x=0, 1	PWMx_BA+0x234	R	PWM 上升沿捕捉数字寄存器 5	0x0000_0000
PWM_FCAPDAT 5 x=0, 1	PWMx_BA+0x238	R	PWM 下降沿捕捉数字寄存器5	0x0000_0000
PWM_CAPIEN x=0, 1	PWMx_BA+0x250	R/W	PWM 捕捉中断使能寄存器	0x0000_0000
PWM_CAPIF x=0, 1	PWMx_BA+0x254	R/W	PWM 捕捉中断标志寄存器	0x0000_0000
PWM_PBUFO x=0, 1	PWMx_BA+0x304	R	PWM PERIOD0 缓存	0x0000_0000
PWM_PBUF2 x=0, 1	PWMx_BA+0x30C	R	PWM PERIOD2 缓存	0x0000_0000
PWM_PBUF4 x=0, 1	PWMx_BA+0x314	R	PWM PERIOD4 缓存	0x0000_0000
PWM_CMPBUFO x=0, 1	PWMx_BA+0x31C	R	PWM CMPDAT0 缓存	0x0000_0000
PWM_CMPBUF1 x=0, 1	PWMx_BA+0x320	R	PWM CMPDAT1 缓存	0x0000_0000
PWM_CMPBUF2 x=0, 1	PWMx_BA+0x324	R	PWM CMPDAT2 缓存	0x0000_0000
PWM_CMPBUF3 x=0, 1	PWMx_BA+0x328	R	PWM CMPDAT3 缓存	0x0000_0000
PWM_CMPBUF4 x=0, 1	PWMx_BA+0x32C	R	PWM CMPDAT4 缓存	0x0000_0000
PWM_CMPBUF5 x=0, 1	PWMx_BA+0x330	R	PWM CMPDAT5 缓存	0x0000_0000



6.7.7 寄存器描述

PWM 控制寄存器0(PWM_CTL0)

寄存器	偏移量	R/W	描述	复位后值
PWM_CTL0	PWMx_BA+0x00	R/W	PWM 控制寄存器0	0x0000_0000

31	30	29	28	27	26	25	24
DBGTRIOFF	DBGHALT	Reserved					
23	22	21	20	19	18	17	16
Reserved	IMMLDEN5	IMMLDEN4	IMMLDEN3	IMMLDEN2	IMMLDEN1	IMMLDEN0	
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		CTRLD5	CTRLD4	CTRLD3	CTRLD2	CTRLD1	CTRLD0

位	描述
[31]	DBGTRIOFF ICE 调试模式禁能（写保护） 0= ICE 调试模式影响PWM输出 ICE 调试模式期间PWM管脚将强制作为三态模式。 1= 禁止ICE 调试模式影响PWM输出。 PWM管脚将保持输出不管是否在ICE 调试模式 注：该寄存器写保护，参考寄存器REGWRPROT。
[30]	DBGHALT ICE 调试模式计数暂停（写保护） 如果 计数暂停被使能， PWM所有计数器将保持当前值直到退出ICE调试模式。 0 =禁止ICE 调试调试模式计数暂停 1 = 使能ICE 调试模式计数暂停 注：该寄存器写保护，参考寄存器REGWRPROT。
[29:22]	Reserved 保留.
[21:16]	IMMLDENn 立即装载使能 每位n控制相应PWM通道n 0 = 每个周期末端PERIOD将载入到PBUF.通过设定CTRLD位在每个周期的终点或者中点CMPDAT将载入到CMPBUF。 1=当软件更新PERIOD/CMPDAT， PERIOD/CMPDAT将立即分别载入到PBUF和CMPBUF。 注：如果IMMLDENn使能， WINLDENn 和 CTRLDn无效
[15:6]	Reserved 保留.

[5:0]	CTRLDn	中心重载 每位n控制相应PWM通道n 在上下计数方式，在每个周期末端PERIOD将载入到PBUF. 每个周期中点CMPDAT将载入到CMPBUF。
-------	--------	---



PWM控制寄存器1 (PWM_CTL1)

寄存器	偏移量	R/W	描述	复位后值
PWM_CTL1	PWMx_BA+0x04	R/W	PWM 控制寄存器 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PWMMODE4	PWMMODE2	PWMMODE0	
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				CNTTYPE4			
7	6	5	4	3	2	1	0
Reserved		CNTTYPE2		Reserved		CNTTYPE0	

位	描述	
[31:27]	Reserved	保留.
[26:24]	PWMMODEn	<p>PWM模式</p> <p>每位n控制相应PWM通道n</p> <p>0=PWM独立模式 1=PWM互补模式</p> <p>注：当操作在组功能，这些位必须都设置位同样模式。</p>
[23:10]	Reserved	保留.
[9:8]	CNTTYPE4	<p>PWM计数行为方式4</p> <p>每位n控制相应PWM通道n</p> <p>00=向上计数方式（支持捕捉模式） 01=向下计数方式（支持捕捉模式） 10=上下计数方式 11 =保留</p>
[7:6]	Reserved	保留.
[5:4]	CNTTYPE2	<p>PWM计数行为方式2</p> <p>每位n控制相应PWM通道n</p> <p>00=向上计数方式（支持捕捉模式） 01=向下计数方式（支持捕捉模式） 10=上下计数方式 11 =保留</p>

[3:2]	Reserved	保留.
[1:0]	CNTTYPE0	PWM计数行为方式0 每位n控制相应PWM通道n 00=向上计数方式（支持捕捉模式） 01=向下计数方式（支持捕捉模式） 10=上下计数方式 11 =保留



PWM时钟源寄存器 (PWM_CLKSRC)

寄存器	偏移量	R/W	描述	复位后值
PWM_CLKSR_C	PWMx_BA+0x10	R/W	PWM 时钟源寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					ECLKSRC4		
15	14	13	12	11	10	9	8
Reserved					ECLKSRC2		
7	6	5	4	3	2	1	0
Reserved					ECLKSRC0		

位	描述	
[31:19]	Reserved	保留
[18:16]	ECLKSRC4	PWM_CH45外部时钟源选择 000 = PWMx_CLK, x 表示 0 或 1. 001 = TIMERO 溢出. 010 = TIMER1 溢出. 011 = TIMER2 溢出. 100 = TIMER3 溢出. 其他 = 保留.
[15:11]	Reserved	保留.
[10:8]	ECLKSRC2	PWM_CH23外部时钟源选择 000 = PWMx_CLK, x 表示 0 或 1. 001 = TIMERO 溢出. 010 = TIMER1 溢出. 011 = TIMER2 溢出. 100 = TIMER3 溢出. 其他 = 保留.
[7:3]	Reserved	保留.
[2:0]	ECLKSRC0	PWM_CH01外部时钟源选择 000 = PWMx_CLK, x 表示 0 或 1. 001 = TIMERO 溢出.

		010 = TIMER1 溢出. 011 = TIMER2 溢出. 100 = TIMER3 溢出. 其他 = 保留.
--	--	--

PWM 时钟预分频寄存器0_1, 2_3, 4_5 (PWM_CLKPSC0_1, 2_3, 4_5)

寄存器	偏移量	R/W	描述	复位后值
PWM_CLKPS_C0_1	PWMx_BA+0x14	R/W	PWM时钟预分频寄存器0_1	0x0000_0000
PWM_CLKPS_C2_3	PWMx_BA+0x18	R/W	PWM时钟预分频寄存器2_3	0x0000_0000
PWM_CLKPS_C4_5	PWMx_BA+0x1C	R/W	PWM时钟预分频寄存器4_5	0x0000_0000

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

位	描述	
[31:12]	Reserved	保留
[11:0]	CLKPSC	PWM 计数器时钟预分频 PWM计数器时钟由时钟预分频器决定。每个PWM组共享一个PWM寄存器时钟预分频器。 PWM计数器时钟被(CLKPSC+1)除。



PWM计数使能寄存器 (PWM_CNTEN)

寄存器	偏移量	R/W	描述	复位后值
PWM_CNTEN	PWMx_BA+0x20	R/W	PWM 计数使能寄存器	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			CNTEN4	Reserved	CNTEN2	Reserved	CNTEN0

位	描述	
[31:5]	Reserved	保留
[4]	CNTEN4	PWM计数使能4 0=PWM计数器和时钟分频器停止工作 1=PWM计数器和时钟分频器开始工作
[3]	Reserved	保留.
[2]	CNTEN2	PWM计数使能2 0=PWM计数器和时钟分频器停止工作 1=PWM计数器和时钟分频器开始工作
[1]	Reserved	Reserved.
[0]	CNTEN0	PWM 计数使能0 0=PWM计数器和时钟分频器停止工作 1=PWM计数器和时钟分频器开始工作



PWM 清除计数器寄存器(PWM_CNTCLR)

寄存器	偏移量	R/W	描述	复位后值
PWM_CNTCLR	PWMx_BA+0x24	R/W	PWM清除计数器寄存器	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			CNTCLR4	Reserved	CNTCLR2	Reserved	CNTCLR0

位	描述	
[31:5]	Reserved	保留
[4]	CNTCLR4	<p>清除PWM计数器控制位 4 此位由硬件自动清除 0 =不清除 1 =清16位PWM计数器到0000H</p>
[3]	Reserved	Reserved.
[2]	CNTCLR2	<p>清除PWM计数器控制位2 此位由硬件自动清除 0 =不清除 1 =清16位PWM计数器到0000H</p>
[1]	Reserved	Reserved.
[0]	CNTCLR0	<p>清除PWM计数器控制位0 此位由硬件自动清除 0 =不清除 1 =清16位PWM计数器到0000H</p>

PWM周期寄存器0, 2, 4 (PWM_PERIOD0, 2, 4)

寄存器	偏移量	R/W	描述	复位后值
PWM_PERIOD0	PWMx_BA+0x30	R/W	PWM周期寄存器0	0x0000_0000
PWM_PERIOD2	PWMx_BA+0x38	R/W	PWM周期寄存器2	0x0000_0000
PWM_PERIOD4	PWMx_BA+0x40	R/W	PWM周期寄存器4	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留.
[15:0]	PERIOD	<p>PWM周期寄存器</p> <p>向上计数模式，此模式下，PWM计数器从0计数到PERIOD并从0重新开始。</p> <p>向下计数模式：此模式下，PWM计数器从PERIOD计数到0并从PERIOD重新开始。</p> <p>PWM周期时间=(PERIOD+1) * PWM_CLK 周期.</p> <p>上下计数模式：这个模式PWM计数器从0计数到PERIOD然后递减到0，并重复。</p> <p>PWM周期时间=2 * PERIOD * PWM_CLK周期.</p>

PWM比较寄存器 0~5 (PWM_CMPDAT0~5)

寄存器	偏移量	R/W	描述	复位后值
PWM_CMPDA_T0	PWMx_BA+0x50	R/W	PWM比较寄存器0	0x0000_0000
PWM_CMPDA_T1	PWMx_BA+0x54	R/W	PWM比较寄存器1	0x0000_0000
PWM_CMPDA_T2	PWMx_BA+0x58	R/W	PWM比较寄存器2	0x0000_0000
PWM_CMPDA_T3	PWMx_BA+0x5C	R/W	PWM比较寄存器3	0x0000_0000
PWM_CMPDA_T4	PWMx_BA+0x60	R/W	PWM比较寄存器4	0x0000_0000
PWM_CMPDA_T5	PWMx_BA+0x64	R/W	PWM比较寄存器5	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	CMPDAT	<p>PWM比较寄存器</p> <p>CMPDAT用于与CNT比较来产生PWM波形，中断和触发ADC。</p> <p>独立模式，CMPDAT0~5作为6个独立PWM_CH0~5比较点。</p> <p>互补模式， CMPDAT0, 2, 4作为第一比较点而CMPDAT1, 3, 5作为第二比较点对应于相应的三个互补组PWM_CH0 和 PWM_CH1, PWM_CH2 和 PWM_CH3, PWM_CH4 和 PWM_CH5。</p>



PWM死区控制寄存器 0_1, 2_3, 4_5 (PWM_DTCTL0_1, 2_3, 4_5)

寄存器	偏移量	R/W	描述	复位后值
PWM_DTCTL_0_1	PWMx_BA+0x70	R/W	PWM死区时间控制寄存器0_1	0x0000_0000
PWM_DTCTL_2_3	PWMx_BA+0x74	R/W	PWM死区时间控制寄存器2_3	0x0000_0000
PWM_DTCTL_4_5	PWMx_BA+0x78	R/W	PWM死区时间控制寄存器4_5	0x0000_0000

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

位	描述
[31:25]	Reserved 保留
[24]	DTCKSEL 死区时钟选择（写保护） 0 = 死区时钟源来自 PWM_CLK. 1 = 死区时钟源来自预分频输出 注：该寄存器写保护，参考REGWRPROT寄存器
[23:17]	Reserved 保留.
[16]	DTEN 使能PWM组死区插入 (PWM_CH0, PWM_CH1) (PWM_CH2, PWM_CH3) (PWM_CH4, PWM_CH5) (写保护) 死区插入只有当该组互补PWM使能才激活。如果死区插入未激活，该互补组输出管脚没有任何延时。 0 = 禁止管脚组死区插入。 1 = 使能管脚组死区插入。 注：该寄存器写保护，参考REGWRPROT寄存器
[15:12]	Reserved 保留.
[11:0]	DTCNT 死区计数器（写保护） 死区时间可以根据以下公式计算： 死区时间=(DTCNT[11:0]+1) * PWM_CLK 周期 注：该寄存器写保护，参考REGWRPROT寄存器



PWM 计数寄存器0, 2, 4 (PWM_CNT0, 2, 4)

寄存器	偏移量	R/W	描述	复位后值
PWM_CNT0	PWMx_BA+0x90	R	PWM 计数寄存器 0	0x0000_0000
PWM_CNT2	PWMx_BA+0x98	R	PWM计数寄存器 2	0x0000_0000
PWM_CNT4	PWMx_BA+0xA0	R	PWM计数寄存器 4	0x0000_0000

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

位	描述	
[31:17]	Reserved	保留
[16]	DIRF	PWM方向标志(只读) 0 =向下计数 1 =向上计数
[15:0]	CNT	PWM数据寄存器 (只读) 用户可以从CNT检测到16位周期计数器值。



PWM发生寄存器0 (PWM_WGCTL0)

寄存器	偏移量	R/W	描述	复位后值
PWM_WGCTL0	PWMx_BA+0xB0	R/W	PWM发生寄存器0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PRDPCTL5		PRDPCTL4	
23	22	21	20	19	18	17	16
PRDPCTL3		PRDPCTL2		PRDPCTL1		PRDPCTL0	
15	14	13	12	11	10	9	8
Reserved				ZPCTL5		ZPCTL4	
7	6	5	4	3	2	1	0
ZPCTL3		ZPCTL2		ZPCTL1		ZPCTL0	

位	描述	
[31:28]	Reserved	保留
[27:16]	PRDPCTLn	<p>PWM周期（中心）点控制</p> <p>每位n控制相应PWM通道n</p> <p>00 =不变 01 = PWM周期（中心）点输出低 10 = PWM周期（中心）点输出高 11 = PWM周期（中心）点输出翻转</p> <p>当PWM计数器计数到(PERIODn+1)PWM可以控制输出电平</p> <p>注：当PWM寄存器工作在上下计数方式，该位是中心点控制。</p>
[15:12]	Reserved	保留.
[11:0]	ZPCTLn	<p>PWM零点控制</p> <p>每位n控制相应PWM通道n</p> <p>00 =不变 01 = PWM零点输出低 10 = PWM零点输出高 11 = PWM零点输出翻转</p> <p>当PWM计数器计数到零点PWM可以控制输出电平</p>



PWM发生寄存器1 (PWM_WGCTL1)

寄存器	偏移量	R/W	描述	复位后值
PWM_WGCTL1	PWMx_BA+0xB4	R/W	PWM发生寄存器1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPDCTL5		CMPDCTL4	
23	22	21	20	19	18	17	16
CMPDCTL3		CMPDCTL2		CMPDCTL1		CMPDCTL0	
15	14	13	12	11	10	9	8
Reserved				CMPUCTL5		CMPUCTL4	
7	6	5	4	3	2	1	0
CMPUCTL3		CMPUCTL2		CMPUCTL1		CMPUCTL0	

位	描述	
[31:28]	Reserved	保留
[27:16]	CMPDCTL_n	<p>PWM 向下比较点控制 每位n控制相应PWM通道n 00 =不变 01 = PWM向下比较点输出低 10 = PWM向下比较点输出高 11 =向下比较点输出翻转 当寄存器向下计数到CMPDAT,PWM可控制输出电平 注：在互补模式， CMPDCTL 1,3,5没有用，这时的输出由CMPDCTL 0,2,4控制</p>
[15:12]	Reserved	保留.
[11:0]	CMPUCTL_n	<p>PWM向上比较点控制 每位n控制相应PWM通道n 00 =不变 01 = PWM 向上比较点输出低 10 = PWM 向上比较点输出高 11 = PWM向上比较点输出翻转 当寄存器向上计数到CMPDAT,PWM可控制输出电平 注：在互补模式， CMPUCTL 1,3,5没有用，这时的输出由CMPUCTL 0, 2,4控制</p>

PWM屏蔽使能寄存器(PWM_MSKEN)

寄存器	偏移量	R/W	描述	复位后值
PWM_MSKEN	PWMx_BA+0xB8	R/W	PWM掩码使能寄存器	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		MSKEN5	MSKEN4	MSKEN3	MSKEN2	MSKEN1	MSKEN0

位	描述	
[31:6]	Reserved	保留
[5:0]	MSKENn	<p>PWM屏蔽使能</p> <p>每位n控制相应PWM通道n</p> <p>当该位使能，PWM输出信号将被屏蔽。相应的PWM通道n将输出MSKDATn (PWM_MSK[5:0])数据</p> <p>0 = 输出信号不屏蔽</p> <p>1 = PWM输出信号被屏蔽并输出MSKDATn数据</p>



PWM 屏蔽 数据寄存器(PWM_MSK)

寄存器	偏移量	R/W	描述	复位后值
PWM_MSK	PWMx_BA+0xBC	R/W	PWM屏蔽数据寄存器	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
Reserved		MSKDAT5	MSKDAT4	MSKDAT3	MSKDAT2	MSKDAT1	MSKDAT0

位	描述	
[31:6]	Reserved	保留
[5:0]	MSKDATn	<p>PWM屏蔽数据位</p> <p>如果相应屏蔽功能使能，该数据位控制PWMn输出管脚状态。每位n控制相应PWM通道n</p> <p>0 = 输出逻辑低到PWMn</p> <p>1 = 输出逻辑高到PWMn</p>



PWM刹车噪音滤波寄存器(PWM_BNF)

寄存器	偏移量	R/W	描述	复位后值
PWM_BNF	PWMx_BA+0xC0	R/W	PWM刹车噪音滤波寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							BK1SRC
23	22	21	20	19	18	17	16
Reserved							BK0SRC
15	14	13	12	11	10	9	8
BRK1PINV	BRK1FCNT			BRK1FCS			BRK1FEN
7	6	5	4	3	2	1	0
BRK0PINV	BRK0FCNT			BRK0FCS			BRK0FEN

位	描述	
[31:25]	Reserved	保留
[24]	BK1SRC	<p>刹车1管脚源选择 对于PWM0设定： 0 = 刹车1管脚源来自PWM0_BRAKE1 1 = 刹车1管脚源来自PWM1_BRAKE1. 对于PWM1设定： 0 = 刹车1管脚源来自PWM1_BRAKE1. 1 = 刹车1管脚源来自PWM0_BRAKE1</p>
[23:17]	Reserved	保留.
[16]	BK0SRC	<p>刹车0管脚源选择 对于PWM0设定： 0 = 刹车0管脚源来自PWM0_BRAKE0 1 = 刹车0管脚源来自PWM1_BRAKE0 对于PWM1设定： 0 = 刹车0管脚源来自PWM1_BRAKE0 1 = 刹车0管脚源来自PWM0_BRAKE0</p>
[15]	BRK1PINV	<p>刹车1管脚翻转 0 = PWMx_BRAKE1管脚状态被传到负边缘检测器。 1 = PWMx_BRAKE1管脚翻转状态被传到负边缘检测器。</p>
[14:12]	BRK1FCNT	刹车1边缘检测滤波器计数 寄存器位控制刹车1滤波计数器从0计数到BRK1FCNT
[11:9]	BRK1FCS	刹车1边缘检测滤波器时钟选择 000 = 滤波器时钟 = HCLK.

		001 = 滤波器时钟 = HCLK/2. 010 = 滤波器时钟 = HCLK/4. 011 = 滤波器时钟 = HCLK/8. 100 = 滤波器时钟 = HCLK/16. 101 = 滤波器时钟 = HCLK/32. 110 = 滤波器时钟 = HCLK/64. 111 = 滤波器时钟 = HCLK/128.
[8]	BRK1FEN	刹车1噪音滤波器使能 0 = 禁止PWM刹车1噪音滤波器 1 = 使能PWM刹车1噪音滤波器
[7]	BRK0PINV	刹车0管脚翻转 0 = PWMx_BRAKE0管脚状态被传到负边缘检测器。 1 = PWMx_BRAKE0管脚翻转状态被传到负边缘检测器。
[6:4]	BRK0FCNT	刹车0边缘检测滤波器计数 寄存器位控制刹车1滤波计数器从0计数到BRK0FCNT
[3:1]	BRK0FCS	刹车0边缘检测滤波器时钟选择 000 = 滤波器时钟 = HCLK. 001 = 滤波器时钟 = HCLK/2. 010 = 滤波器时钟 = HCLK/4. 011 = 滤波器时钟 = HCLK/8. 100 = 滤波器时钟 = HCLK/16. 101 = 滤波器时钟 = HCLK/32. 110 = 滤波器时钟 = HCLK/64. 111 = 滤波器时钟 = HCLK/128.
[0]	BRK0FEN	PWM刹车0噪音滤波器使能 0 = 禁止PWM刹车0噪音滤波器 1 = 使能PWM刹车0噪音滤波器



PWM系统故障刹车控制寄存器 (PWM_FAILBRK)

寄存器	偏移量	R/W	描述	复位后值
PWM_FAILBRK	PWMx_BA+0xC4	R/W	PWM系统故障刹车控制寄存器	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				CORBRKEN	Reserved	BODBRKEN	CSSBRKEN

位	描述	
[31:4]	Reserved	保留
[3]	CORBRKEN	内核锁死检测触发PWM刹车功能0使能 0 = 禁止通过内核锁死检测触发刹车功能 1 = 使能通过内核锁死检测触发刹车功能
[2]	Reserved	保留.
[1]	BODBRKEN	欠压检测触发PWM刹车功能0使能 0 = 禁止BOD触发刹车功能 1 = 使能BOD触发刹车功能
[0]	CSSBRKEN	时钟安全系统检测触发PWM刹车功能0使能 0 = 禁止通过CSS检测触发刹车功能 1 = 使能通过CSS检测触发刹车功能



PWM刹车边缘检测控制寄存器 0_1, 2_3, 4_5(PWM_BRKCTL0_1, 2_3, 4_5)

寄存器	偏移量	R/W	描述	复位后值
PWM_BRKCTL0_1	PWMx_BA+0xC8	R/W	PWM 刹车边缘检测控制寄存器 0_1	0x0000_0000
PWM_BRKCTL2_3	PWMx_BA+0xCC	R/W	PWM 刹车边缘检测控制寄存器 2_3	0x0000_0000
PWM_BRKCTL4_5	PWMx_BA+0xD0	R/W	PWM 刹车边缘检测控制寄存器4_5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				BRKAODD		BRKAEVEN	
15	14	13	12	11	10	9	8
SYSLEN	Reserved	BRKP1LEN	BRKP0LEN	Reserved			
7	6	5	4	3	2	1	0
SYSEEN	Reserved	BRKP1EEN	BRKP0EEN	Reserved			

位	描述	
[31:20]	Reserved	保留
[19:18]	BRKAODD	PWM奇数通道刹车行为选择 (写保护) 00 = PWM奇数通道电平检测刹车功能不影响信道输出。 01 = 当电平检测刹车发生，PWM奇数通道三态输出。 10 =当电平检测刹车发生，PWM奇数通道低电平输出。 11 =当电平检测刹车发生，PWM奇数通道高电平输出。 注：该寄存器写保护，参考REGWRPROT寄存器
[17:16]	BRKAEVEN	PWM偶数通道刹车作用选择 (写保护) 00 = PWM偶数通道电平检测刹车功能不影响信道输出。 01 =当电平检测刹车发生，PWM偶数通道三态输出。 10 =当电平检测刹车发生，PWM偶数通道低电平输出。 11 =当电平检测刹车发生，PWM偶数通道高电平输出。 注：该寄存器写保护，参考REGWRPROT寄存器
[15]	SYSLEN	使能系统故障作为电平检测刹车源 (写保护) 0 =禁止系统故障条件作为电平检测刹车源 1 =使能系统故障条件作为电平检测刹车源 注：该寄存器写保护，参考REGWRPROT寄存器



[14]	Reserved	保留.
[13]	BRKP1LEN	<p>使能管脚BKP1作为电平检测刹车源（写保护）</p> <p>0 = 禁止PWMx_BRAKE1管脚作为电平检测刹车源 1 = 使能PWMx_BRAKE1管脚作为电平检测刹车源 注：该寄存器写保护，参考REGWRPROT寄存器</p>
[12]	BRKPOLEN	<p>使能管脚BKP0作为电平检测刹车源（写保护）</p> <p>0 = 禁止PWMx_BRAKE0管脚作为电平检测刹车源 1 = 使能PWMx_BRAKE0管脚作为电平检测刹车源 注：该寄存器写保护，参考REGWRPROT寄存器</p>
[11:8]	Reserved	保留.
[7]	SYSEEN	<p>使能系统故障作为边缘检测刹车源（写保护）</p> <p>0 = 禁止系统故障条件作为边缘检测刹车源 1 = 使能系统故障条件作为边缘检测刹车源 注：该寄存器写保护，参考REGWRPROT寄存器</p>
[6]	Reserved	保留.
[5]	BRKP1EEN	<p>使能PWMx_BRAKE1管脚作为边缘检测刹车源（写保护）</p> <p>0 = 禁止BKP1管脚作为边缘检测刹车源 1 = 使能BKP1管脚作为边缘检测刹车源 注：该寄存器写保护，参考REGWRPROT寄存器</p>
[4]	BRKPOEEN	<p>使能PWMx_BRAKE0管脚作为边缘检测刹车源（写保护）</p> <p>0 = 禁止BKPO管脚作为边缘检测刹车源 1 = 使能BKPO管脚作为边缘检测刹车源 注：该寄存器写保护，参考REGWRPROT寄存器</p>
[3:0]	Reserved	保留.

PWM管脚极性反转控制 (PWM_POLCTL)

寄存器	偏移量	R/W	描述	复位后值
PWM_POLCTL_L	PWMx_BA+0xD4	R/W	PWM 管脚极性反转寄存器	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		PINV5	PINV4	PINV3	PINV2	PINV1	PINV0

位	描述	
[31:6]	Reserved	保留.
[5:0]	PINVn	PWM管脚极性反转控制 该寄存器控制PWM输出的极性状态。每位n控制相应PWM通道n 0 = 禁止PWM输出极性反转 1 = 使能PWM输出极性反转

PWM输出使能寄存器 (PWM_POEN)

寄存器	偏移量	R/W	描述	复位后值
PWM_POEN	PWMx_BA+0xD8	R/W	PWM 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		POEN5	POEN4	POEN3	POEN2	POEN1	POEN0

位	描述	
[31:6]	Reserved	保留
[5:0]	POENn	PWM管脚输出使能 每位n控制相应PWM通道n 0 = PWM管脚在三态模式 1 = PWM管脚在输出模式



PWM软件刹车控制寄存器(PWM_SWBRK)

寄存器	偏移量	R/W	描述	复位后值
PWM_SWBRK	PWMx_BA+0xDC	W	PWM软件刹车控制寄存器	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					BRKLTRG4	BRKLTRG2	BRKLTRG0
7	6	5	4	3	2	1	0
Reserved					BRKETRG4	BRKETRG2	BRKETRG0

位	描述	
[31:11]	Reserved	保留.
[10:8]	BRKLTRGn	<p>PWM电平刹车软件触发 (只写) (写保护)</p> <p>每位n控制相应PWM组 n</p> <p>写1到该位将触发电平刹车，并将PWM_INTSTS1寄存器的BRKLIFn置1.</p> <p>注：该寄存器写保护，参考REGWRPROT寄存器</p>
[7:3]	Reserved	保留.
[2:0]	BRKETRGn	<p>PWM边缘刹车软件触发 (只写) (写保护)</p> <p>每位n控制相应PWM组 n</p> <p>写1到该位将触发边缘刹车，并将PWM_INTSTS1寄存器的BRKEIFn置1.</p> <p>注：该寄存器写保护，参考REGWRPROT寄存器</p>



PWM 中断使能寄存器0 (PWM_INTEN0)

寄存器	偏移量	R/W	描述	复位后值
PWM_INTEN0	PWMx_BA+0xE0	R/W	PWM 中断使能寄存器 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIEN5	CMPDIEN4	CMPDIEN3	CMPDIEN2	CMPDIEN1	CMPDIEN0
23	22	21	20	19	18	17	16
Reserved		CMPUIEN5	CMPUIEN4	CMPUIEN3	CMPUIEN2	CMPUIEN1	CMPUIEN0
15	14	13	12	11	10	9	8
Reserved			PIEN4	Reserved	PIEN2	Reserved	PIEN0
7	6	5	4	3	2	1	0
Reserved			ZIEN4	Reserved	ZIEN2	Reserved	ZIEN0

位	描述
[31:30]	Reserved 保留
[29:24]	CMPDIENn PWM比较向下计数中断使能 每位n控制相应PWM通道n 0 =禁止比较向下计数中断 1 =使能比较向下计数中断 注：在互补模式，CMPDIEN 1,3,5没有用，这时的输出由CMPDIEN 0,2,4控制
[23:22]	Reserved.
[21:16]	CMPUIENn PWM比较向上计数中断使能 每位n控制相应PWM通道n 0 =禁止比较向上计数中断 1 =使能比较向上计数中断 注：在互补模式，CMPUIEN11,3,5没有用，这时的输出由CMPUIEN10,2,4,控制
[15:13]	Reserved 保留.
[12]	PIEN4 PWM周期点中断使能4 0 =禁止周期点中断 1 =使能周期点中断 注：上下计数方式是周期点指的是中点。
[11]	Reserved 保留.
[10]	PIEN2 PWM周期点中断使能2 0 =禁止周期点中断

		1 =使能周期点中断 注：上下计数方式是周期点指的是中点。
[9]	Reserved	保留.
[8]	PIEN0	PWM周期点中断使能0 0 =禁止周期点中断 1 =使能周期点中断 注：上下计数方式是周期点指的是中点。
[7:5]	Reserved	保留.
[4]	ZIEN4	PWM零点中断使能 4 0 =禁止零点中断 1 =使能零点中断 注：在互补模式，奇数信道读出一直是0.
[3]	Reserved	保留.
[2]	ZIEN2	PWM零点中断使能2 0 =禁止零点中断 1 =使能零点中断 注：在互补模式，奇数信道读出一直是0.
[1]	Reserved	保留
[0]	ZIENO	PWM零点中断使能 0 0 =禁止零点中断 1 =使能零点中断 注：在互补模式，奇数信道读出一直是0.



PWM中断使能寄存器1 (PWM_INTEN1)

寄存器	偏移量	R/W	描述	复位后值
PWM_INTEN1	PWMx_BA+0xE4	R/W	PWM中断使能寄存器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					BRKLIEN4_5	BRKLIEN2_3	BRKLIENO_1
7	6	5	4	3	2	1	0
Reserved					BRKEIEN4_5	BRKEIEN2_3	BRKEIENO_1

位	描述
[31:11]	Reserved 保留.
[10]	BRKLIEN4_5 PWM通道4/5电平检测刹车中断使能（写保护） 0 =禁止通道4/5电平检测刹车中断 1 =使能通道4/5电平检测刹车中断 注：该寄存器写保护，参考REGWRPROT寄存器
[9]	BRKLIEN2_3 PWM通道2/3电平检测刹车中断使能（写保护） 0 =禁止通道2/3电平检测刹车中断 1 =使能通道2/3电平检测刹车中断 注：该寄存器写保护，参考REGWRPROT寄存器
[8]	BRKLIENO_1 PWM通道0/1电平检测刹车中断使能（写保护） 0 =禁止通道0/1电平检测刹车中断 1 =使能通道0/1电平检测刹车中断 注：该寄存器写保护，参考REGWRPROT寄存器
[7:3]	Reserved 保留.
[2]	BRKEIEN4_5 PWM通道4/5边缘检测刹车中断使能（写保护） 0 =禁止通道4/5边缘检测刹车中断 1 =使能通道4/5边缘检测刹车中断 注：该寄存器写保护，参考REGWRPROT寄存器
[1]	BRKEIEN2_3 PWM通道2/3边缘检测刹车中断使能（写保护） 0 =禁止通道2/3边缘检测刹车中断

		1 = 使能通道2/3边缘检测刹车中断 注：该寄存器写保护，参考REGWRPROT寄存器
[0]	BRKEIENO_1	PWM通道0/1边缘检测刹车中断使能（写保护） 0 = 禁止通道0/1边缘检测刹车中断 1 = 使能通道0/1边缘检测刹车中断 注：该寄存器写保护，参考REGWRPROT寄存器



PWM中断标志寄存器0 (PWM_INTSTS0)

寄存器	偏移量	R/W	描述	复位后值
PWM_INTSTS0	PWMx_BA+0xE8	R/W	PWM 中断标志寄存器 0	0x0000_0000

31	30	29	28	27	26	25	24
	Reserved	CMPDIF5	CMPDIF4	CMPDIF3	CMPDIF2	CMPDIF1	CMPDIF0
23	22	21	20	19	18	17	16
	Reserved	CMPUIF5	CMPUIF4	CMPUIF3	CMPUIF2	CMPUIF1	CMPUIF0
15	14	13	12	11	10	9	8
	Reserved		PIF4	Reserved	PIF2	Reserved	PIF0
7	6	5	4	3	2	1	0
	Reserved		ZIF4	Reserved	ZIF2	Reserved	ZIF0

位	描述	
[31:30]	Reserved	保留
[29:24]	CMPDIFn	<p>PWM比较向下计数中断标志</p> <p>每位n控制相应PWM通道n</p> <p>当PWM计数器向下计数到PWM_CMPDATn，标志被硬件置1。软件写1到该位将清标志。</p> <p>注1：如果CMPDAT 等于PERIOD，向下计数方式下该位不工作。</p> <p>注2：在互补模式，CMPDIF 1,3,5没有用，这时的输出由CMPDIF0, 2,4控制</p>
[23:22]	Reserved	保留.
[21:16]	CMPUIFn	<p>PWM比较向上计数中断标志</p> <p>当PWM计数器向上计数到PWM_CMPDATn，标志被硬件置1。软件写1到该位将清标志。</p> <p>每位n控制相应PWM通道n</p> <p>注1：如果CMPDAT 等于PERIOD，向上计数模式下该位不工作。</p> <p>注2：在互补模式，CMPUIF 1,3,5没有用，这时的输出由CMPUIF 0, 2,4控制</p>
[15:13]	Reserved	保留.
[12]	PIF4	<p>PWM周期点中断标志 4</p> <p>当PWM_CH4计数器到 PWM_PERIOD4硬件置该位，软件写1清该位到零。</p>
[11]	Reserved	保留.
[10]	PIF2	<p>PWM周期点中断标志2</p> <p>当PWM_CH2计数器到 PWM_PERIOD2硬件置该位，软件写1清该位到零。</p>
[9]	Reserved	保留.



[8]	PIF0	PWM 周期点中断标志0 当PWM_CH0计数器到 PWM_PERIOD0硬件置该位，软件写1清该位到零。
[7:5]	Reserved	保留.
[4]	ZIF4	PWM零点中断标志4 当PWM_CH4计数器到零硬件置该位，软件写1清该位到零。
[3]	Reserved	保留
[2]	ZIF2	PWM零点中断标志 2 当PWM_CH2计数器到零硬件置该位，软件写1清该位到零。
[1]	Reserved	保留.
[0]	ZIF0	PWM 零点中断标志0 当PWM_CH0计数器到零硬件置该位，软件写1清该位到零。



PWM中断标志寄存器1 (PWM_INTSTS1)

寄存器	偏移量	R/W	描述	复位后值
PWM_INTSTS1	PWMx_BA+0xEC	R/W	PWM 中断标志寄存器1	0x0000_0000

31b	30	29	28	27	26	25	24
Reserved	BRKLSTS5	BRKLSTS4	BRKLSTS3	BRKLSTS2	BRKLSTS1	BRKLSTS0	
23	22	21	20	19	18	17	16
Reserved	BRKESTS5	BRKESTS4	BRKESTS3	BRKESTS2	BRKESTS1	BRKESTS0	
15	14	13	12	11	10	9	8
Reserved	BRKLIF5	BRKLIF4	BRKLIF3	BRKLIF2	BRKLIF1	BRKLIF0	
7	6	5	4	3	2	1	0
Reserved	BRKEIF5	BRKEIF4	BRKEIF3	BRKEIF2	BRKEIF1	BRKEIF0	

位	描述	
[31:30]	Reserved	保留
[29]	BRKLSTS5	<p>PWM通道5电平检测刹车状态 (只读) 0 = PWM通道5电平检测刹车状态被释放。 1 = 当PWM通道5电平检测到任何一个已被使能的刹车源有个下降沿时，该位被置1表示PWM通道5处在刹车状态。 注：该位只读并被硬件自动清除。当已使能刹车源恢复到高电平，直到当前周期完成后PWM输出才会释放刹车状态。PWM波形将在下个完整的PWM周期开始输出。</p>
[28]	BRKLSTS4	<p>PWM通道4电平检测刹车状态 (只读) 0 = PWM通道4电平检测刹车状态被释放。 1 = 当PWM通道4电平检测到任何一个已被使能的刹车源有个下降沿时，该位被置1表示PWM通道4处在刹车状态。 注：该位只读并被硬件自动清除。当已使能刹车源恢复到高电平，直到当前周期完成后PWM输出才会释放刹车状态。PWM波形将在下个完整的PWM周期开始输出。</p>
[27]	BRKLSTS3	<p>PWM通道3电平检测刹车状态 (只读) 0 = PWM通道3电平检测刹车状态被释放。 1 = 当PWM通道3电平检测到任何一个已被使能的刹车源有个下降沿时，该位被置1表示PWM通道3处在刹车状态。 注：该位只读并被硬件自动清除。当已使能刹车源恢复到高电平，直到当前周期完成后PWM输出才会释放刹车状态。PWM波形将在下个完整的PWM周期开始输出。</p>
[26]	BRKLSTS2	<p>PWM通道2电平检测刹车状态 (只读) 0 = PWM通道2电平检测刹车状态被释放。 1 = 当PWM通道2电平检测到任何一个已被使能的刹车源有个下降沿时，该位被置1表示PWM通道2处在刹车状态。 注：该位只读并被硬件自动清除。当已使能刹车源恢复到高电平，直到当前周期完成后PWM输出才会释放刹车状态。PWM波形将在下个完整的PWM周期开始输出。</p>

[25]	BRKLSTS1	PWM通道1电平检测刹车状态（只读） 0 = PWM通道1电平检测刹车状态被释放。 1 =当PWM通道1电平检测到任何一个已被使能的刹车源有个下降沿时，该位被置1表示PWM通道1处在刹车状态。 注：该位只读并被硬件自动清除。当已使能刹车源恢复到高电平，直到当前周期完成后PWM输出才会释放刹车状态。PWM波形将在下个完整的PWM周期开始输出。
[24]	BRKLSTS0	PWM通道0电平检测刹车状态（只读） 0 = PWM通道0电平检测刹车状态被释放。 1 =当PWM通道0电平检测到任何一个已被使能的刹车源有个下降沿时，该位被置1表示PWM通道0处在刹车状态。 注：该位只读并被硬件自动清除。当已使能刹车源恢复到高电平，直到当前周期完成后PWM输出才会释放刹车状态。PWM波形将在下个完整的PWM周期开始输出。
[23:22]	Reserved	保留.
[21]	BRKESTS5	PWM通道5边缘检测刹车状态 0 = PWM通道5边缘检测刹车状态释放 1 =当PWM通道5边缘刹车检测到任何被使能的刹车源有一个下降沿时，该位被置1表示通道5处在刹车状态，写1清除。
[20]	BRKESTS4	PWM通道4边缘检测刹车状态 0 = PWM通道4边缘检测刹车状态释放 1 =当PWM通道4边缘刹车检测到任何被使能的刹车源有一个下降沿时，该位被置1表示通道4处在刹车状态，写1清除。
[19]	BRKESTS3	PWM通道3边缘检测刹车状态 0 = PWM通道3边缘检测刹车状态释放 1 =当PWM通道3边缘刹车检测到任何被使能的刹车源有一个下降沿时，该位被置1表示通道3处在刹车状态，写1清除。
[18]	BRKESTS2	PWM通道2边缘检测刹车状态 0 = PWM通道2边缘检测刹车状态释放 1 =当PWM通道2边缘刹车检测到任何被使能的刹车源有一个下降沿时，该位被置1表示通道2处在刹车状态，写1清除。
[17]	BRKESTS1	PWM通道1边缘检测刹车状态 0 = PWM通道1边缘检测刹车状态释放 1 =当PWM通道1边缘刹车检测到任何被使能的刹车源有一个下降沿时，该位被置1表示通道1处在刹车状态，写1清除。
[16]	BRKESTS0	PWM通道0边缘检测刹车状态 0 = PWM通道0边缘检测刹车状态释放 1 =当PWM通道0边缘刹车检测到任何被使能的刹车源有一个下降沿时，该位被置1表示通道0处在刹车状态，写1清除。
[15:14]	Reserved	保留.
[13]	BRKLIF5	PWM通道5电平检测刹车中断标志（写保护） 0 = PWM通道5电平检测刹车事件没发生 1 =当PWM通道5电平检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器



[12]	BRKLIF4	PWM通道4电平检测刹车中断标志（写保护） 0 = WM通道4电平检测刹车事件没发生 1 =当PWM通道4电平检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[11]	BRKLIF3	PWM通道3电平检测刹车中断标志（写保护） 0 = PWM通道3电平检测刹车事件没发生 1 =当PWM通道3电平检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[10]	BRKLIF2	PWM 通道2电平检测刹车中断标志（写保护） 0 = PWM通道2电平检测刹车事件没发生 1 = 当PWM通道2电平检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[9]	BRKLIF1	PWM 通道1电平检测刹车中断标志（写保护） 0 =PWM通道1电平检测刹车事件没发生 1 =当PWM通道1电平检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[8]	BRKLIF0	PWM 通道0电平检测刹车中断标志（写保护） 0 =PWM通道0电平检测刹车事件没发生 1 = 当PWM通道0电平检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[7:6]	Reserved	保留.
[5]	BRKEIF5	PWM通道5边缘检测刹车中断标志（写保护） 0 = PWM通道5边缘检测刹车事件没发生 1 =当PWM通道5边缘检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[4]	BRKEIF4	PWM 通道4边缘检测刹车中断标志（写保护） 0 = 通道4边缘检测刹车事件没发生 1 =当PWM通道4边缘检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[3]	BRKEIF3	PWM通道3边缘检测刹车中断标志（写保护） 0 = PWM通道3边缘检测刹车事件没发生 1 =当PWM通道3边缘检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[2]	BRKEIF2	PWM通道2边缘检测刹车中断标志（写保护） 0 = PWM通道2边缘检测刹车事件没发生 1 =当PWM通道2边缘检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器
[1]	BRKEIF1	PWM通道1边缘检测刹车中断标志（写保护） 0 =通道1边缘检测刹车事件没发生 1 = 当PWM通道1边缘检测中断事件发生，该位被置1，写1清除。

		注：该寄存器写保护，参考REGWRPROT寄存器
[0]	BRKEIF0	PWM通道0边缘检测刹车中断标志（写保护） 0 = PWM通道0边缘检测刹车事件没发生 1 = 当PWM通道0边缘检测中断事件发生，该位被置1，写1清除。 注：该寄存器写保护，参考REGWRPROT寄存器



PWM触发ADC源选择寄存器0 (PWM_ADCTS0)

寄存器	偏移量	R/W	描述	复位后值
PWM_ADCTS0	PWMx_BA+0xF8	R/W	PWM 触发ADC源选择寄存器 0	0x0000_0000

31	30	29	28	27	26	25	24
TRGEN3	Reserved			TRGSEL3			
23	22	21	20	19	18	17	16
TRGEN2	Reserved			TRGSEL2			
15	14	13	12	11	10	9	8
TRGEN1	Reserved			TRGSEL1			
7	6	5	4	3	2	1	0
TRGEN0	Reserved			TRGSEL0			

位	描述	
[31]	TRGEN3	PWM_CH3触发ADC使能位
[30:28]	Reserved	保留.
[27:24]	TRGSEL3	PWM_CH3触发ADC源选择 0000 = PWM_CH2 零点. 0001 = PWM_CH2 周期点 0010 = PWM_CH2 零或周期点 0011 = PWM_CH2 向上计数 CMPDAT 点. 0100 = PWM_CH2 向下计数 CMPDAT 点. 0101 = 保留. 0110 = 保留. 0111 = 保留. 1000 = PWM_CH3 向上计数 CMPDAT 点. 1001 = PWM_CH3 向下计数 CMPDAT 点. 其他 = 保留.
[23]	TRGEN2	PWM_CH2触发ADC使能位
[22:20]	Reserved	保留.
[19:16]	TRGSEL2	PWM_CH2触发ADC源选择 0000 = PWM_CH2 零点. 0001 = PWM_CH2 周期点 0010 = PWM_CH2 零或周期点

		<p>0011 = PWM_CH2 向上计数 CMPDAT 点.</p> <p>0100 = PWM_CH2 向下计数 CMPDAT 点.</p> <p>0101 = 保留.</p> <p>0110 = 保留.</p> <p>0111 = 保留.</p> <p>1000 = PWM_CH3 向上计数 CMPDAT 点.</p> <p>1001 = PWM_CH3 向下计数 CMPDAT 点.</p> <p>其他 = 保留.</p>
[15]	TRGEN1	PWM_CH1触发ADC使能位
[14:12]	Reserved	保留.
[11:8]	TRGSEL1	<p>PWM_CH1触发ADC源选择</p> <p>0000 = PWM_CH0 零点.</p> <p>0001 = PWM_CH0 周期点</p> <p>0010 = PWM_CH0 零或周期点</p> <p>0011 = PWM_CH0 向上计数 CMPDAT 点.</p> <p>0100 = PWM_CH0 向下计数 CMPDAT 点.</p> <p>0101 = 保留.</p> <p>0110 = 保留.</p> <p>0111 = 保留.</p> <p>1000 = PWM_CH1 向上计数 CMPDAT 点.</p> <p>1001 = PWM_CH1 向下计数 CMPDAT 点.</p> <p>其他 = 保留.</p>
[7]	TRGEN0	PWM_CH0触发ADC使能位
[6:4]	Reserved	保留.
[3:0]	TRGSEL0	<p>PWM_CH0触发ADC源选择</p> <p>0000 = PWM_CH0 零点.</p> <p>0001 = PWM_CH0 周期点</p> <p>0010 = PWM_CH0 零或周期点</p> <p>0011 = PWM_CH0 向上计数 CMPDAT 点.</p> <p>0100 = PWM_CH0 向下计数 CMPDAT 点.</p> <p>0101 = 保留.</p> <p>0110 = 保留.</p> <p>0111 = 保留.</p> <p>1000 = PWM_CH1 向上计数 CMPDAT 点.</p> <p>1001 = PWM_CH1 向下计数 CMPDAT 点.</p> <p>其他 = 保留.</p>



PWM触发ADC源选择寄存器1 (PWM_ADCTS1)

Register	Offset	R/W	Description					Reset Value
PWM_ADCTS1	PWMx_BA+0xFC	R/W	PWM 触发ADC源选择寄存器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
TRGEN5	Reserved			TRGSEL5			
7	6	5	4	3	2	1	0
TRGEN4	Reserved			TRGSEL4			

位	描述	
[31:16]	Reserved	保留
[15]	TRGEN5	PWM_CH5触发ADC使能位
[14:12]	Reserved	保留.
[11:8]	TRGSEL5	PWM_CH5触发ADC源选择 0000 = PWM_CH4 零点. 0001 = PWM_CH4 周期点 0010 = PWM_CH4 零或周期点 0011 = PWM_CH4 向上计数 CMPDAT 点. 0100 = PWM_CH4 向下计数 CMPDAT 点. 0101 = 保留. 0110 = 保留. 0111 = 保留. 1000 = PWM_CH5 向上计数 CMPDAT 点. 1001 = PWM_CH5 向下计数 CMPDAT 点. 其他 = 保留.
[7]	TRGEN4	PWM_CH4触发ADC使能位
[6:4]	Reserved	保留.
[3:0]	TRGSEL4	PWM_CH4触发ADC源选择 0000 = PWM_CH4 零点. 0001 = PWM_CH4 周期点 0010 = PWM_CH4 零或周期点 0011 = PWM_CH4 向上计数 CMPDAT 点.

		<p>0100 = PWM_CH4 向下计数 CMPDAT 点. 0101 = 保留. 0110 = 保留. 0111 = 保留. 1000 = PWM_CH5 向上计数 CMPDAT 点. 1001 = PWM_CH5 向下计数 CMPDAT 点. 其他 = 保留.</p>
--	--	--



PWM同步开始寄存器(PWM_SSCTL)

寄存器	偏移量	R/W	描述	复位后值
PWM_SSCTL	PWMx_BA+0x110	R/W	PWM 同步开始寄存器	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						SSRC	
7	6	5	4	3	2	1	0
Reserved			SSEN4	Reserved	SSEN2	Reserved	SSEN0

位	描述	
[31:10]	Reserved	保留
[9:8]	SSRC	PWM 同步开始源选择 00 = 同步开始源来自PWMO. 01 = 同步开始源来自PWM1. 10 = 同步开始源来自BPWMO. 11 = 同步开始源来自 BPWM1.
[7:5]	Reserved	保留.
[4]	SSEN4	PWM 同步开始功能使能4 当同步开始功能被使能, PWM_CH4计数器使能位(CNTEN4)可以通过写PWM同步开始触发位(CNTSEN)来使能 0 = 禁止同步开始功能 1 = 使能同步开始功能
[3]	Reserved	保留.
[2]	SSEN2	PWM 同步开始功能使能2 当同步开始功能被使能, PWM_CH2计数器使能位(CNTEN2)可以通过写PWM同步开始触发位(CNTSEN)来使能 0 = 禁止同步开始功能 1 = 使能同步开始功能
[1]	Reserved	保留.
[0]	SSEN0	PWM同步开始功能使能0 当同步开始功能被使能, PWM_CH0计数器使能位(CNTENO)可以通过写PWM同步开始触发位(CNTSEN)来使能 0 = 禁止同步开始功能

		1 =使能同步开始功能
--	--	-------------



PWM 同步开始触发寄存器(PWM_SSTRG)

寄存器	偏移量	R/W	描述	复位后值
PWM_SSTRG	PWMx_BA+0x114	W	PWM 同步开始触发寄存器	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							CNTSEN

位	描述	
[31:1]	Reserved	保留
[0]	CNTSEN	<p>PWM计数器同步开始使能（只读）</p> <p>PWM 计数器同步功能使能是用于使被选的PWM通道(包括 PWM0_CHx 和 PWM1_CHx)开始同时计数。</p> <p>如果 相应的PWM通道计数器同步开始功能被使能，写该位1，也同时设置了计数器使能位(CNTENn, n 表示通道 0 到 5)</p> <p>注：该位目前只在PWM0_BA.</p>



PWM状态寄存器(PWM_STATUS)

寄存器	偏移量	R/W	描述	复位后值
PWM_STATUS	PWMx_BA+0x120	R/W	PWM 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		ADCTRG5	ADCTRG4	ADCTRG3	ADCTRG2	ADCTRG1	ADCTRG0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			CNTMAX4	Reserved	CNTMAX2	Reserved	CNTMAX0

位	描述	
[31:22]	Reserved	保留
[21:16]	ADCTRGn	<p>ADC转换状态开始 每位n控制相应的PWM通道n。 0 = 表明ADC转换触发事件没有发生 1 = 表明ADC转换触发事件发生了，软件可以写1清该位。</p>
[15:5]	Reserved	保留.
[4]	CNTMAX4	<p>计数器4等于0xFFFF锁存状态 0 = 表明计数器从没到达它最大值0xFFFF 1 = 表明计数器到达它最大值，软件可以写1清该位</p>
[3]	Reserved	保留.
[2]	CNTMAX2	<p>计数器2等于0xFFFF锁存状态 0 = 表明计数器从没到达它最大值0xFFFF 1 = 表明计数器到达它最大值，软件可以写1清该位</p>
[1]	Reserved	保留.
[0]	CNTMAX0	<p>计数器0等于0xFFFF锁存状态 0 = 表明计数器从没到达它最大值0xFFFF 1 = 表明计数器到达它最大值，软件可以写1清该位</p>



PWM输入捕捉使能寄存器 (PWM_CAPINEN)

寄存器	偏移量	R/W	描述	复位后值
PWM_CAPIN EN	PWMx_BA+0x200	R/W	PWM 输入捕捉使能寄存器	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		CAPINEN5	CAPINEN4	CAPINEN3	CAPINEN2	CAPINEN1	CAPINEN0

位	描述	
[31:6]	Reserved	保留.
[5:0]	CAPINENn	<p>输入捕捉使能</p> <p>每位n控制相应的PWM通道n。</p> <p>0 =禁止PWM输入捕捉， PWM信道捕捉功能的输入总是被认为是0;</p> <p>1 =使能PWM输入捕捉， PWM信道捕捉功能的输入来自相关的复用管脚。</p>



PWM捕捉控制寄存器 (PWM_CAPCTL)

寄存器	偏移量	R/W	描述	复位后值
PWM_CAPCTL	PWMx_BA+0x204	R/W	PWM 捕捉控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	FCRLDEN5	FCRLDEN4	FCRLDEN3	FCRLDEN2	FCRLDEN1	FCRLDEN0	
23	22	21	20	19	18	17	16
Reserved	RCRLDEN5	RCRLDEN4	RCRLDEN3	RCRLDEN2	RCRLDEN1	RCRLDEN0	
15	14	13	12	11	10	9	8
Reserved	CAPINV5	CAPINV4	CAPINV3	CAPINV2	CAPINV1	CAPINV0	
7	6	5	4	3	2	1	0
Reserved	CAPEN5	CAPEN4	CAPEN3	CAPEN2	CAPEN1	CAPEN0	

位	描述
[31:30]	Reserved 保留
[29:24]	FCRLDENn 下降沿捕捉重载使能 每位n控制相应的PWM通道n。 0 = 禁止下降沿捕捉重载计数器 1 = 使能下降沿捕捉重载计数器
[23:22]	Reserved 保留.
[21:16]	RCRLDENn 上升沿捕捉重载使能 每位n控制相应的PWM通道n。 0 = 禁止上升沿捕捉重载计数器 1 = 使能上升沿捕捉重载计数器
[15:14]	Reserved 保留.
[13:8]	CAPINVn 捕捉反向使能 每位n控制相应的PWM通道n。 0 = 禁止捕捉源反向 1 = 使能捕捉源反向，将来自GPIO的信号反向
[7:6]	Reserved 保留.
[5:0]	CAPENn 捕捉功能使能 每位n控制相应的PWM通道n。 0 = 禁止捕捉功能。RCAPDAT/FCAPDA不会更新。 1 = 使能捕捉功能。当检测到输入信号的上升/下降沿时锁存PWM计数器值并保存到RCAPDAT（向上锁存）和FCAPDAT（向下锁存）。



PWM 捕捉状态寄存器(PWM_CAPSTS)

寄存器	偏移量	R/W	描述	复位后值
PWM_CAPSTS	PWMx_BA+0x208	R	PWM 捕捉寄存器	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		CFIFOV5	CFIFOV4	CFIFOV3	CFIFOV2	CFIFOV1	CFIFOV0
7	6	5	4	3	2	1	0
Reserved		CRIFOV5	CRIFOV4	CRIFOV3	CRIFOV2	CRIFOV1	CRIFOV0

位	描述	
[31:14]	Reserved	保留
[13:8]	CFIFOVn	<p>下降沿捕捉中断溢出状态（只读）</p> <p>如果相应的CAPFIF是1而又发生了下降沿捕捉事件此标志为1。每位n控制相应的PWM通道n。</p> <p>注：当用户清相应的CAPFIF，该位将自动清零。</p>
[7:6]	Reserved	保留.
[5:0]	CRIFOVn	<p>上升沿捕捉中断溢出状态（只读）</p> <p>如果相应的CAPRIF是1而又发生了上升沿捕捉事件此标志为1。每位n控制相应的PWM通道n。</p> <p>注：当用户清相应的CAPRIF，该位将自动清零。</p>



PWM上升沿捕捉数据寄存器 0~5 (PWM_RCAPDAT 0~5)

寄存器	偏移量	R/W	描述	复位后值
PWM_RCAPD AT0	PWMx_BA+0x20C	R	PWM 上升沿捕捉数据寄存器0	0x0000_0000
PWM_RCAPD AT1	PWMx_BA+0x214	R	PWM上升沿捕捉数据寄存器1	0x0000_0000
PWM_RCAPD AT2	PWMx_BA+0x21C	R	PWM上升沿捕捉数据寄存器2	0x0000_0000
PWM_RCAPD AT3	PWMx_BA+0x224	R	PWM上升沿捕捉数据寄存器3	0x0000_0000
PWM_RCAPD AT4	PWMx_BA+0x22C	R	PWM上升沿捕捉数据寄存器4	0x0000_0000
PWM_RCAPD AT5	PWMx_BA+0x234	R	PWM上升沿捕捉数据寄存器5	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留.
[15:0]	RCAPDAT	PWM上升沿捕捉数据寄存器（只读） 当上升沿捕捉条件发生， PWM计数器值将被保存到该寄存器。



PWM下降沿捕捉数据寄存器 0~5 (PWM_FCAPDAT 0~5)

寄存器	偏移量	R/W	描述	复位后值
PWM_FCAPPD_AT0	PWMx_BA+0x210	R	PWM 下降沿捕捉寄存器 0	0x0000_0000
PWM_FCAPPD_AT1	PWMx_BA+0x218	R	PWM 下降沿捕捉寄存器 1	0x0000_0000
PWM_FCAPPD_AT2	PWMx_BA+0x220	R	PWM 下降沿捕捉寄存器 2	0x0000_0000
PWM_FCAPPD_AT3	PWMx_BA+0x228	R	PWM 下降沿捕捉寄存器 3	0x0000_0000
PWM_FCAPPD_AT4	PWMx_BA+0x230	R	PWM 下降沿捕捉寄存器 4	0x0000_0000
PWM_FCAPPD_AT5	PWMx_BA+0x238	R	PWM 下降沿捕捉寄存器 5	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	FCAPDAT	PWM下降沿捕捉寄存器(只读) 当下降沿捕捉条件发生， PWM计数器值将被保存到该寄存器。



PWM捕捉中断使能寄存器(PWM_CAPIEN)

寄存器	偏移量	R/W	描述	复位后值
PWM_CAPIEN	PWMx_BA+0x250	R/W	PWM 捕捉中断使能寄存器	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		CAPFIEN5	CAPFIEN4	CAPFIEN3	CAPFIEN2	CAPFIEN1	CAPFIENO
7	6	5	4	3	2	1	0
Reserved		CAPRIEN5	CAPRIEN4	CAPRIEN3	CAPRIEN2	CAPRIEN1	CAPRIENO

位	描述	
[31:14]	Reserved	保留
[13:8]	CAPFIENn	<p>PWM 下降沿捕捉锁存中断使能</p> <p>每位n控制相应的PWM通道n。</p> <p>0 = 禁止下降沿锁存中断</p> <p>1 = 使能下降沿锁存中断</p>
[7:6]	Reserved	保留.
[5:0]	CAPRIENn	<p>PWM上升沿捕捉锁存中断使能</p> <p>每位n控制相应的PWM通道n。</p> <p>0 = 禁止上升沿锁存中断</p> <p>1 = 使能上升沿锁存中断</p>



PWM捕捉中断标志寄存器(PWM_CAPIF)

寄存器	偏移量	R/W	描述	复位后值
PWM_CAPIF	PWMx_BA+0x254	R/W	PWM 捕捉中断标志寄存器	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		CAPFIF5	CAPFIF4	CAPFIF3	CAPFIF2	CAPFIF1	CAPFIF0
7	6	5	4	3	2	1	0
Reserved		CAPRIF5	CAPRIF4	CAPRIF3	CAPRIF2	CAPRIF1	CAPRIF0

位	描述	
[31:14]	Reserved	保留
[13:8]	CAPFIFn	<p>PWM下降沿捕捉锁存中断标志 该位写1清0。每位n控制相应的PWM通道n。 0 =没有下降沿捕捉锁存条件发生。 1 =下降沿捕捉锁存条件发生，该标志被置高。</p>
[7:6]	Reserved	保留.
[5:0]	CAPRIFn	<p>PWM上升沿捕捉锁存中断标志 该位写1清0。每位n控制相应的PWM通道n。 0 =没有上升沿捕捉锁存条件发生。 1 =上升沿捕捉锁存条件发生，该标志被置高。</p>

PWM周期寄存器缓存 0, 2, 4 (PWM_PBUF0, 2, 4)

寄存器	偏移量	R/W	描述	复位后值
PWM_PBUF0	PWMx_BA+0x304	R	PWM PERIOD0 缓存	0x0000_0000
PWM_PBUF2	PWMx_BA+0x30C	R	PWM PERIOD2 缓存	0x0000_0000
PWM_PBUF4	PWMx_BA+0x314	R	PWM PERIOD4 缓存	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	PBUF	PWM周期寄存器缓存（只读） 用作有效周期寄存器

PWM 比较寄存器缓存0~5 (PWM_CMPBUF0~5)

寄存器	偏移量	R/W	描述	复位后值
PWM_CMPBUF0	PWMx_BA+0x31C	R	PWM CMP0 缓存	0x0000_0000
PWM_CMPBUF1	PWMx_BA+0x320	R	PWM CMP1 缓存	0x0000_0000
PWM_CMPBUF2	PWMx_BA+0x324	R	PWM CMP2 缓存	0x0000_0000
PWM_CMPBUF3	PWMx_BA+0x328	R	PWM CMP3 缓存	0x0000_0000
PWM_CMPBUF4	PWMx_BA+0x32C	R	PWM CMP4 缓存	0x0000_0000
PWM_CMPBUF5	PWMx_BA+0x330	R	PWM CMP5 缓存	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	CMPBUF	PWM比较寄存器缓存（只读） 用作有效CMP寄存器



6.8 基本 PWM 发生器和捕获定时器 (BPWM)

6.8.1 概述

NUC131提供2组BPWM发生器----- BPWM0 和 BPWM1，如图 6-57。每组BPWM提供6个BPWM输出或输入捕获通道。有一个12位预分频器来支持灵活的时钟，用于带有16位比较器的16位BPWM计数器。BPWM 计数器支持递增计数，递减计数和可逆计数，6个通道共享一个计数器。BPWM 使用比较器与计数器比较来产生中断。这些中断用于产生BPWM脉冲，中断和触发ADC启动转换的信号。BPWM 输出控制单元，支持极性输出，独立的管脚屏蔽和三态输出使能。

BPWM产生器也支持输入捕获功能，当输入信道有一个上升转变，下降转变或上下两种转变时，可以锁存BPWM计数器值到对应寄存器。

6.8.2 特性

6.8.2.1 BPWM 功能特性

- 支持最大的频率达到100MHz
- 支持两组BPWM，每组提供6个通道
- 支持独立的模式用于BPWM输出或捕获输入通道
- 支持从1到4096的12位预分频器
- 支持16位分辨率计数器，每个模块提供1个BPWM计数器
 - 递增，递减和可逆计数器操作模式
- 每个BPWM管脚支持屏蔽功能和三态使能
- 支持对于如下事件的中断：
 - BPWM计数器计数到0, 周期值或比较值
- 支持触发ADC基于如下事件：
 - BPWM计数器计数0, 周期值或比较值

6.8.2.2 捕获功能特性

- 支持最多12个捕获输入通道，16位分辨率
- 支持上升或下降捕获条件
- 支持输入上升/下降捕获中断
- 支持带有计数器重载选项的上升/下降捕获



6.8.2.3 PWM & BPWM 特性比较表

特性	PWM	BPWM
计数器数目	2 个通道共享1个定时器, 一共 6 个 定时器	6 个通道共享1个定时器, 一共 1 个定 时器
互补模式	√	×
停滞时间功能	√	×
刹车功能	√	×
捕获加载	2 个通道加载一个定时器	6 个通道加载一个定时器

表 6-14 PWM & BPWM 特性比较表

6.8.3 框图

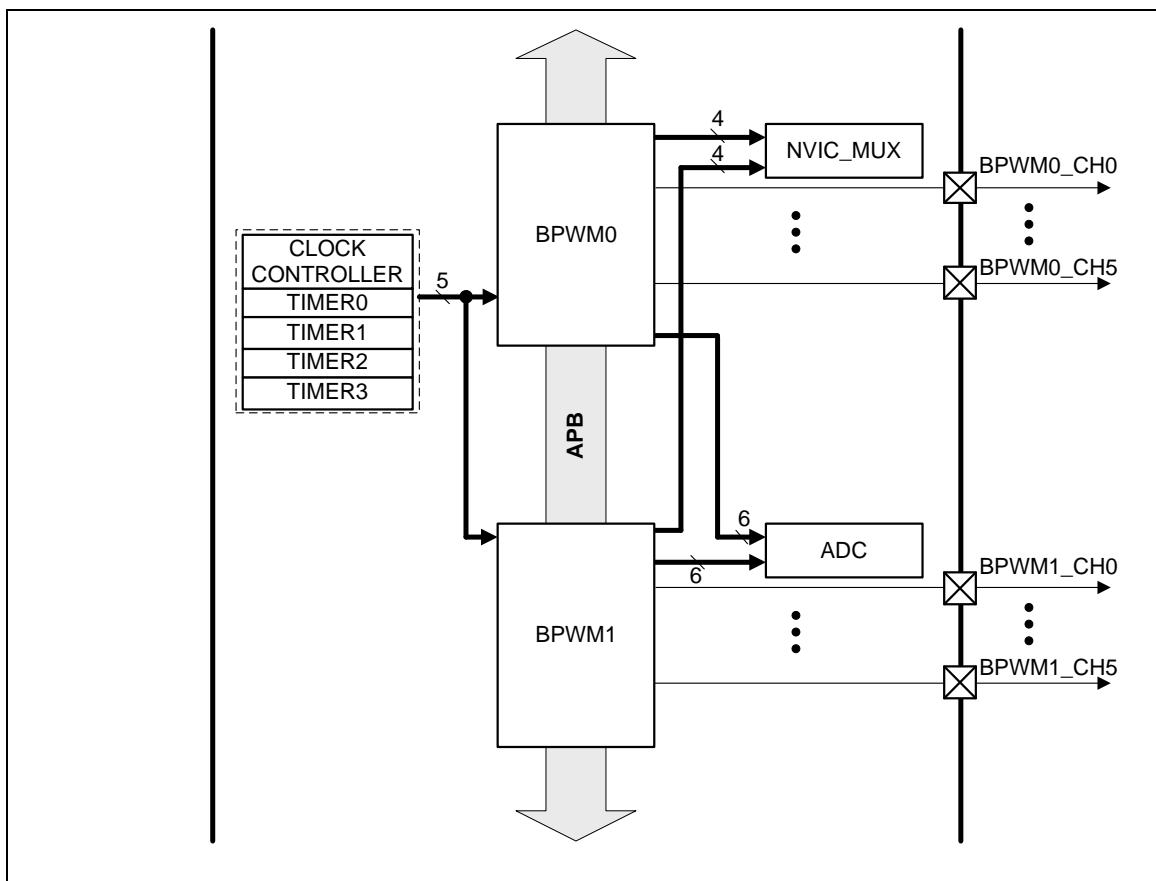


图 6-57 BPWM 产生器框图

PWM系统时钟频率可以设置为等于或双倍于HCLK,如图 6-58,详细的寄存器设定,请参考表 6-15.

每个PWM产生器只有一个时钟源输入,可以从系统时钟或4个定时器触发PWM输出中选择,如图 6-59,通过(BPWM_CLKSRC[2:0])设定BPWM_CLK0。

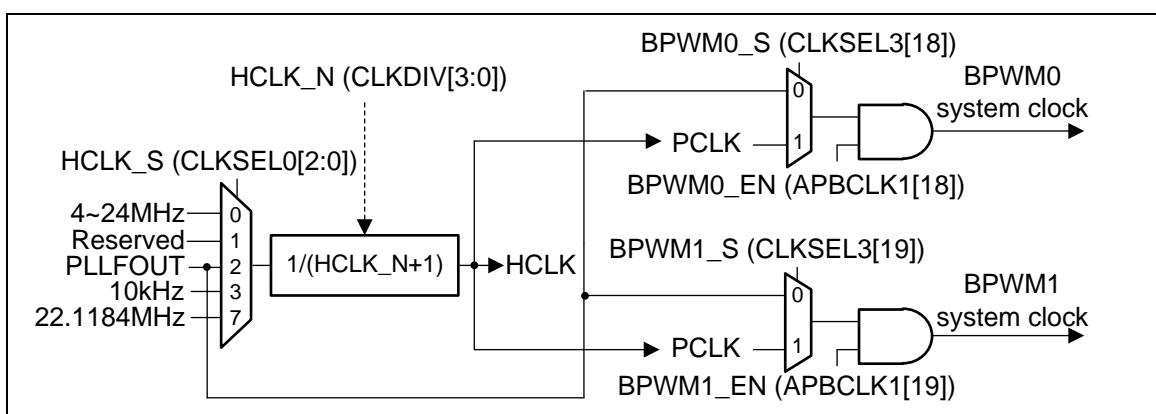


图 6-58 BPWM 系统时钟源控制

BPWM 系统时钟/HCLK 分频比例	HCLK_S (CLKSEL0[2:0])	HCLK_N (CLKDIV[3:0])	BPWMn_S (CLKSEL3[X]), (N, X)代表 (0, 18) 或 (1, 19)
1/1	无关的	无关的	1
2/1	2	1	0

表 6-15 BPWM 系统时钟源控制寄存器设定表

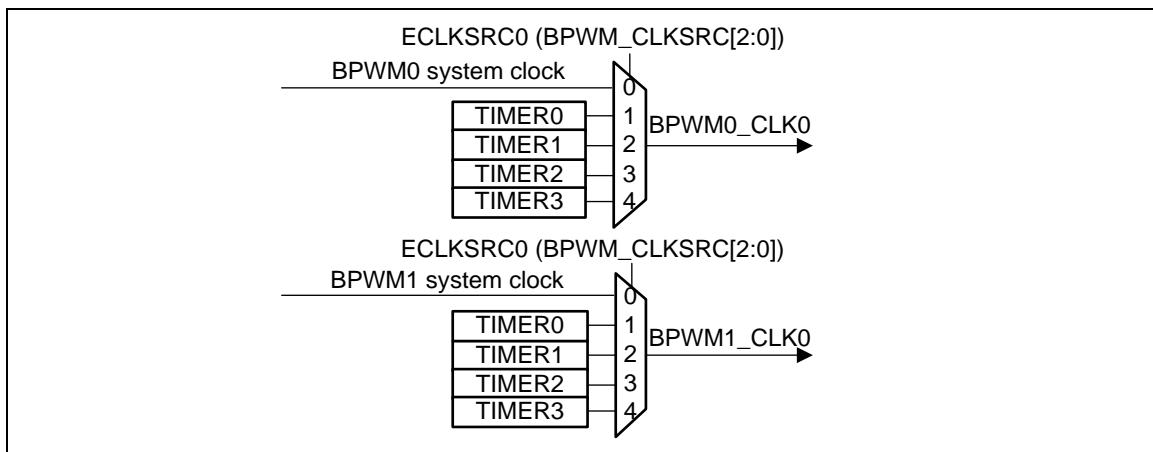


图 6-59 BPWM 时钟源控制

图 6-60为BPWM独立模式架构。所有的6个通道共享同一个计数器。当计数器计数到0, PERIOD (BPWM_PERIODn[15:0])或等于比较器数值时, 事件将产生。这些事件传递到对应的产生器后, 将产生BPWM脉冲, 中断信号和触发ADC启动转换信号。输出控制器用于改变BPWM脉冲输出状态。

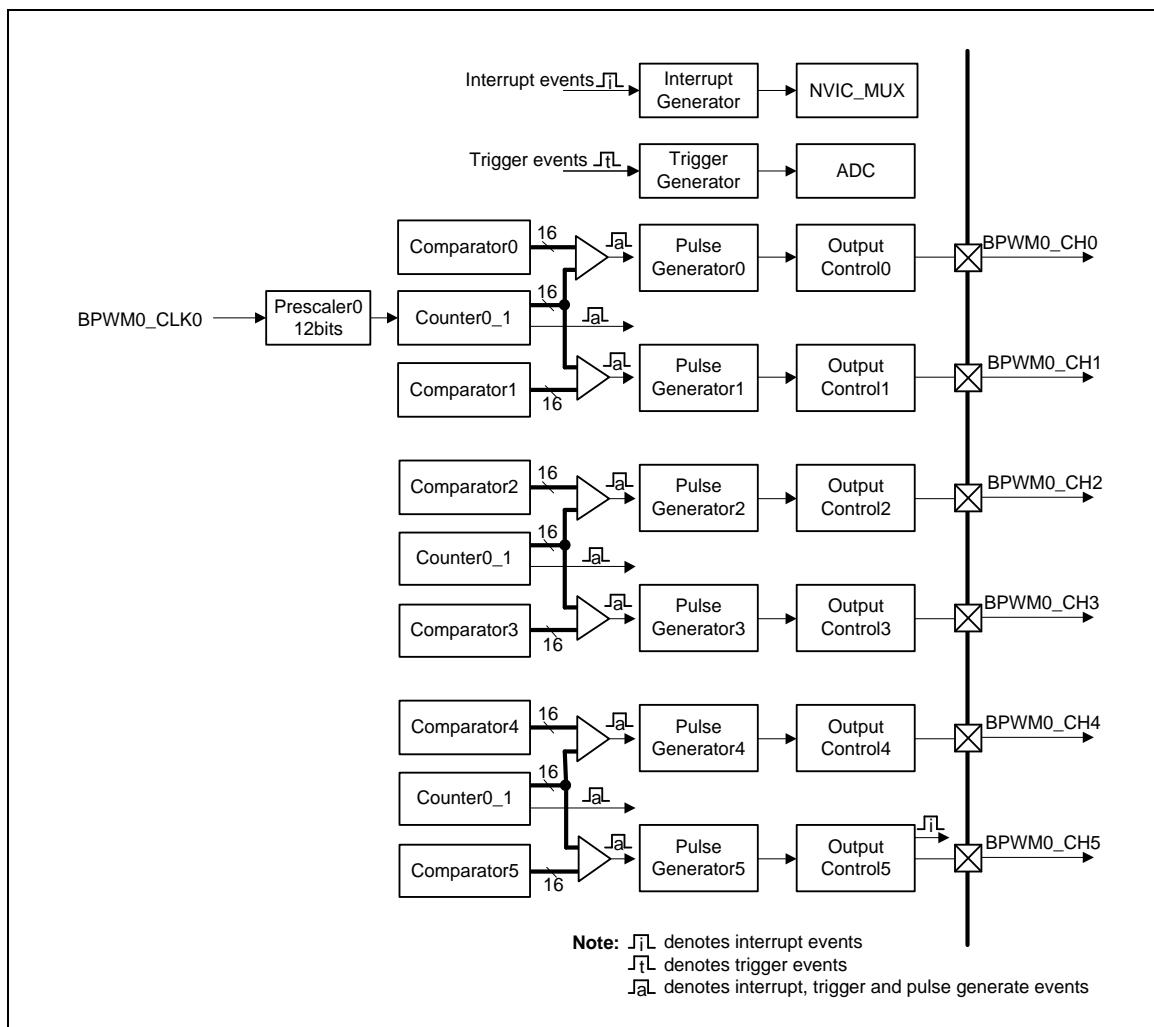


图 6-60 BPWM 独立模式框图

6.8.4 基本配置

BPWM管脚功能配置的寄存器有：GPB_MFP, GPC_MFP和GPD_MFP。BPWM时钟使能寄存器是APBCLK1[19:18]。BPWM时钟源选择寄存器是CLKSEL3[19:18]。

6.8.5 功能描述

6.8.5.1 BPWM预分频器

BPWM预分频器用于时钟除频，预分频器计数CLKPSC +1次数，BPWM计数器只计一次。CLKPSC(时钟预分频寄存器)的设置寄存器是CLKPSC (BPWM_CLKPSCn[11:0], n 代表 0, 2, 4). 图 6-61中描述BPWM通道0 CLKPSC的波形。

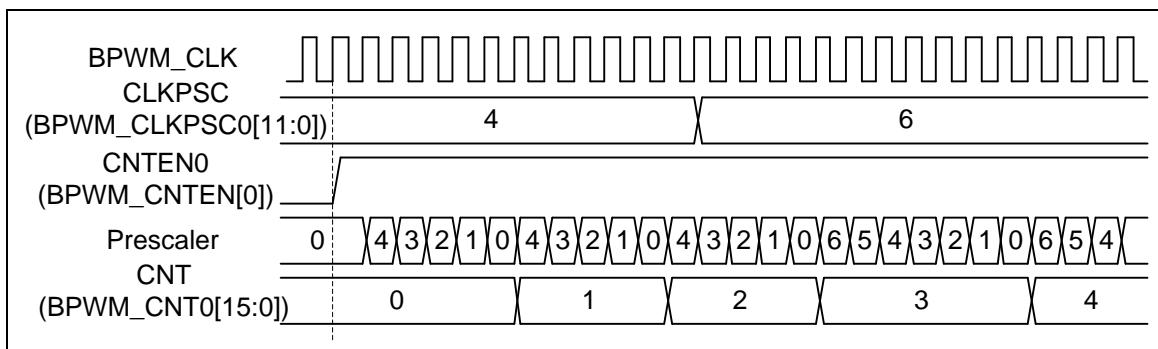


图 6-61 BPWM_CH0 CLKPSC 波形图

6.8.5.2 BPWM计数器

BPWM 支持3种计数器操作类型：递增计数器，递减计数器和可逆计数器。

6.8.5.3 递增计数器类型

在递增计数器操作中，16位BPWM计数器是一个递增的计数器，从0到PERIOD (BPWM_PERIODn[15:0], n代表通道数)来完成一个BPWM周期。当前计数器数值可以通过寄存器CNT (BPWM_CNTn[15:0])读出。当计数器计数到0时BPWM产生0点事件，并且当计数到PERIOD还会产生周期点事件。下图描述一个递增计数器的例子，其中BPWM周期时间= (PERIOD+1) * BPWM时钟时间。

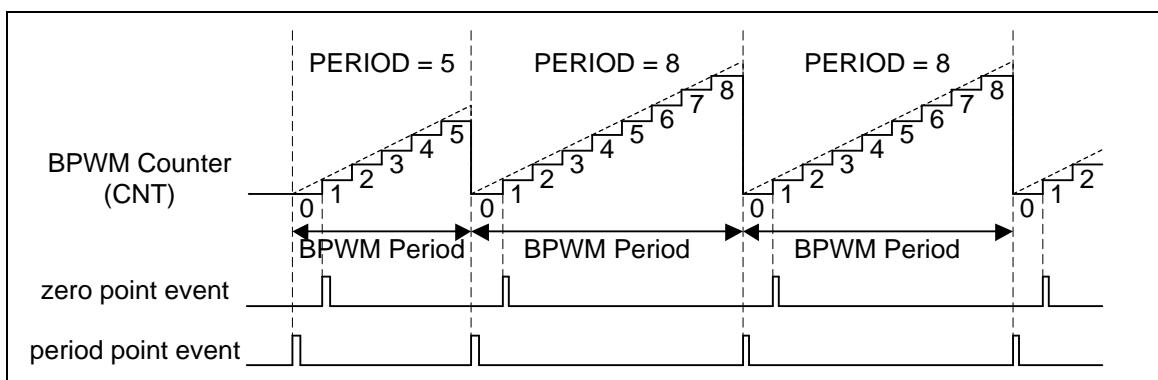


图 6-62 BPWM 递增计数器类型

6.8.5.4 递减计数器类型

在递减计数器操作中，16位BPWM计数器是一个递减的计数器，并且从PERIOD开始递减计数，到0时完成一个BPWM周期。当前的计数器数值可以通过CNT读出。当计数器计数到0，BPWM产生0点事件，当计数到PERIOD时，产生周期点事件。下图描述了一个递减计数器的例子，其中BPWM 周期时间 = (PERIOD+1) * BPWM 时钟时间。

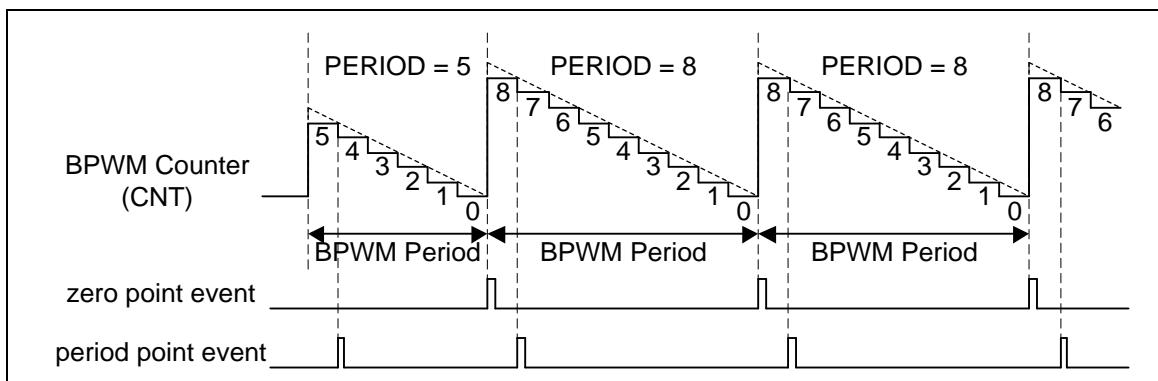


图 6-63 BPWM 递减计数器类型

6.8.5.5 可逆计数器类型

在可逆计数器操作中，16位BPWM是一个可逆的计数器，计数器从0开始到PERIOD，然后再回数到0来完成一个BPWM的周期。当前的计数器数值可以通过CNT读出。当计数器计数到0，BPWM产生0点事件，当计数器计数到PERIOD时，则会产生中间点事件。下图描述了一个可逆计数器的例子，其中BPWM 周期时间 = $(2 \times \text{PERIOD}) * \text{BPWM 时钟时间}$ 。DIRF (BPWM_CNTn[16])是计数器方向指示标志，高是递增计数，低是递减计数。

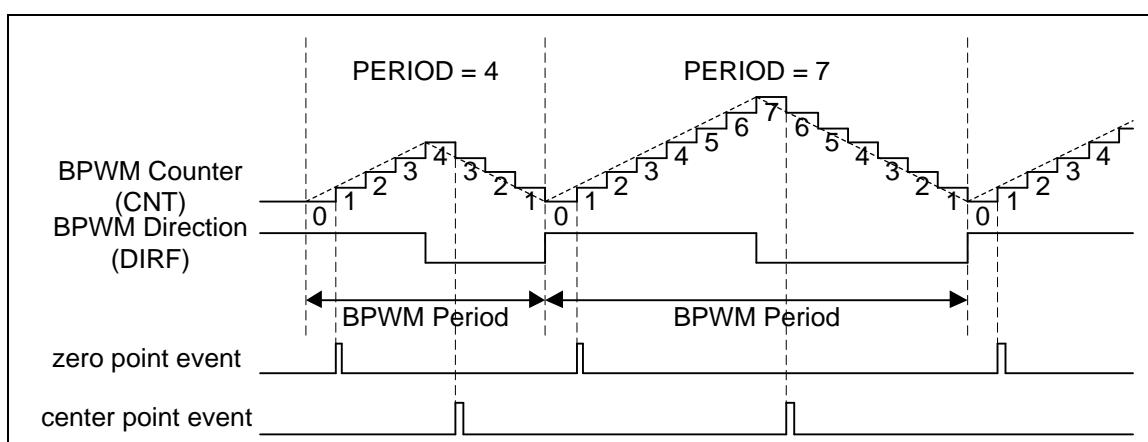


图 6-64 BPWM 可逆计数器类型

6.8.5.6 BPWM 比较器

CMPDAT (BPWM_CMPDATn[15:0]) 为 BPWM 第 n 个通道的比较器寄存器。每个通道只有一个 CMPDAT。CMPDAT 的数值是与计数器数值持续比较的。当计数器的数值与比较器的数值相等的时候，BPWM 产生一个中断事件，并且通过这个事件来产生 BPWM 脉冲、中断或触发 ADC 启动转换。在可逆计数器类型，在 BPWM 周期内将产生两个事件，如图 6-65 所示。

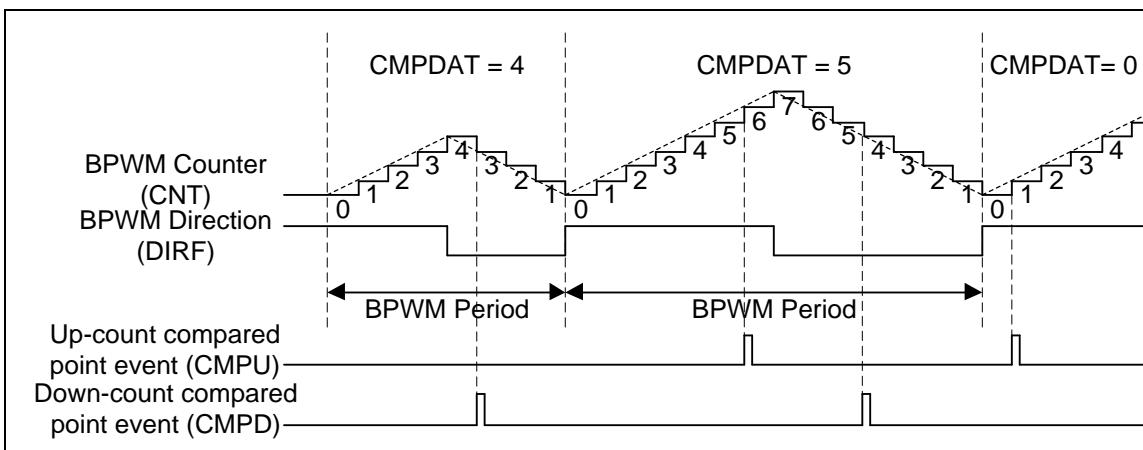


图 6-65 BPWM CMPDAT 中断在可逆计数器类型

6.8.5.7 BPWM 双重缓冲

双重缓冲是使用两个缓冲区来分离软件写和硬件操作的时序。当寄存器被软件改动以后，硬件会依据加载模式，按照不同时序，加载寄存器数值到缓冲区寄存器。硬件的动作是基于缓冲区的数值。这样可以避免因为软件和硬件操作不同步而造成问题。

对于PERIOD 和 CMPDAT，BPWM有双重缓冲功能。双重缓冲功能用于各种加载模式中，接下来将会详细描述。举例，如图 6-66，在周期加载模式中，软件改写PERIOD 和 CMPDAT后，BPWM将在下一个周期开始时加载新的数值到缓冲区 PBUF (BPWM_PBUFn[15:0]) 和 CMPBUF (BPWM_CMPBUFn[15:0])中，无需影响当前周期计数器的操作。加载数值到缓冲区的模式共有3种：周期加载模式，立刻加载模式和中间加载模式。

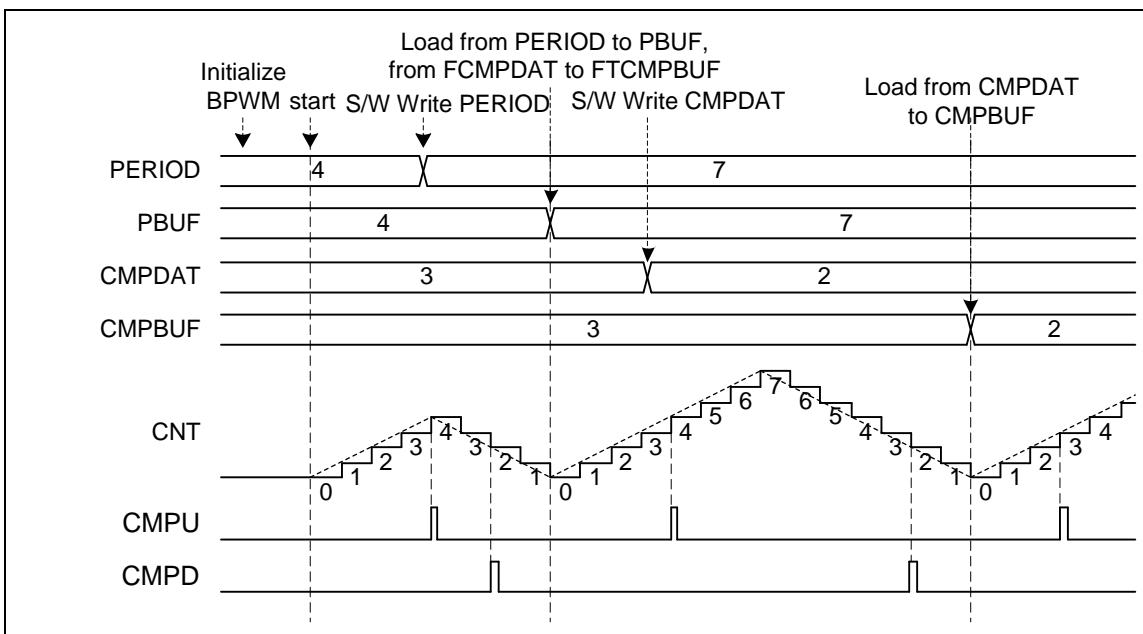


图 6-66 BPWM 双重缓冲区说明

6.8.5.8 周期加载模式

周期加载模式是默认的加载模式，在所有加载模式中，优先级最低。当周期结束时，例如：在递增操作模式中当BPWM计数器从0到PERIOD、或在递减操作模式中从PERIOD到0、或在可逆计数器操作模式中从0到PERIOD再归0，才会将PERIOD 和 CMPDAT 都加载到它们的缓冲区中。

图 6-67为递增操作中周期加载模式的时序，PERIOD DATA0为PERIOD原始数值，PERIOD DATA1为第一次通过软件更新的PERIOD数值，以此类推。CMPDAT也按照这个原则。下面为描述图 6-67的步骤顺序。通过观察PWM周期和CMPU事件，用户可以知道PERIOD 和CMPDAT的更新条件。

1. 在点 1，软件写 CMPDAT DATA1 到 CMPDAT.
2. 在点 2，在 PWM 周期结束时，硬件加载 CMPDAT DATA1 到 CMPBUF
3. 在点 3，软件写 PERIOD DATA1 到 PERIOD.
4. 在点 4，在周期结束时，硬件加载 PERIOD DATA1 到 PBUF.
5. 在点 5，软件写 PERIOD DATA2 到 PERIOD.
6. 在点 6，在周期结束时，硬件加载 PERIOD DATA2 到 PBUF.

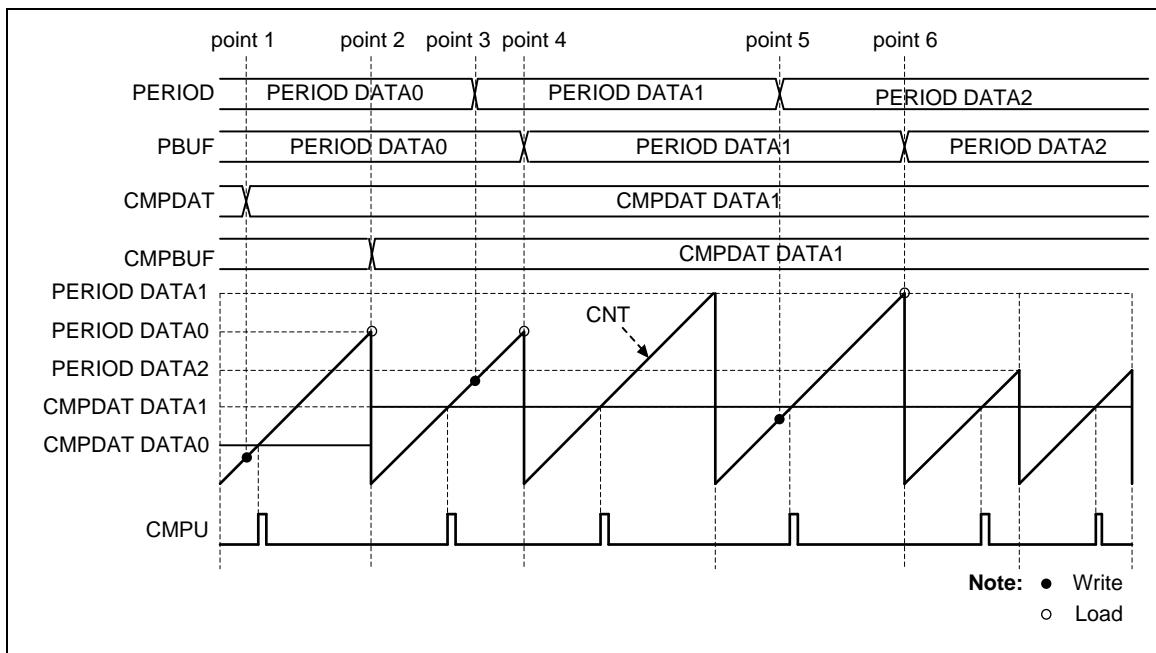


图 6-67 在递增计数器中使用周期加载模式

6.8.5.9 立即加载模式

如果BPWM通道n对应的IMMLDENn (BPWM_CTL0[21:16])位=1，当软件更新PERIOD 和 CMPDAT 后，软件会将PERIOD 和 CMPDAT中数值立刻加载到缓冲器中。如果更新的PERIOD数值比当前计数器数值小，计数器将循环计数。立即加载模式的优先级最高。如果IMMLDENn已经设置， n信道的其他加载模式将无效。图 6-68 描述一个例子，步骤如下。

1. 在点 1，软件写 CMPDAT DATA1，硬件立即加载 CMPDAT DATA1 到 CMPBUF.
2. 在点 2，软件写 PERIOD DATA1，PERIOD DATA1 比当前的计数器数值大，计数器将继续计数直到等于 PERIOD DATA1 才完成周期加载。

3. 在点 3，软件写 PERIOD DATA2，PERIOD DATA2 小于当前计数器数值，计数器将继续计数到最大的数值 0xFFFF，然后从 0 循环计数到 PERIOD DATA2 来完成这个周期计数。

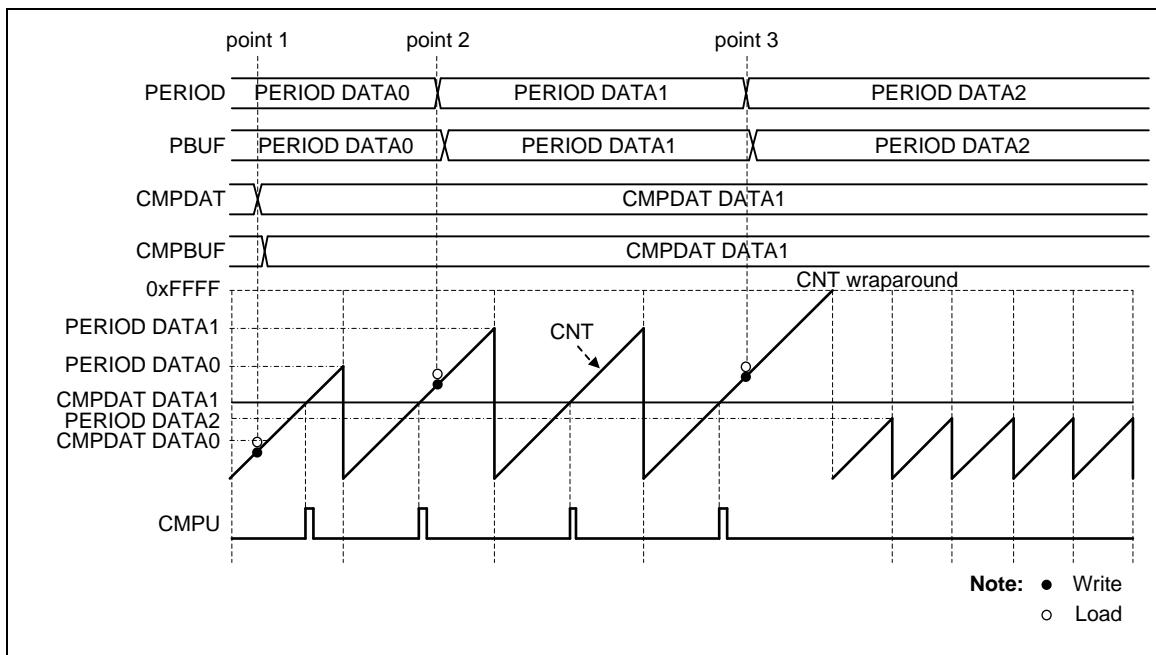


图 6-68 在递增计数器中使用立刻加载模式

6.8.5.10 中间加载模式

如果BPWM通道n对应的CTRLDn (BPWM_CTL0[5:0])位=1，而且计数类型为可逆计数器类型，CMPDAT将会在周期的中间（即：计数器计数到PERIOD时）加载CMPBUFn。PERIOD加载时序与周期加载模式相同。图 6-69将举例说明，具体步骤如下。

1. 在点 1，软件写 CMPDAT DATA1.
2. 在点 2，在 PWM 周期的中间点，硬件加载 CMPDAT DATA1 到 CMPBUF.
3. 在点 3，软件写 PERIOD DATA1.
4. 在点 4，在 PWM 结束点，硬件加载 PERIOD DATA1 到 PBUF.
5. 在点 5，软件写 CMPDAT DATA2.
6. 在点 6，在 PWM 周期的中间点，硬件加载 CMPDAT DATA2 到 CMPBUF.
7. 在点 7，软件写 PERIOD DATA2.
8. 在点 8，在 PWM 周期结束点，硬件加载 PERIOD DATA2 到 PBUF.

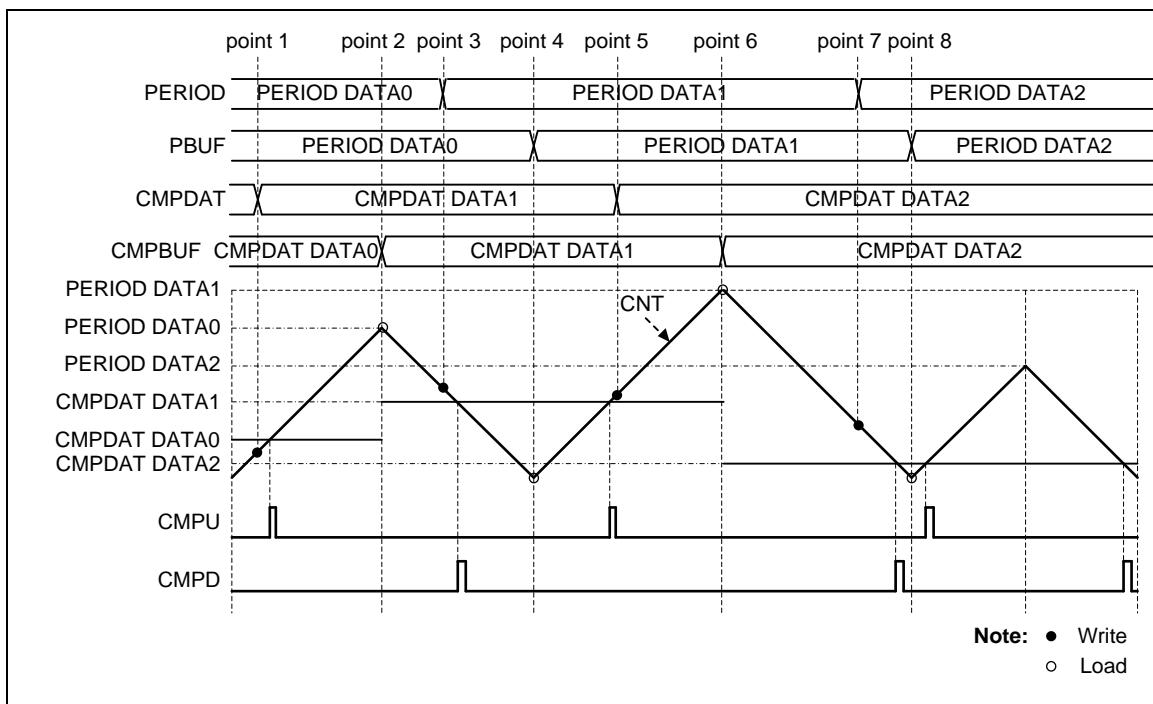


图 6-69 可逆计数器类型的中间加载模式

6.8.5.11 BPWM 脉冲产生器

BPWM 使用计数器和比较器中断事件来产生BPWM脉冲。这些中断事件是：零点，在递增计数器类型和递减计数器类型中的周期点，在可逆计数器类型中的中间点，以及三种计数器类型中的计数器等于比较器的计数点。在可逆计数器类型中，有两个计数器等于比较器的计数点，一个是递增计数时的计数点，另一个是递减计数时的计数点。

每个中断事件点都可以决定BPWM的波形，通过设定BPWM_WGCTL0 和 BPWM_WGCTL1寄存器，BPWM波形输出可以有以下选择：保持不变(X)，输出低(L)，输出高(H) 或者翻转输出(T)。使用这些点可以轻松的产生不对称的BPWM脉冲或如图 6-70所示的各种各样的波形图。如图，比较器n用于产生BPWM脉冲。n 表示通道数：0到5. CMPU 代表在递增的期间，CNT与CMPDAT相等。CMPD代表在递减的期间，CNT与CMPDAT相等。

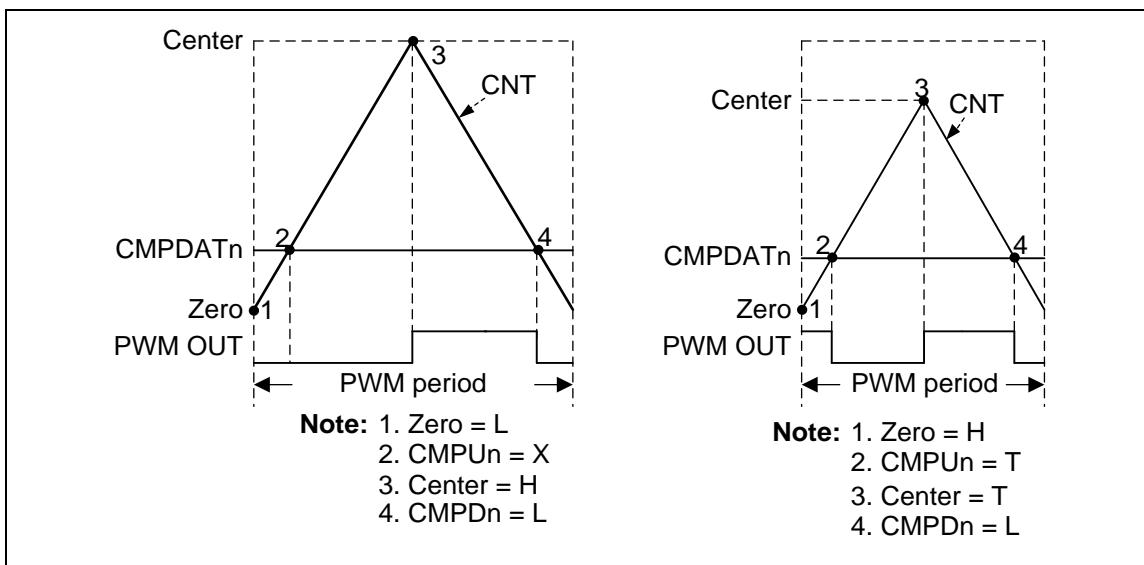


图 6-70 BPWM 脉冲产生

多个事件中断产生时，可能有时设置值为相同数值，因此，事件在不同计数器类型中产生时的优先级如下所示，递增计数器类型(表 6-16)，递减计数器类型(表 6-17)，可逆计数器类型(表 6-18)。通过使用中断的优先级，使用者可以轻松产生0% 到100%的占空比的波形，如图 6-71。

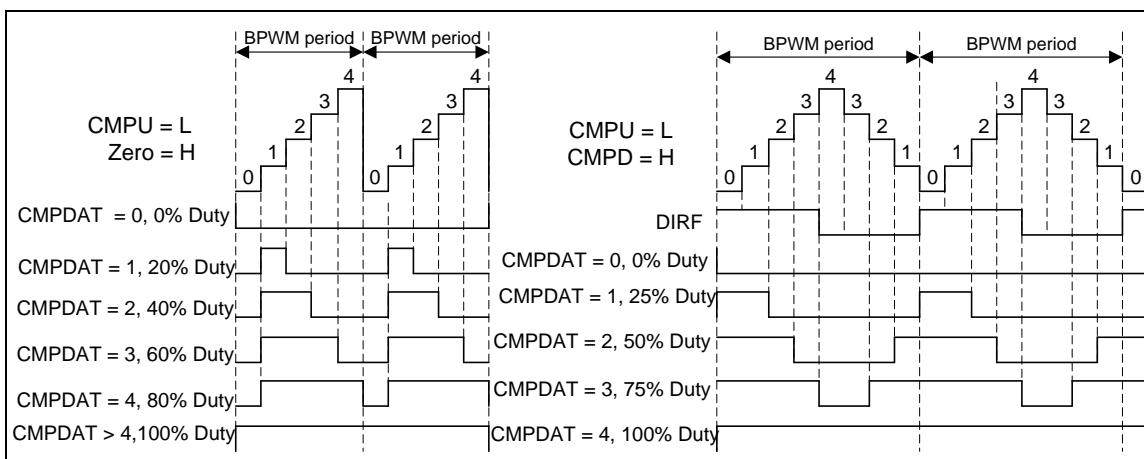


图 6-71 BPWM 0% 到 100% 脉冲产生

优先级	递增中断
1 (最高)	CNT = 周期 值(PERIOD)
2	CNT = CMPUm
3	CNT = CMPUn
4 (最低)	CNT = 零

表 6-16 递增计数器中 BPWM 脉冲中断优先级

优先级	递减中断
1 (最高)	CNT = 零
2	CNT = CMPDm
3	CNT = CMPDn
4 (最低)	CNT = 周期值(PERIOD)

表 6-17 递减计数器重 BPWM 脉冲中断优先级

优先级	递增中断	递减中断
1 (最高)	CNT = CMPUm	CNT = CMPDm
2	CNT= CMPUn	CNT = CMPDn
3	CNT = 零	CNT = 中间(PERIOD)
4	CNT = CMPDm	CNT = CMPUm
5 (最低)	PERIOD = CMPDn	CNT = CMPUn

表 6-18 可逆计数器中 BPWM 脉冲中断优先级

6.8.5.12 BPWM 输出控制

BPWM 脉冲产生后，有三个步骤可以控制BPWM通道的输出，分别为：屏蔽，管脚极性和输出使能。如图 6-72。

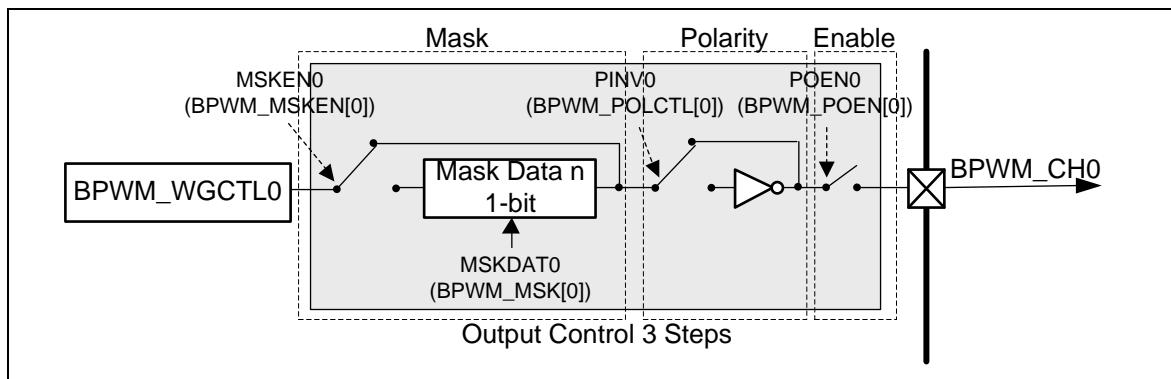


图 6-72 BPWM_CH0 输出控制的三个步骤

6.8.5.13 BPWM 屏蔽输出功能

每个BPWM输出信道都可以手动配置。通过设置对应在BPWM 屏蔽使能控制寄存器(BPWM_MSKEN)和BPWM屏蔽数据寄存器(BPWM_MSK)中的设定位，在每个独立比较单元区间都可驱动BPWM通道输出指定逻辑电平。当控制不同类型的电子换向电机(ECM)，比如BLDC马达，BPWM 屏蔽位将会非常有

用。BPWM_MSKEN寄存器包含六个比特位，MSKENn(BPWM_MSKEN[5:0])决定哪些BPWM通道输出将被屏蔽（覆盖）。MSKENn(BPWM_MSKEN[5:0])为高有效。BPWM_MSK寄存器也包含六个比特位。当通道被屏蔽后，MSKDATn(BPWM_MSK[5:0])通过MSKDAT决定BPWM通道输出波形。图 6-73举例说明如何用BPWM 屏蔽控制实现通道屏蔽（覆盖）的功能。

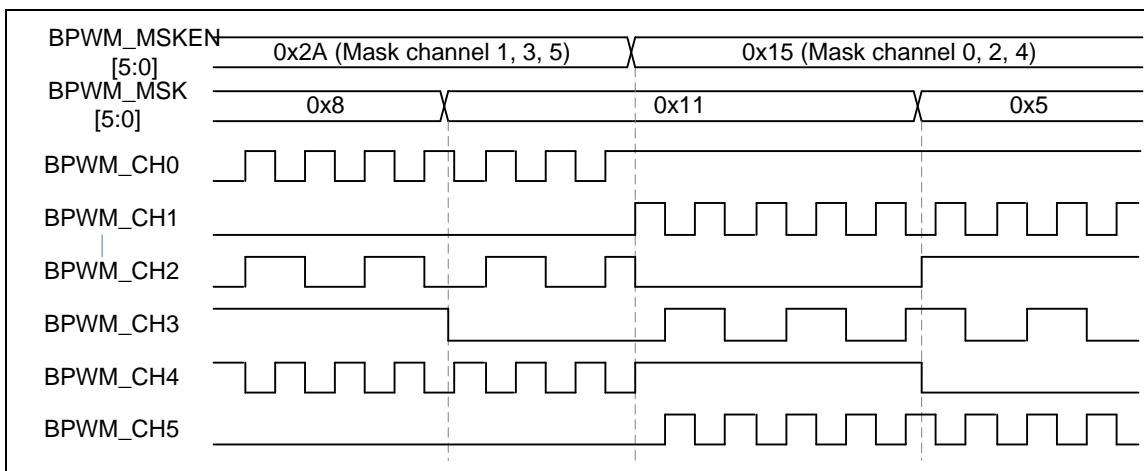


图 6-73 屏蔽控制波形图

6.8.5.14 极性控制

从BPWM_CH0 到BPWM_CH5，每个BPWM口都有一个独立的极性控制模组来配置BPWM输出的有效状态。BPWM默认输出高有效。这意味着BPWM关状态是低，开状态是高。但对于每个独立的BPWM通道而言，通过设定BPWM负极性控制寄存器(BPWM_POLCTL)，这种设定是可更改的。图 6-74描述了BPWM在不同极性设定时启动的初始状态。

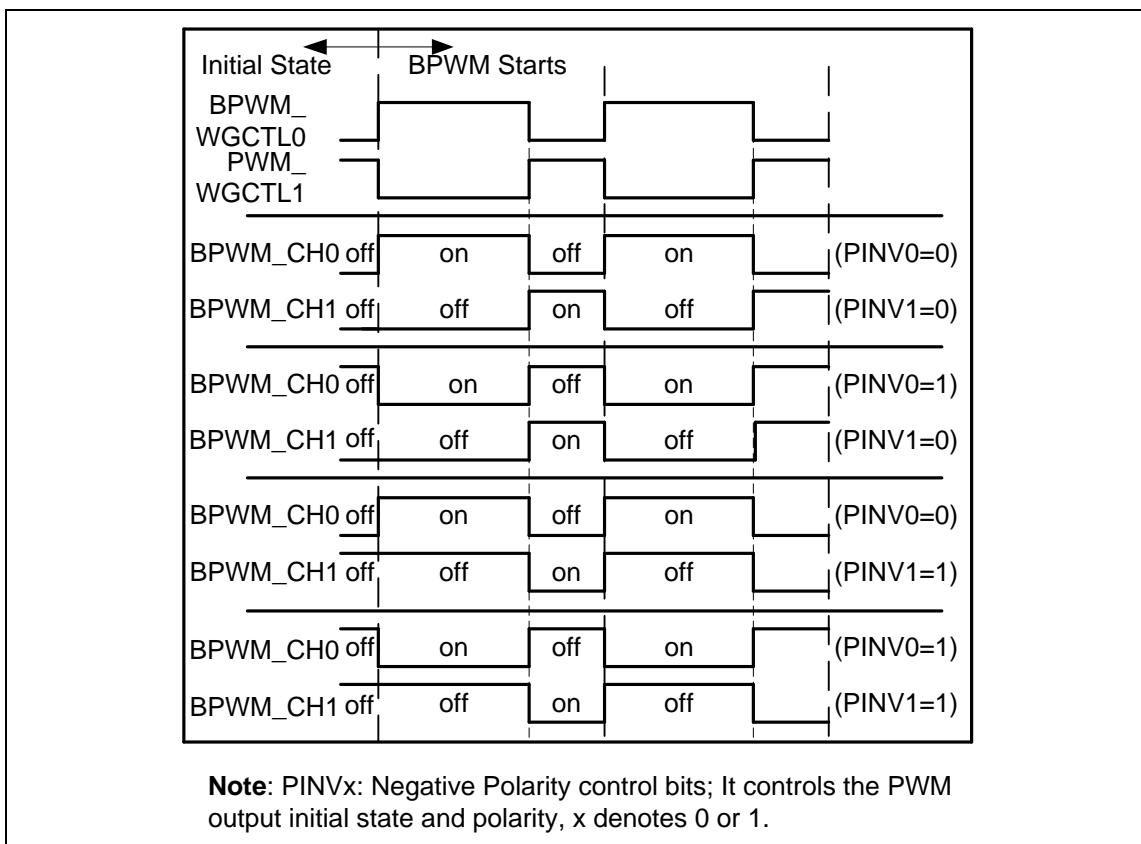


图 6-74 初始状态和极性控制

6.8.5.15 BPWM 中断产生器

对于每个BPWM, 有两个独立的中断, 如图 6-75所示。

BPWM 中断 (BPWM_INT) 来源于 BPWM 互补中断。计数器可以产生 0 点中断标志 ZIFn (BPWM_INTSTS0[5:0]) 和周期点中断标志 PIFn (BPWM_INTSTS0[13:8])。当BPWM通道n计数器等于寄存器BPWM_CMPDATn中数值, 根据计数方向, 不同的中断标志将被触发。如果这个匹配发生在递增的方向, 递增的中断标志 CMPUIFn (BPWM_INTSTS0[21:16]) 将被挂起, 如果这个匹配是发生在反方向, 递减的中断标志 CMPDIFn (BPWM_INTSTS0[29:24]) 将被挂起。如果相应的中断使能位为使能状态, 将触发中断产生中断信号。

另一个中断是捕捉中断(CAP_INT)。它共享NVIC的BPWM_INT向量, 当CAPRIFn (BPWM_CAPIF[5:0]) 被触发并且捕获上升中断使能位CAPRIENn (BPWM_CAPIEN[5:0])=1, CAP_INT 将产生。或在下降沿条件, 当下降沿中断使能位CAPFIENn (BPWM_CAPIEN[13:8]) =1, CAPFIFn (BPWM_CAPIF[13:8]) 会被触发。

下图说明了BPWM的中断架构。

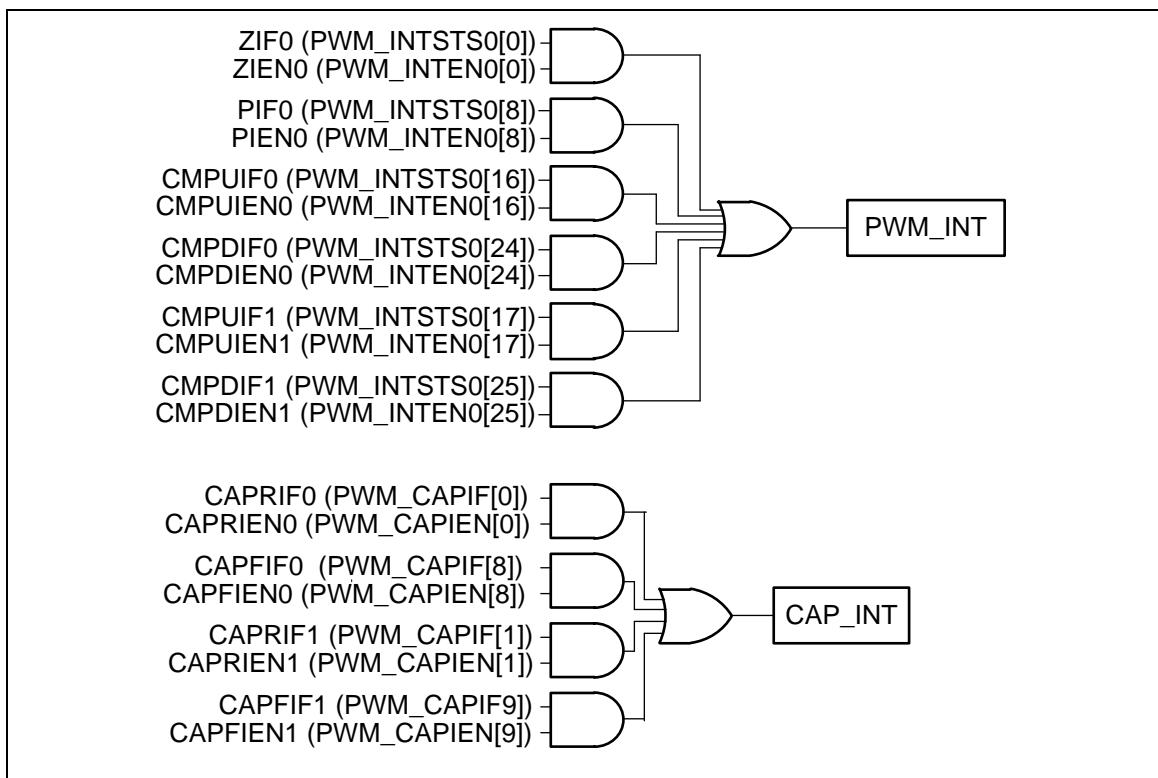


图 6-75 BPWM_CH0 和 BPWM_CH1 一对中断架构图

6.8.5.16 BPWM 触发ADC产生器

BPWM 可以触发ADC转换器启动。每对BPWM通道共享相同的触发源。设定TRGSEL n 可以 选择触发源，其中TRGSEL n 是指TRGSEL0, TRGSEL1, ..., 以及TRGSEL5，各自对应的寄存器位置分别为BPWM_ADCTS0[3:0], BPWM_ADCTS0[11:8], BPWM_ADCTS0[19:16], BPWM_ADCTS0[27:24], BPWM_ADCTS1[3:0] 和 BPWM_EADTS1[11:8]。设定TRGEN n 是使能触发ADC输出，其中TRGEN n 是TRGEN0, TRGEN1, ..., TRGEN5，各自对应的寄存器位置在BPWM_ADCTS0[7], BPWM_ADCTS0[15], BPWM_ADCTS0[23], BPWM_ADCTS0[31], BPWM_ADCTS1[7] 和 BPWM_ADCTS1[15]。n (n = 0,1,..,5) 代表BPWM通道数。

每对通道有7个BPWM中断可以作为中断源来选择。图 6-76是一个BPWM_CH0 和 BPWM_CH1的例子。通过设置PERIOD 和CMPDAT寄存器，BPWM可以触发ADC在不同的时间点启动转换。图 6-77为可逆计数器类型中触发ADC的时序波形图。

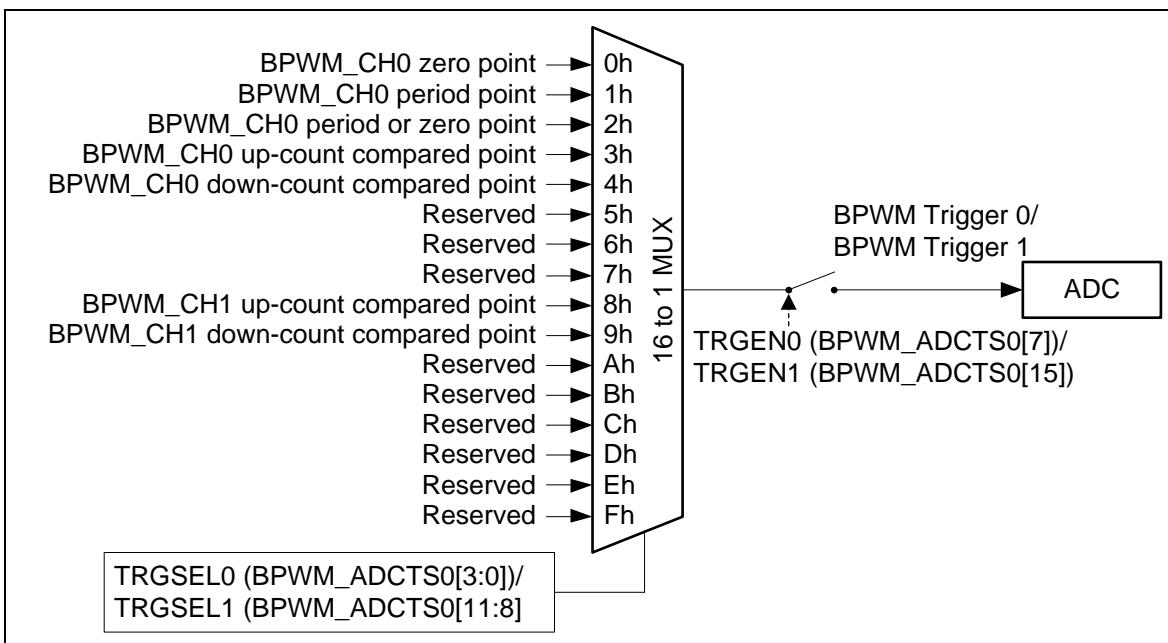


图 6-76 BPWM_CH0 和 BPWM_CH1一对触发ADC框图

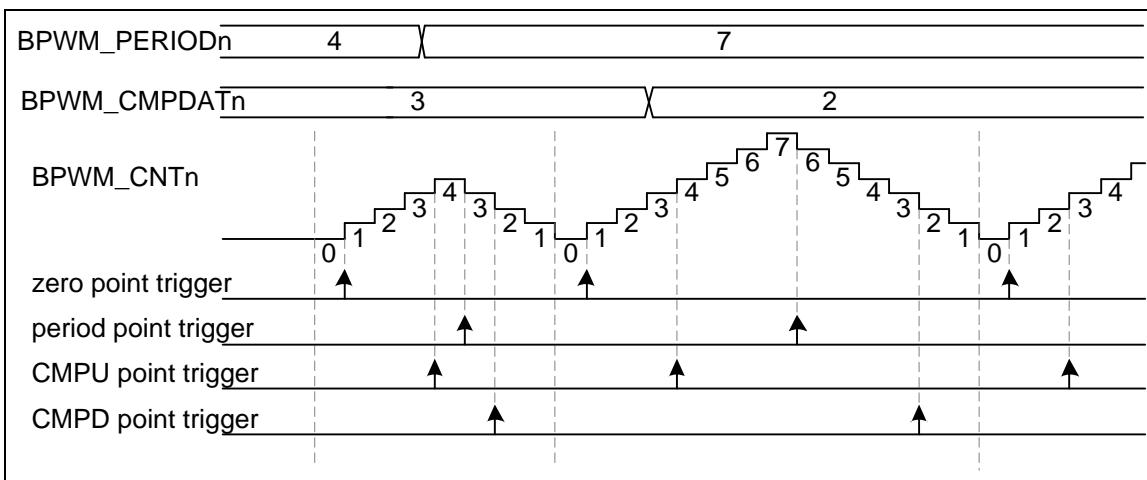


图 6-77 在可逆计数器类型中 BPWM 触发 ADC 的波形图

6.8.5.17 捕获操作

捕获输入信道和BPWM输出信道共享同样的管脚和计数器。计数器可以工作在递增或递减类型模式。如果输入通道有一个上升转变或下降转变，捕获功能将把BPWM计数值分别锁存到寄存器RCAPDATn (BPWM_RCAPDATn[15:0])或寄存器FCAPDATn (BPWM_FCAPDATn[15:0])。

如果发生了上升或下降锁存，并且相应的通道n的上升或下降中断使能位使能，捕获功能将会产生一个中断CAP_INT(使用BPWM_INT向量)，其中CAPRIENn (BPWM_CAPIEN[5:0])是对于上升沿，CAPFIENn (BPWM_CAPIEN[13:8])是对于下降沿。根据RCRLDENn和FCRLDENn设置(其中RCRLDENn和FCRLDENn分别位于寄存器BPWM_CAPCTL[21:16]和BPWM_CAPCTL[29:24])，当上升或下降锁存发生，相应的BPWM计数器可以重新加载为BPWM_PERIODn，注意：相应的GPIO管脚必须通过使能CAPINENn (BPWM_CAPINEN[5:0])，配置为捕获功能。图 6-78是信道0的捕获信道框图。

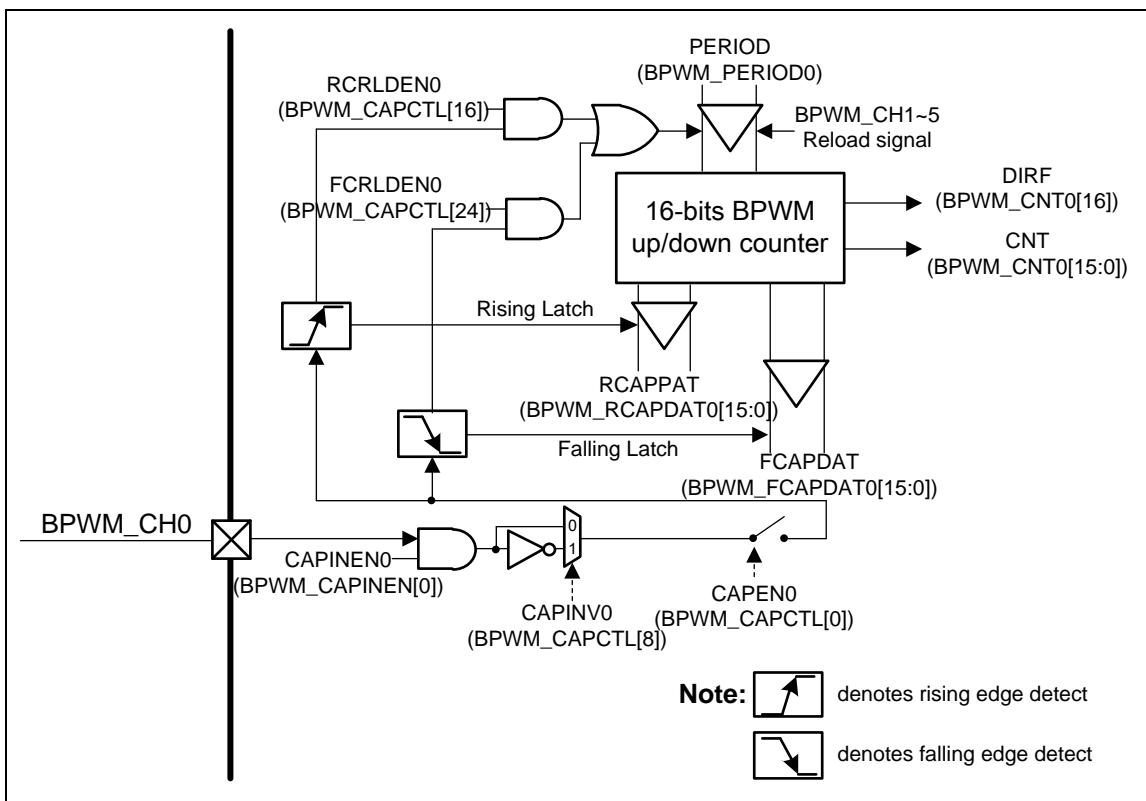


图 6-78 BPWM_CH0 捕获框图

图 6-79 为捕获功能时序，在这个例子中，捕获计数器被设置成BPWM递减计数器类型，而且周期 PERIOD 被设置成8。所以这个计数器计数的方向是从上到下的，从8到0。当输入捕获管脚检测到一个下降沿时，捕获功能把计数器数值锁存到BPWM_FCAPDATn中。当输入捕获管脚检测到一个上升沿，它锁存计数器数值到BPWM_RCAPDATn中。在这个时序框图中，当一开始检测到下降沿时，捕捉器重新加载计数器的数值，数值为周期值（PERIOD），原因是使能了FCRLDENn。但是在第二次下降沿时，计数器不会重新加载，这是因为关闭了FCRLDENn。在这个例子中，计数器在捕获到上升沿时也被重新加载了，原因是RCRLDENn也被使能了。

再者，如果这个例子是设定为递增计数器类型，计数器将重载数值0并且递增到数值PERIOD。需要强调的是计数器是所有通道共享的，所以计数器重新加载的次数也由所有通道的加载信号控制。图 6-79 为中断和中断标志产生的时序例子。当第n通道检测到上升沿，相应的位CAPRIFn (BPWM_CAPIF[5:0]) 会被硬件置起。类似的，通道n检测到下降沿，相应的位CAPFIFn(BPWM_CAPIF[13:8]) 会被硬件置起。CAPRIFn 和 CAPFIFn 可以通过软件写1清除。如果CAPRIFn被置起并且CAPRIENn也被置起，捕获功能会产生一个中断。如果CAPFIFn 被置起并且CAPFIENn也被置起，也会发生中断。

在本图没有描述的一个情况是：当CAPRI已经设定了，如果上升锁存再次发生了，运行状态寄存器 CRIFOVn (BPWM_CAPSTS[5:0]) 将通过硬件被置起，来指示CAPRI超载。同理，当下降锁存再次发生，对于中断标志CAPFI和超载状态CFIFOVn (BPWM_CAPSTS[13:8])，相同的硬件操作也会发生。

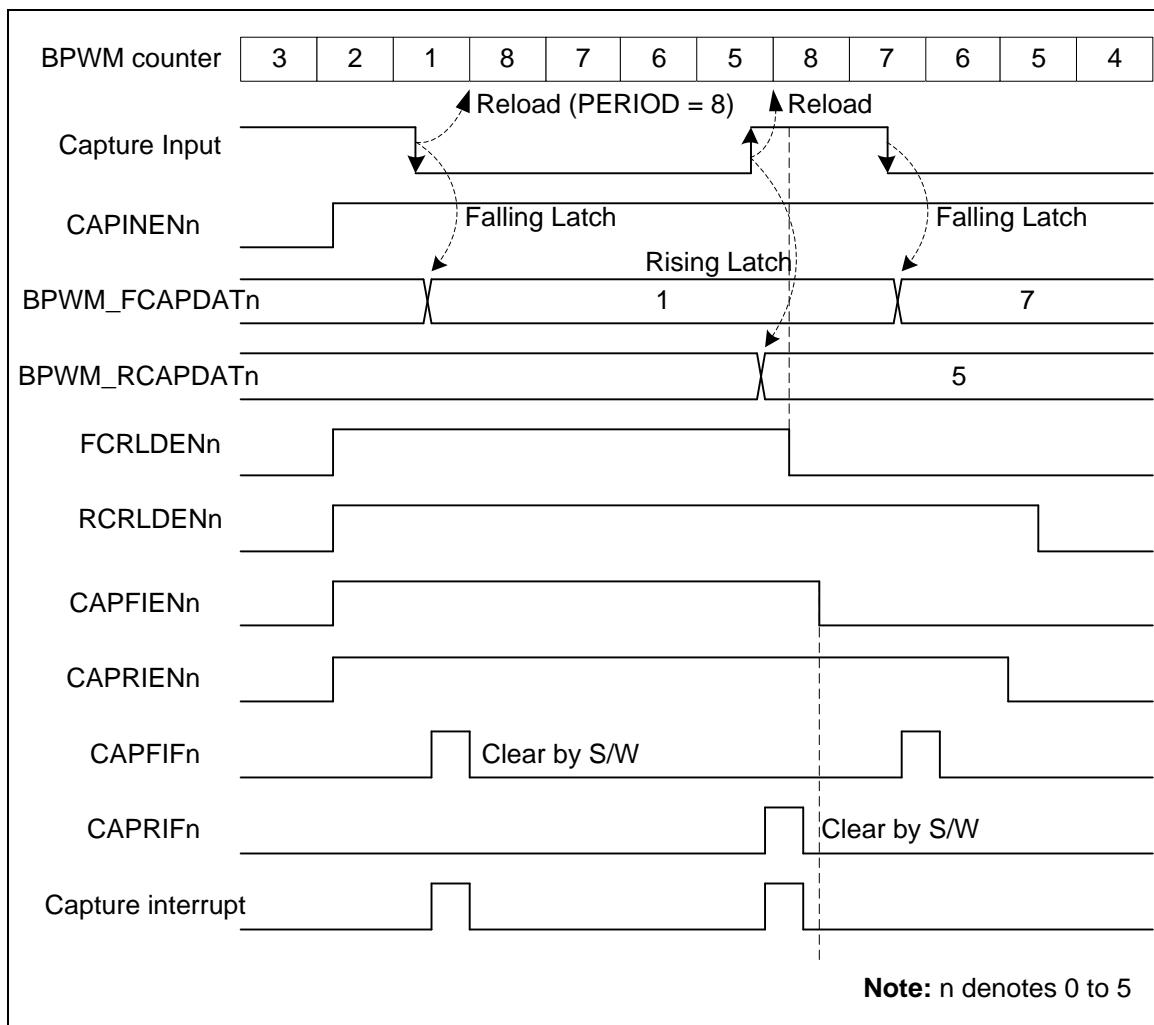


图 6-79 捕获操作波形图

捕获脉冲宽度计算，如下公式：

对于负脉冲的情况，通道低电平脉冲宽度为($\text{BPWM_PERIODn} + 1 - \text{BPWM_RCAPDATn}$)。在图 6-79 的情况，低脉冲的宽度是 $8+1-5 = 4$ 。

对于正脉冲的情况，通道高电平脉冲的宽度为($\text{BPWM_PERIODn} + 1 - \text{BPWM_FCAPDATn}$)。在图 6-79 的情况，高电平的宽度是 $8+1-7 = 2$ 。



6.8.5.18 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
BPWM 基地址:				
BPWM0_BA = 0x4004_4000				
BPWM1_BA = 0x4014_4000				
BPWM_CTL0 x=0, 1	BPWMx_BA+0x00	R/W	BPWM 控制寄存器 0	0x0000_0000
BPWM_CTL1 x=0, 1	BPWMx_BA+0x04	R/W	BPWM 控制寄存器1	0x0000_0000
BPWM_CLKSRC x=0, 1	BPWMx_BA+0x10	R/W	BPWM 时钟源寄存器	0x0000_0000
BPWM_CLKPSC x=0, 1	BPWMx_BA+0x14	R/W	BPWM 时钟预分频寄存器	0x0000_0000
BPWM_CNTEN x=0, 1	BPWMx_BA+0x20	R/W	BPWM 计数器使能寄存器	0x0000_0000
BPWM_CNTCLR x=0, 1	BPWMx_BA+0x24	R/W	BPWM 清计数器寄存器	0x0000_0000
BPWM_PERIOD x=0, 1	BPWMx_BA+0x30	R/W	BPWM 周期寄存器	0x0000_0000
BPWM_CMPDAT0 x=0, 1	BPWMx_BA+0x50	R/W	BPWM 比较寄存器0	0x0000_0000
BPWM_CMPDAT1 x=0, 1	BPWMx_BA+0x54	R/W	BPWM比较寄存器1	0x0000_0000
BPWM_CMPDAT2 x=0, 1	BPWMx_BA+0x58	R/W	BPWM比较寄存器2	0x0000_0000
BPWM_CMPDAT3 x=0, 1	BPWMx_BA+0x5C	R/W	BPWM比较寄存器3	0x0000_0000
BPWM_CMPDAT4 x=0, 1	BPWMx_BA+0x60	R/W	BPWM比较寄存器4	0x0000_0000
BPWM_CMPDAT5 x=0, 1	BPWMx_BA+0x64	R/W	BPWM比较寄存器5	0x0000_0000
BPWM_CNT0 x=0, 1	BPWMx_BA+0x90	R	BPWM 计数器寄存器 0	0x0000_0000

BPWM_WGCTL0 x=0, 1	BPWMx_BA+0xB0	R/W	BPWM 产生寄存器 0	0x0000_0000
BPWM_WGCTL1 x=0, 1	BPWMx_BA+0xB4	R/W	BPWM 产生寄存器 1	0x0000_0000
BPWM_MSKEN x=0, 1	BPWMx_BA+0xB8	R/W	BPWM 屏蔽使能寄存器	0x0000_0000
BPWM_MSK x=0, 1	BPWMx_BA+0xBC	R/W	BPWM 屏蔽数据寄存器	0x0000_0000
BPWM_POLCTL x=0, 1	BPWMx_BA+0xD4	R/W	BPWM 管脚极性翻转寄存器	0x0000_0000
BPWM_POEN x=0, 1	BPWMx_BA+0xD8	R/W	BPWM 输出使能寄存器	0x0000_0000
BPWM_INTEN x=0, 1	BPWMx_BA+0xE0	R/W	BPWM 中断使能寄存器	0x0000_0000
BPWM_INTSTS x=0, 1	BPWMx_BA+0xE8	R/W	BPWM 中断标志寄存器	0x0000_0000
BPWM_ADCTS0 x=0, 1	BPWMx_BA+0xF8	R/W	BPWM 触发ADC选择寄存器0	0x0000_0000
BPWM_ADCTS1 x=0, 1	BPWMx_BA+0xFC	R/W	BPWM 触发ADC选择寄存器1	0x0000_0000
BPWM_SSCTL x=0, 1	BPWMx_BA+0x110	R/W	BPWM 同步开始控制寄存器	0x0000_0000
BPWM_SSTRG x=0, 1	BPWMx_BA+0x114	W	BPWM 同步开始触发寄存器	0x0000_0000
BPWM_STATUS x=0, 1	BPWMx_BA+0x120	R/W	BPWM 状态寄存器	0x0000_0000
BPWM_CAPINE N x=0, 1	BPWMx_BA+0x200	R/W	BPWM 捕获输入使能寄存器	0x0000_0000
BPWM_CAPCTL x=0, 1	BPWMx_BA+0x204	R/W	BPWM 捕获控制寄存器	0x0000_0000
BPWM_CAPSTS x=0, 1	BPWMx_BA+0x208	R	BPWM 捕获状态寄存器	0x0000_0000
BPWM_RCAPDA T0 x=0, 1	BPWMx_BA+0x20C	R	BPWM 上升捕获数据寄存器0	0x0000_0000
BPWM_FCAPDA T0 x=0, 1	BPWMx_BA+0x210	R	BPWM 下降捕获数据寄存器0	0x0000_0000

BPWM_RCAPDA T1 x=0, 1	BPWMx_BA+0x214	R	BPWM 上升捕获数据寄存器1	0x0000_0000
BPWM_FCAPDA T1 x=0, 1	BPWMx_BA+0x218	R	BPWM 下降捕获数据寄存器1	0x0000_0000
BPWM_RCAPDA T2 x=0, 1	BPWMx_BA+0x21C	R	BPWM 上升捕获数据寄存器2	0x0000_0000
BPWM_FCAPDA T2 x=0, 1	BPWMx_BA+0x220	R	BPWM 下降捕获数据寄存器2	0x0000_0000
BPWM_RCAPDA T3 x=0, 1	BPWMx_BA+0x224	R	BPWM 上升捕获数据寄存器3	0x0000_0000
BPWM_FCAPDA T3 x=0, 1	BPWMx_BA+0x228	R	BPWM 下降捕获数据寄存器3	0x0000_0000
BPWM_RCAPDA T4 x=0, 1	BPWMx_BA+0x22C	R	BPWM 上升捕获数据寄存器4	0x0000_0000
BPWM_FCAPDA T4 x=0, 1	BPWMx_BA+0x230	R	BPWM 下降捕获数据寄存器4	0x0000_0000
BPWM_RCAPDA T5 x=0, 1	BPWMx_BA+0x234	R	BPWM 上升捕获数据寄存器5	0x0000_0000
BPWM_FCAPDA T5 x=0, 1	BPWMx_BA+0x238	R	BPWM 下降捕获数据寄存器5	0x0000_0000
BPWM_CAPIEN x=0, 1	BPWMx_BA+0x250	R/W	BPWM 捕获中断使能寄存器	0x0000_0000
BPWM_CAPIF x=0, 1	BPWMx_BA+0x254	R/W	BPWM 捕获中断标志寄存器	0x0000_0000
BPWM_PBUF x=0, 1	BPWMx_BA+0x304	R	BPWM PERIOD 缓冲器	0x0000_0000
BPWM_CMPBUF 0 x=0, 1	BPWMx_BA+0x31C	R	BPWM CMPDAT0缓冲器	0x0000_0000
BPWM_CMPBUF 1 x=0, 1	BPWMx_BA+0x320	R	BPWM CMPDAT1 缓冲器	0x0000_0000

BPWM_CMPBUF 2 x=0, 1	BPWMx_BA+0x32 4	R	BPWM CMPDAT 2 缓冲器	0x0000_0000
BPWM_CMPBUF 3 x=0, 1	BPWMx_BA+0x32 8	R	BPWM CMPDAT 3 缓冲器	0x0000_0000
BPWM_CMPBUF 4 x=0, 1	BPWMx_BA+0x32 C	R	BPWM CMPDAT 4 缓冲器	0x0000_0000
BPWM_CMPBUF 5 x=0, 1	BPWMx_BA+0x33 0	R	BPWM CMPDAT 5 缓冲器	0x0000_0000



6.8.6 寄存器描述

BPWM 控制寄存器 0 (BPWM_CTL0)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CTL0	BPWMx_BA+0x00	R/W	BPWM 控制寄存器 0	0x0000_0000

31	30	29	28	27	26	25	24
DBGTRIOFF	DBGHALT	Reserved					
23	22	21	20	19	18	17	16
Reserved	IMMLDEN5	IMMLDEN4	IMMLDEN3	IMMLDEN2	IMMLDEN1	IMMLDEN0	
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	CTRLD5	CTRLD4	CTRLD3	CTRLD2	CTRLD1	CTRLD0	

位	描述
[31]	DBGTRIOFF ICE 调试模式下，禁止应答(写保护) 0 = ICE 调试模式下，影响PWM输出。 当ICE 调试模式时，BPWM 管脚强制为三态门。 1 = ICE 调试模式，不影响PWM输出。 无论是否在ICE 调试模式，BPWM 管脚会保持输出。 注意： 这是写保护寄存器.. 参考 SYS_REGLCTL 寄存器。
[30]	DBGHALT ICE 调试模式下，计数器停止计数(写保护) 如果停止计数器计数，BPWM所有计数器会保持当前值直到ICE调试模式结束。 0 = ICE调试模式下，计数器继续计数。 1 = ICE调试模式下，计数器停止计数。 注意： 这是写保护寄存器.. 参考 SYS_REGLCTL 寄存器。
[29:22]	Reserved 保留
[21:16]	IMMLDENn 立即加载使能 每一位n控制相应的BPWM n通道。 0 = 在每个周期结束点，PERIOD会加载到PBUF。通过设置CTRLD 位，CMPDAT会在每个周期的结束点或中间点加载到CMPBUF。 1 = 当软件更新PERIOD/CMPDAT，PERIOD/ CMPDAT立即加载到PBUF和CMPBUF。 注意：如果IMMLDENn使能，WINLDENn 和 CTRLdn将无效。
[15:6]	Reserved 保留
[5:0]	CTRLDn 中间点重新加载 每一位n控制相应的BPWM n通道。 在可逆计数器的类型中，PERIOD将在每个周期结束点加载PBUF。CMPDAT将在周期的中间点加载CMPBUF。



BPWM 控制寄存器 1 (BPWM_CTL1)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CTL1	BPWMx_BA+0x04	R/W	BPWM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						CNTTYPE0	

位	描述	
[31:2]	Reserved	保留
[1:0]	CNTTYPE0	<p>BPWM 计数器类型 0</p> <p>每一位n控制相应的BPWM n通道</p> <p>00 = 递增计数器类型 (支持捕获模式) 01 = 递减计数器类型 (支持捕获模式) 10 = 可逆计数器类型 11 = 保留</p>



BPWM 时钟源寄存器 (BPWM_CLKSRC)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CLKSRC	BPWMx_BA+0x10	R/W	BPWM 时钟源寄存器	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					ECLKSRC0		

位	描述	
[31:3]	Reserved	保留
[2:0]	ECLKSRC0	<p>BPWM_CH01 外部时钟源选择</p> <p>000 = BPWMx_CLK, x 表示 0 或 1 001 = TIMER0 溢出 010 = TIMER1 溢出 011 = TIMER2 溢出 100 = TIMER3 溢出 Others = 保留</p>



BPWM 时钟分频寄存器 (BPWM_CLKPSC)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CLKPSC	BPWMx_BA+0x14	R/W	BPWM 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				CLKPSC			
7	6	5	4	3	2	1	0
CLKPSC							

位	描述	
[31:12]	Reserved	保留
[11:0]	CLKPSC	BPWM 计数器时钟分频 BPWM计数器的时钟由时钟分频器决定。每一对BPWM共享一个BPWM计数器时钟分频器。BPWM计数器的时钟是以 (CLKPSC+1) 除频。



BPWM 计数器使能寄存器 (BPWM_CNTEN)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CNTEN	BPWMx_BA+0x20	R/W	BPWM 计数器使能寄存器	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							CNTENO

位	描述	
[31:1]	Reserved	保留
[0]	CNTENO	BPWM 计数器使能 0 0 = BPWM 计数器和时钟分频器停止运行 1 = BPWM 计数器和时钟分频器开始运行



BPWM 清除计数器寄存器 (BPWM_CNTCLR)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CNTCLR	BPWMx_BA+0x24	R/W	BPWM 清除计数器寄存器	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							CNTCLR0

位	描述	
[31:1]	Reserved	保留
[0]	CNTCLR0	清除 BPWM 计数器 它由硬件自动清除 0 = 无影响 1 = 清除16位 BPWM 计数器到 0000H.

BPWM 周期寄存器 (BPWM_PERIOD)

寄存器	偏移地址	R/W	描述	复位值
BPWM_PERIOD	BPWMx_BA+0x30	R/W	BPWM 周期寄存器	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
PERIOD[15:8]							
7	6	5	4	3	2	1	0
PERIOD[7:0]							

位	描述	
[31:16]	Reserved	保留
[15:0]	PERIOD	<p>BPWM 周期寄存器</p> <p>递增计数模式：在这种模式下，BPWM 计数器从0开始，计数到周期值PERIOD，再从0重新计数</p> <p>递减计数模式：在这种模式下，BPWM计数器从周期值PERIOD开始，计数到0，再从周期值PERIOD重新计数</p> <p>BPWM周期时间= $(PERIOD + 1) * BPWM_CLK$周期。</p> <p>可逆计数模式：在这种模式下，BPWM计数器从0开始，计数到PERIOD，然后回数到0，然后再重复.....</p> <p>BPWM周期时间= $2 * PERIOD * BPWM_CLK$周期。</p>

BPWM 比较寄存器 0~5 (BPWM_CMPDAT0~5)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CMPDAT0	BPWMx_BA+0x50	R/W	BPWM 比较寄存器 0	0x0000_0000
BPWM_CMPDAT1	BPWMx_BA+0x54	R/W	BPWM 比较寄存器1	0x0000_0000
BPWM_CMPDAT2	BPWMx_BA+0x58	R/W	BPWM 比较寄存器2	0x0000_0000
BPWM_CMPDAT3	BPWMx_BA+0x5C	R/W	BPWM 比较寄存器3	0x0000_0000
BPWM_CMPDAT4	BPWMx_BA+0x60	R/W	BPWM 比较寄存器4	0x0000_0000
BPWM_CMPDAT5	BPWMx_BA+0x64	R/W	BPWM 比较寄存器5	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	CMPDAT	<p>BPWM 比较寄存器</p> <p>CMPDAT用于与CNT比较，来产生BPWM波形、中断和触发ADC.....</p> <p>在独立模式，CMPDAT0~5表示为6个独立的BPWM_CH0~5 相比较的点。</p> <p>在互补模式，对应3个互补对BPWM_CH0 和 BPWM_CH1, BPWM_CH2 和 BPWM_CH3, BPWM_CH4 和 BPWM_CH5, CMPDAT 0, 2, 4表示为第一个可比较的点， CMPDAT 1, 3, 5, 表示为第二个可比较的点。</p>

BPWM 计数寄存器 (BPWM_CNT)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CNT	BPWMx_BA+0x90	R	BPWM 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
CNT							
7	6	5	4	3	2	1	0
CNT							

位	描述	
[31:17]	Reserved	保留
[16]	DIRF	BPWM 方向指示标志 (只读) 0 = 计数器是向下计数器. 1 = 计数器是向上计数器.
[15:0]	CNT	BPWM 数据寄存器 (只读) 用户可以监控 CNT从而知道在16位周期计数器中的当前值



BPWM 生成寄存器 0 (BPWM_WGCTL0)

寄存器	偏移地址	R/W	描述	复位值
BPWM_WGCTL0	BPWMx_BA+0xB0	R/W	BPWM 生成寄存器 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PRDPCTL5		PRDPCTL4	
23	22	21	20	19	18	17	16
PRDPCTL3		PRDPCTL2		PRDPCTL1		PRDPCTL0	
15	14	13	12	11	10	9	8
Reserved				ZPCTL5		ZPCTL4	
7	6	5	4	3	2	1	0
ZPCTL3		ZPCTL2		ZPCTL1		ZPCTL0	

位	描述	
[31:28]	Reserved	保留
[27:16]	PRDPCTLn	<p>BPWM 周期 (中间) 点控制 每一位n控制相应的BPWM n通道 00 = 无操作. 01 = BPWM 周期 (中间) 点输出低. 10 = BPWM 周期 (中间) 点输出高. 11 = BPWM 周期 (中间) 点输出翻转. 当BPWM 计数器数到(PERIODn+1), BPWM可以控制输出电平.</p> <p>注意:当BPWM 计数器运行在可逆计数器类型时, 该位是中间点控制</p>
[15:12]	Reserved	保留
[11:0]	ZPCTLn	<p>BPWM 零点控制 每一位n控制相应的BPWM n通道 00 = 无操作. 01 = BPWM 零点输出低. 10 = BPWM 零点输出高. 11 = BPWM零点输出翻转. 当BPWM 计数器数到零时, BPWM可以控制输出电平</p>



BPWM 生成寄存器 1 (BPWM_WGCTL1)

寄存器	偏移地址	R/W	描述	复位值
BPWM_WGCTL1	BPWMx_BA+0xB4	R/W	BPWM 生成寄存器 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPDCTL5		CMPDCTL4	
23	22	21	20	19	18	17	16
CMPDCTL3		CMPDCTL2		CMPDCTL1		CMPDCTL0	
15	14	13	12	11	10	9	8
Reserved				CMPUCTL5		CMPUCTL4	
7	6	5	4	3	2	1	0
CMPUCTL3		CMPUCTL2		CMPUCTL1		CMPUCTL0	

位	描述	
[31:28]	Reserved	保留
[27:16]	CMPDCTLn	<p>BPWM 比较下降点控制</p> <p>每一位n控制相应的BPWM n通道</p> <p>00 = 无操作</p> <p>01 = BPWM 比较下降点输出低.</p> <p>10 = BPWM 比较下降点输出高.</p> <p>11 = BPWM 比较下降点输出翻转.</p> <p>当BPWM计数器下降数到CMPDAT, BPWM可以控制输出电平。</p> <p>注意: 在互补模式下, CMPDCTL1, 3, 5对应于CMPDCTL 0,2,4</p>
[15:12]	Reserved	保留
[11:0]	CMPUCTLn	<p>BPWM 比较上升点控制</p> <p>每一位n控制相应的BPWM n通道</p> <p>00 = 无操作</p> <p>01 = BPWM 比较上升点输出低</p> <p>10 = BPWM 比较上升点输出高</p> <p>11 = BPWM 比较上升点输出翻转</p> <p>当BPWM分频器上升数到CMPDAT, BPWM可以控制输出电平。</p> <p>注意: 在互补模式下, CMPDCTL1, 3, 5对应于CMPDCTL 0,2,4</p>

BPWM 屏蔽使能寄存器 (BPWM_MSKEN)

寄存器	偏移地址	R/W	描述	复位值
BPWM_MSKE_N	BPWMx_BA+0xB8	R/W	BPWM 屏蔽使能寄存器	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		MSKEN5	MSKEN4	MSKEN3	MSKEN2	MSKEN1	MSKEN0

位	描述	
[31:6]	Reserved	保留
[5:0]	MSKENn	<p>BPWM 屏蔽使能 每一位n控制相应的BPWM n通道</p> <p>当位使能时，BPWM的输出信号将会被屏蔽。相对应的BPWM n通道会输出MSKDATn (BPWM_MSKE[5:0]) 数据</p> <p>0 = BPWM 输出信号是非屏蔽的 1 = BPWM 输出信号是屏蔽的，并输出MSKDATn 数据</p>

BPWM 屏蔽数据寄存器 (BPWM_MSK)

寄存器	偏移地址	R/W	描述	复位值
BPWM_MSK	BPWMx_BA+0xBC	R/W	BPWM 屏蔽数据寄存器	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
Reserved		MSKDAT5	MSKDAT4	MSKDAT3	MSKDAT2	MSKDAT1	MSKDAT0

位	描述	
[31:6]	Reserved	保留
[5:0]	MSKDAT _n	<p>BPWM 屏蔽数据位</p> <p>如果对应屏蔽功能使能，那么数据位控制BPWM_n输出管脚的状态。每一位n控制相应的BPWM n 通道</p> <p>0 = BPWM_n输出逻辑低</p> <p>1 = BPWM_n输出逻辑高</p>

BPWM 管脚极反转控制寄存器 (BPWM_POLCTL)

寄存器	偏移地址	R/W	描述	复位值
BPWM_POLCTL	BPWMx_BA+0xD4	R/W	BPWM 管脚极反转控制寄存器	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		PINV5	PINV4	PINV3	PINV2	PINV1	PINV0

位	描述	
[31:6]	Reserved	保留
[5:0]	PINVn	<p>BPWM 管脚极反转控制</p> <p>寄存器控制BPWM输出的极性状态。每一位n控制相应的BPWM n通道</p> <p>0 = BPWM 输出极性反转禁用 1 = BPWM 输出极性反转使能</p>



BPWM 输出使能寄存器 (BPWM POEN)

寄存器	偏移地址	R/W	描述	复位值
BPWM_POEN	BPWMx_BA+0xD8	R/W	BPWM 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		POEN5	POEN4	POEN3	POEN2	POEN1	POEN0

位	描述	
[31:6]	Reserved	保留
[5:0]	POENn	BPWM 管脚输出使能 每一位n控制相应的BPWM n通道 0 = BPWM 管脚工作在三态门 1 = BPWM 管脚工作在输出模式



BPWM 中断使能寄存器 (BPWM_INTEN)

寄存器	偏移地址	R/W	描述	复位值
BPWM_INTEN	BPWMx_BA+0xE0	R/W	BPWM 中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIEN5	CMPDIEN4	CMPDIEN3	CMPDIEN2	CMPDIEN1	CMPDIENO
23	22	21	20	19	18	17	16
Reserved		CMPUIEN5	CMPUIEN4	CMPUIEN3	CMPUIEN2	CMPUIEN1	CMPUIENO
15	14	13	12	11	10	9	8
Reserved							PIENO
7	6	5	4	3	2	1	0
Reserved							ZIENO

位	描述	
[31:30]	Reserved	保留
[29:24]	CMPDIEnn	<p>BPWM 递减比较计数器中断使能 每位n控制相应的BPWM n通道 0 = 递减比较中断禁止 1 = 递减比较中断使能 注:在互补模式, CMPDIEN1, 3, 5对应 CMPDIEN 通道0, 2, 4.</p>
[23:22]	Reserved	保留
[21:16]	CMPUIEnn	<p>BPWM 递增比较中断使能 每位n控制相应的BPWM n通道 0 = 递增比较中断禁止 1 = 递增比较中断使能 注:在互补模式, CMPUIEN1, 3, 5对应CMPDIEN 通道0, 2, 4.</p>
[15:9]	Reserved	保留
[8]	PIENO	<p>BPWM 周期点中断使能0 0 = 周期点中断使能禁止 1 = 周期点中断使能使能 注:在可逆计时器类型周期点意味着中间点</p>
[7:1]	Reserved	保留
[0]	ZIENO	<p>BPWM 零点中断使能0 0 = 零点中断禁止 1 = 零点中断使能 注: 在互补模式, 奇数信道一直读0</p>



BPWM 中断标志寄存器 (BPWM_INTSTS)

寄存器	偏移地址	R/W	描述				复位值
BPWM_INTST S0	BPWMx_BA+0xE8	R/W	BPWM 中断标志寄存器 0				0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIF5	CMPDIF4	CMPDIF3	CMPDIF2	CMPDIF1	CMPDIF0
23	22	21	20	19	18	17	16
Reserved		CMPUIF5	CMPUIF4	CMPUIF3	CMPUIF2	CMPUIF1	CMPUIF0
15	14	13	12	11	10	9	8
Reserved							PIFO
7	6	5	4	3	2	1	0
Reserved							ZIFO

位	描述	
[31:30]	Reserved	保留
[29:24]	CMPDIFn	<p>BPWM递减比较中断标志 每位n控制相应的BPWM n通道 当BPWM 计数器递减并且达到BPWM_CMPDATn, 标志通过硬件置起, 软件可以写1清该位。</p> <p>注1: 如果CMPDAT与PERIOD相等, 这个标志在递减类型中无效。 注2: 在互补模式, CMPUIEN1, 3, 5对应CMPDIEN 通道0, 2, 4.</p>
[23:22]	Reserved	保留
[21:16]	CMPUIFn	<p>BPWM 比较递增中断标志 当BPWM递增并且达到BPWM_CMPDATn, 标志通过硬件置起, 软件可以写1清该位为0。 每位n控制相应的BPWM n通道.</p> <p>注1: 如果CMPDAT与PERIOD, 这个标志将在递增类型选择中无效。 注2: 在互补模式, CMPUIEN1, 3, 5对应CMPDIEN 通道0, 2, 4.</p>
[15:9]	Reserved	保留
[8]	PIFO	<p>BPWM 周期点中断标志 当BPWM_CH0计数器达到BPWM_PERIOD0时, 硬件置该位为1。软件可以写1清该位为0。</p>
[7:1]	Reserved	保留
[0]	ZIFO	<p>BPWM 零点中断标志0 当BPWM_CH0计数器达到0, 硬件置该位为1。软件可以写1清该位为0。</p>



BPWM 触发ADC选择寄存器0 (BPWM_ADCTS0)

寄存器	偏移地址	R/W	描述	复位值
BPWM_ADCTS0	BPWMx_BA+0xF8	R/W	BPWM 触发ADC选择寄存器0	0x0000_0000

31	30	29	28	27	26	25	24
TRGEN3	Reserved			TRGSEL3			
23	22	21	20	19	18	17	16
TRGEN2	Reserved			TRGSEL2			
15	14	13	12	11	10	9	8
TRGEN1	Reserved			TRGSEL1			
7	6	5	4	3	2	1	0
TRGEN0	Reserved			TRGSEL0			

位	描述	
[31]	TRGEN3	BPWM_CH3 触发 ADC 使能位
[30:28]	Reserved	保留
[27:24]	TRGSEL3	BPWM_CH3 触发ADC选择 0000 = BPWM_CH2的 0点 0001 = BPWM_CH2的周期点 0010 = BPWM_CH2的0或周期点 0011 = BPWM_CH2的递增 CMPDAT 点 0100 = BPWM_CH2的递减 CMPDAT 点 0101 = 保留 0110 = 保留 0111 = 保留 1000 = BPWM_CH3的递增 CMPDAT 点 1001 = BPWM_CH3的递减 CMPDAT 点 Others = 保留
[23]	TRGEN2	BPWM_CH2 触发 ADC 使能位
[22:20]	Reserved	保留
[19:16]	TRGSEL2	BPWM_CH2 触发ADC选择 0000 = BPWM_CH2的0点 0001 = BPWM_CH2的周期点 0010 = BPWM_CH2的0或周期点



		0011 = BPWM_CH2的递增 CMPDAT 点 0100 = BPWM_CH2的递减 CMPDAT 点 0101 = 保留 0110 = 保留 0111 = 保留 1000 = BPWM_CH3的递增 CMPDAT 点 1001 = BPWM_CH3的递减 CMPDAT 点 Others = 保留
[15]	TRGEN1	BPWM_CH1 触发 ADC 使能位
[14:12]	Reserved	保留
[11:8]	TRGSEL1	BPWM_CH1 触发 ADC 选择 0000 = BPWM_CH0的0点 0001 = BPWM_CH0的周期点 0010 = BPWM_CH0的0或周期点 0011 = BPWM_CH0的递增 CMPDAT 点 0100 = BPWM_CH0的递减 CMPDAT 点 0101 = 保留 0110 = 保留 0111 = 保留 1000 = BPWM_CH1的递增 CMPDAT 点 1001 = BPWM_CH1的递减 CMPDAT 点 Others = 保留
[7]	TRGEN0	BPWM_CH0 触发 ADC 使能位
[6:4]	Reserved	保留
[3:0]	TRGSEL0	BPWM_CH0 触发 ADC 选择 0000 = BPWM_CH0的0点 0001 = BPWM_CH0的周期点 0010 = BPWM_CH0的0或周期点 0011 = BPWM_CH0的递增 CMPDAT 点 0100 = BPWM_CH0的递减 CMPDAT 点 0101 = 保留 0110 = 保留 0111 = 保留 1000 = BPWM_CH1的递增 CMPDAT 点 1001 = BPWM_CH1的递减 CMPDAT 点 Others = 保留

**BPWM ADC选择寄存器1(BPWM_ADCTS1)**

寄存器	偏移地址	R/W	描述					复位值
BPWM_ADCTS1	BPWMx_BA+0xFC	R/W	BPWM 触发ADC选择寄存器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
TRGEN5	Reserved			TRGSEL5			
7	6	5	4	3	2	1	0
TRGEN4	Reserved			TRGSEL4			

位	描述	
[31:16]	Reserved	保留
[15]	TRGEN5	BPWM_CH5 触发 ADC 使能位
[14:12]	Reserved	保留
[11:8]	TRGSEL5	BPWM_CH5 触发ADC 选择 0000 = BPWM_CH4的0点 0001 = BPWM_CH4的周期点 0010 = BPWM_CH4的0或周期点 0011 = BPWM_CH4的递增 CMPDAT 点 0100 = BPWM_CH4的递减 CMPDAT 点 0101 = 保留 0110 = 保留 0111 = 保留 1000 = BPWM_CH5的递增 CMPDAT 点 1001 = BPWM_CH5的递减 CMPDAT 点 Others =保留
[7]	TRGEN4	BPWM_CH4 触发 ADC 使能位
[6:4]	Reserved	保留
[3:0]	TRGSEL4	BPWM_CH4 触发 ADC 选择 0000 = BPWM_CH4的0点 0001 = BPWM_CH4的周期点

	0010 = BPWM_CH4的0或周期点 0011 = BPWM_CH4的递增CMPDAT 点 0100 = BPWM_CH4的递减CMPDAT 点 0101 =保留 0110 =保留 0111 =保留 1000 = BPWM_CH5的递增 CMPDAT 点 1001 = BPWM_CH5的递减 CMPDAT 点 Others =保留
--	---

BPWM 同步开始控制寄存器 (BPWM_SSCTL)

寄存器	偏移地址	R/W	描述				复位值
BPWM_SSCTL	BPWMx_BA+0x10	R/W	BPWM 同步开始控制寄存器				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						SSRC	
7	6	5	4	3	2	1	0
Reserved						SSEN0	

位	描述	
[31:10]	Reserved	保留
[9:8]	SSRC	BPWM 同步启动选择 00 = 同步启动来源于BPWM0 01 = 同步启动来源于BPWM1 10 = 同步启动来源于BPWM0 11 = 同步启动来源于BPWM1
[7:1]	Reserved	保留
[0]	SSEN0	BPWM 同步开始功能使能0 当同步开始功能已经使能了，通过写BPWM同步开始触发位(CNTSEN)，可以将BPWM_CH0 计数器使能位(CNTENO)置位。 0 = BPWM 同步开始功能禁止 1 = BPWM 同步开始功能使能

BPWM 同步开始触发寄存器(BPWM_SSTRG)

寄存器	偏移地址	R/W	描述	复位值
BPWM_SSTRG	BPWMx_BA+0x14	W	BPWM 同步开始触发寄存器	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							CNTSEN

位	描述	
[31:1]	Reserved	保留
[0]	CNTSEN	<p>BPWM 计数器同步开始使能 (只写)</p> <p>PWM 计数器同步使能功能是，将选中的BPWM通道(包括BPWM0_CHx和BPWM1_CHx)启动同时计数。</p> <p>如果相关的BPWM通道计数器同步开始启动功能使能，向该位写1也可以置位计数器(CNTENn, n 代表通道 0 到 5)。</p> <p>注：该位仅出现在BPWM0_BA</p>



BPWM 状态寄存器(BPWM_STATUS)

寄存器	偏移地址	R/W	描述	复位值
BPWM_STAT US	BPWMx_BA+0x120	R/W	BPWM 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		ADCTRG5	ADCTRG4	ADCTRG3	ADCTRG2	ADCTRG1	ADCTRG0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							
CNTMAX0							

位	描述	
[31:22]	Reserved	保留
[21:16]	ADCTRGn	ADC 启动转换状态 每一位n控制相应的BPWM n 通道 0 =没有启动ADC转换触发中断 1 =启动了一个ADC转换触发中断，软件写1清该位
[15:1]	Reserved	保留
[0]	CNTMAX0	时间坐标计数器0等于0xFFFF(锁存状态) 0 = 时间坐标还没有达到最大值0xFFFF 1 = 时间坐标达到了最大值0xFFFF，软件可以写1清该位



BPWM 捕获输入使能寄存器 (BPWM_CAPINEN)

寄存器	偏移地址	R/W	描述					复位值
BPWM_CAPINEN	BPWMx_BA+0x200	R/W	BPWM捕获输入使能寄存器					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		CAPINEN5	CAPINEN4	CAPINEN3	CAPINEN2	CAPINEN1	CAPINEN0	

位	描述	
[31:6]	Reserved	保留
[5:0]	CAPINENn	<p>捕获输入使能</p> <p>每一位n控制相应的BPWM n 通道</p> <p>0 = BPWM 通道捕获输入禁用。输入的BPWM信道捕获功能一直认作0.</p> <p>1 = BPWM 通道捕获输入使能。输入的BPWM信道捕获功能取决于相关的多功能管脚设定。</p>



BPWM捕获控制寄存器 (BPWM_CAPCTL)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CAPCTL	BPWMx_BA+0x204	R/W	BPWM捕获控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		FCRLDEN5	FCRLDEN4	FCRLDEN3	FCRLDEN2	FCRLDEN1	FCRLDEN0
23	22	21	20	19	18	17	16
Reserved		RCRLDEN5	RCRLDEN4	RCRLDEN3	RCRLDEN2	RCRLDEN1	RCRLDEN0
15	14	13	12	11	10	9	8
Reserved		CAPINV5	CAPINV4	CAPINV3	CAPINV2	CAPINV1	CAPINV0
7	6	5	4	3	2	1	0
Reserved		CAPEN5	CAPEN4	CAPEN3	CAPEN2	CAPEN1	CAPEN0

位	描述	
[31:30]	Reserved	保留
[29:24]	FCRLDENn	下降捕获重载使能 每一位n控制相应的BPWM n通道 0 = 捕获到下降沿时，禁止重载计数器 1 = 捕获到下降沿时，启动重载计数器
[23:22]	Reserved	保留
[21:16]	RCRLDENn	上升捕获重载使能 每一位n控制相应的BPWM n通道 0 = 捕获到上升沿时，禁止重载计数器 1 = 捕获到上升沿时，启动重载计数器
[15:14]	Reserved	保留
[13:8]	CAPINVn	捕获反向使能 每一位n控制相应的BPWM n通道 0 = 捕获源反向禁用 1 = 捕获源反向使能，把GPIO输入的信号反向
[7:6]	Reserved	保留
[5:0]	CAPENn	捕获功能使能 每一位n控制相应的BPWM n通道 0 = 捕获功能禁用。RCAPDAT/FCAPDAT寄存器将不再更新 1 = 捕获功能使能。当侦测到输入信号为上升或下降沿，BPWM计数器数值将被捕获锁存，并且保存在RCAPDAT(上升锁存)和FCAPDAT(下降锁存)。



BPWM 捕获状态寄存器 (BPWM_CAPSTS)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CAPSTS	BPWMx_BA+0x208	R	BPWM 捕获状态寄存器	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		CFIFOV5	CFIFOV4	CFIFOV3	CFIFOV2	CFIFOV1	CFIFOV0
7	6	5	4	3	2	1	0
Reserved		CRIFOV5	CRIFOV4	CRIFOV3	CRIFOV2	CRIFOV1	CRIFOV0

位	描述	
[31:14]	Reserved	保留
[13:8]	CFIFOVn	<p>捕获下降中断标志溢出状态 (只读) 这个标志指示, 当相应的CAPFIF是1时, 是否发生了下降锁存。每一位n控制相应的BPWM n通道。 注: 当使用者清除相应的CAPFIF, 该位会被自动清除</p>
[7:6]	Reserved	保留
[5:0]	CRIFOVn	<p>捕获上升中断标志溢出状态 (只读) 这个标志指示, 当相应的CAPFIF是1, 是否发生了上升锁存。每一位n控制相应的BPWM n通道。 注: 当使用者清除相应的CAPFIF, 该位会被自动清除</p>



BPWM上升捕获数据寄存器 0~5 (BPWM_RCAPDAT 0~5)

寄存器	偏移地址	R/W	描述	复位值
BPWM_RCAPDAT0	BPWMx_BA+0x20C	R	BPWM 上升捕获数据寄存器0	0x0000_0000
BPWM_RCAPDAT1	BPWMx_BA+0x214	R	BPWM上升捕获数据寄存器1	0x0000_0000
BPWM_RCAPDAT2	BPWMx_BA+0x21C	R	BPWM上升捕获数据寄存器2	0x0000_0000
BPWM_RCAPDAT3	BPWMx_BA+0x224	R	BPWM上升捕获数据寄存器3	0x0000_0000
BPWM_RCAPDAT4	BPWMx_BA+0x22C	R	BPWM上升捕获数据寄存器4	0x0000_0000
BPWM_RCAPDAT5	BPWMx_BA+0x234	R	BPWM上升捕获数据寄存器5	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	RCAPDAT	BPWM上升捕获数据寄存器(只读) 当上升捕获条件发生了， BPWM计数器数值将被保存在这个寄存器。



BPWM下降捕获数据寄存器 0~5 (BPWM_FCAPDAT 0~5)

寄存器	偏移地址	R/W	描述	复位值
BPWM_FCAPDAT0	BPWMx_BA+0x210	R	BPWM下降捕获数据寄存器0	0x0000_0000
BPWM_FCAPDAT1	BPWMx_BA+0x218	R	BPWM下降捕获数据寄存器1	0x0000_0000
BPWM_FCAPDAT2	BPWMx_BA+0x220	R	BPWM下降捕获数据寄存器2	0x0000_0000
BPWM_FCAPDAT3	BPWMx_BA+0x228	R	BPWM下降捕获数据寄存器3	0x0000_0000
BPWM_FCAPDAT4	BPWMx_BA+0x230	R	BPWM下降捕获数据寄存器4	0x0000_0000
BPWM_FCAPDAT5	BPWMx_BA+0x238	R	BPWM下降捕获数据寄存器5	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	FCAPDAT	BPWM 下降捕获数据寄存器(只读) 当下降捕获条件发生，BPWM计数器数值会被存在这个寄存器

BPWM捕获中断使能寄存器(BPWM_CAPIEN)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CAPIEN	BPWMx_BA+0x250	R/W	BPWM捕获中断使能寄存器	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		CAPFIEN5	CAPFIEN4	CAPFIEN3	CAPFIEN2	CAPFIEN1	CAPFIENO
7	6	5	4	3	2	1	0
Reserved		CAPRIEN5	CAPRIEN4	CAPRIEN3	CAPRIEN2	CAPRIEN1	CAPRIENO

位	描述	
[31:14]	Reserved	保留
[13:8]	CAPFIENn	BPWM 捕获下降锁存中断使能 每一位n控制相应的BPWM n 通道 0 = 捕获下降沿锁存中断禁用 1 = 捕获下降沿锁存中断使能
[7:6]	Reserved	保留
[5:0]	CAPRIENn	BPWM 捕获上升锁存中断使能 每一位n控制相应的BPWM n 通道 0 = 捕获上升沿锁存中断禁用 1 = 捕获上升沿锁存中断使能



BPWM捕获中断标志寄存器(BPWM_CAPIF)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CAPIF	BPWMx_BA+0x25 ₄	R/W	BPWM 捕获中断标志寄存器	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		CAPFIF5	CAPFIF4	CAPFIF3	CAPFIF2	CAPFIF1	CAPFIFO
7	6	5	4	3	2	1	0
Reserved		CAPRIF5	CAPRIF4	CAPRIF3	CAPRIF2	CAPRIF1	CAPRIFO

位	描述	
[31:14]	Reserved	保留
[13:8]	CAPFIFn	<p>BPWM 捕获下降锁存中断标志</p> <p>该位写1清零。每一位n控制相应的BPWM n通道。</p> <p>0 = 没有捕获到下降锁存</p> <p>1= 捕获到下降锁存，这个标志位会设为高</p>
[7:6]	Reserved	保留
[5:0]	CAPRIFn	<p>BPWM 捕获上升锁存中断标志</p> <p>该位写1清零。每一位n控制相应的BPWM n通道。</p> <p>0 = 没有捕获到上升锁存</p> <p>1= 捕获到上升锁存，这个标志位会设为高</p>



BPWM周期寄存器缓冲区(BPWM_PBUF)

寄存器	偏移地址	R/W	描述				复位值
BPWM_PBUF	BPWMx_BA+0x30 4	R	BPWM PERIOD 缓冲区				0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	PBUF	BPWM 周期寄存器缓冲区（只读） 作为周期PERIOD 有效寄存器

BPWM比较缓冲区寄存器0~5 (BPWM_CMPBUF0~5)

寄存器	偏移地址	R/W	描述	复位值
BPWM_CMPBUF0	BPWMx_BA+0x31C	R	BPWM CMP0 缓冲区	0x0000_0000
BPWM_CMPBUF1	BPWMx_BA+0x320	R	BPWM CMP1 缓冲区	0x0000_0000
BPWM_CMPBUF2	BPWMx_BA+0x324	R	BPWM CMP2 缓冲区	0x0000_0000
BPWM_CMPBUF3	BPWMx_BA+0x328	R	BPWM CMP3 缓冲区	0x0000_0000
BPWM_CMPBUF4	BPWMx_BA+0x32C	R	BPWM CMP4 缓冲区	0x0000_0000
BPWM_CMPBUF5	BPWMx_BA+0x330	R	BPWM CMP5 缓冲区	0x0000_0000

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

位	描述	
[31:16]	Reserved	保留
[15:0]	CMPBUF	BPWM 比较缓冲区寄存器（只读） 作为CMP有效寄存器



6.9 看门定时狗 (WDT)

6.9.1 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。

6.9.2 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 $(1 / \text{WDT_CLK}) * 63$
- 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CWDTCEN (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

6.9.3 框图

看门狗定时器时钟控制和框图如下：

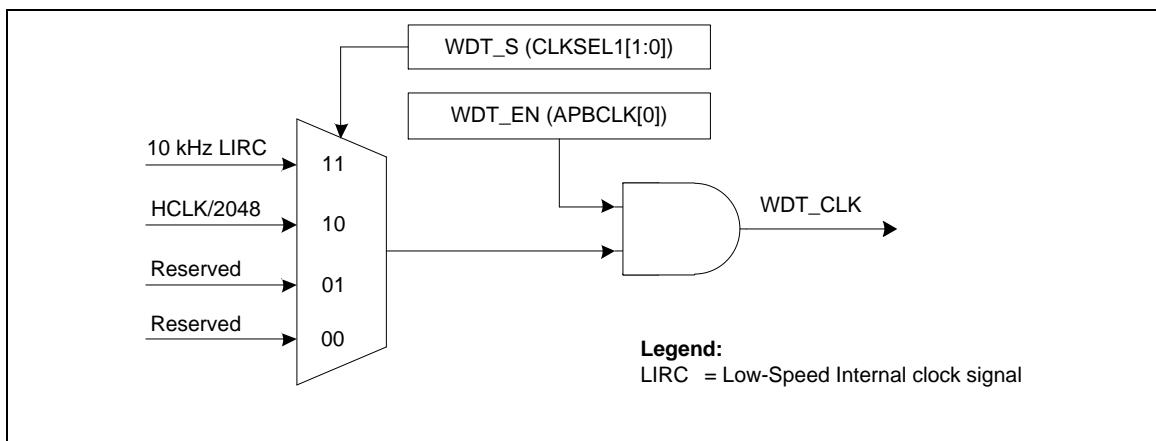


图 6-80 看门狗定时器时钟控制

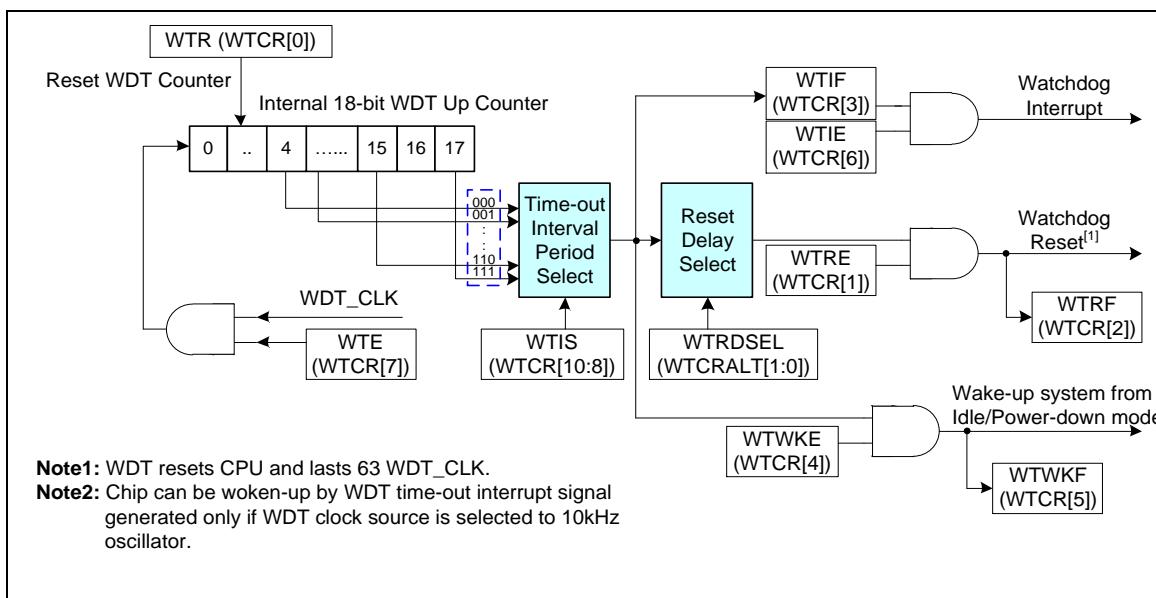


图 6-81 看门狗定时器框图

6.9.4 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

或者用户也可以把CONFIG0[31]配置为0，从而在芯片上电或RESET后强制使能看门狗定时器，并使看门狗定时器以10k时钟为时钟源工作。

6.9.5 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6-35显示了WDT溢出时间间隔选择，图图 6-82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，可以选择8个不同的溢出时间间隔。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6-82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	定时溢出中段时间间隔 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 6-19 看门狗定时溢出间隔周期

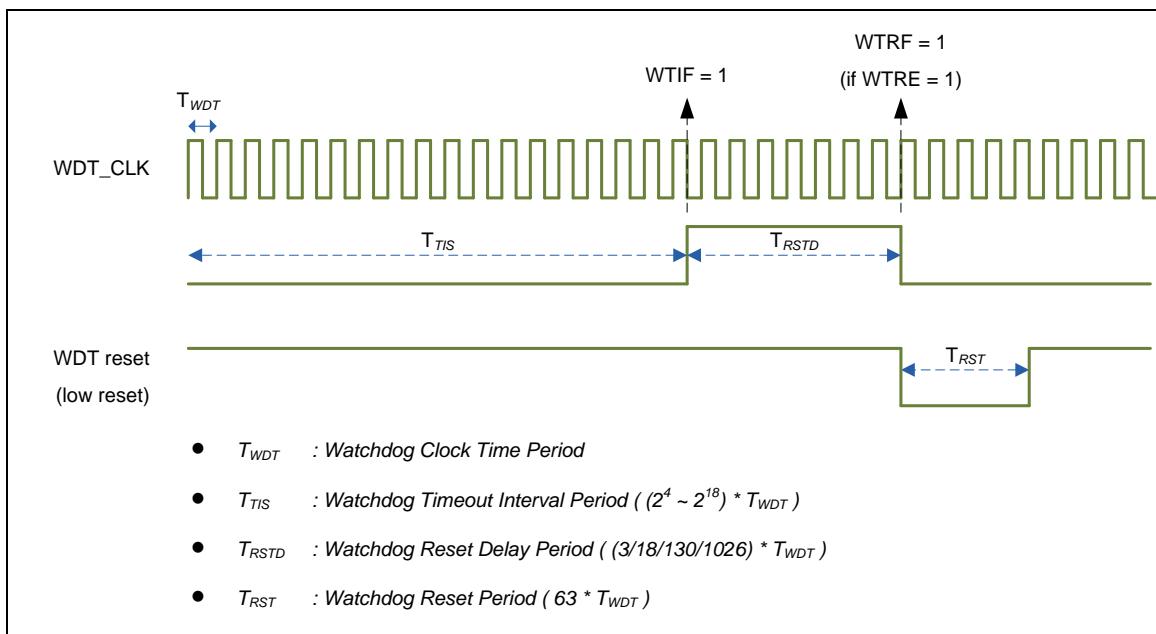


图 6-82 看门狗定时器定时溢出间隔和复位周期时序图



6.9.6 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
WDT基址:				
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



6.9.7 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CWDEN (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	<p>看门狗定时器复位延时选择 (写保护)</p> <p>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。</p> <p>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。</p> <p>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.</p> <p>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</p>



6.10 窗口看门狗定时器(WWDT)

6.10.1 概述

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。

6.10.2 特性

- 6位向下计数值(WWDTRVAL[5:0]) 和6位比较窗口值(WWDTCR[21:16])，使得窗口周期更加灵活
- 支持4位值选择看门狗预分频值，预分频计数器最大可达11位

6.10.3 框图

窗口看门狗定时器的时钟控制和框图如下：

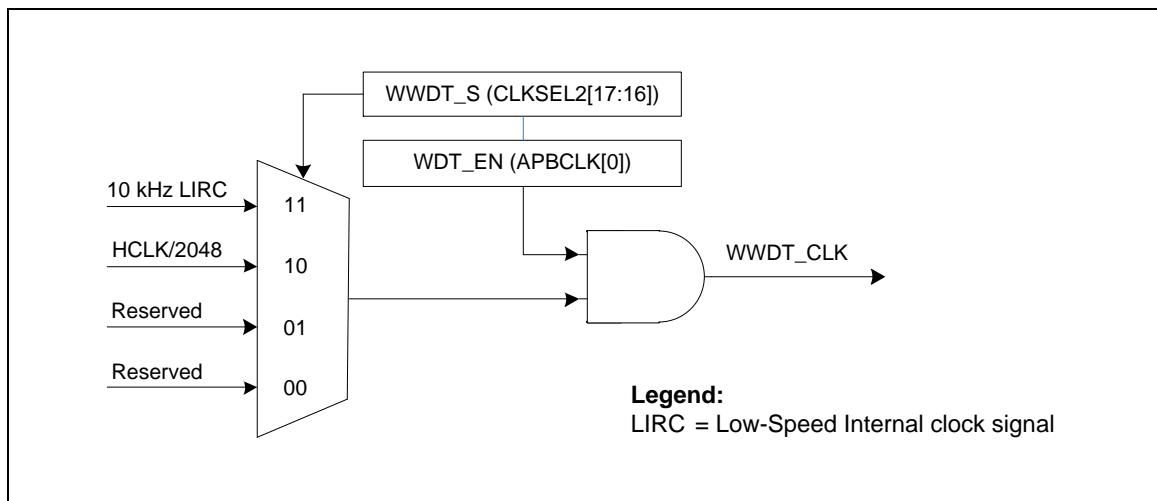


图 6-83 窗口看门狗定时器时钟控制图

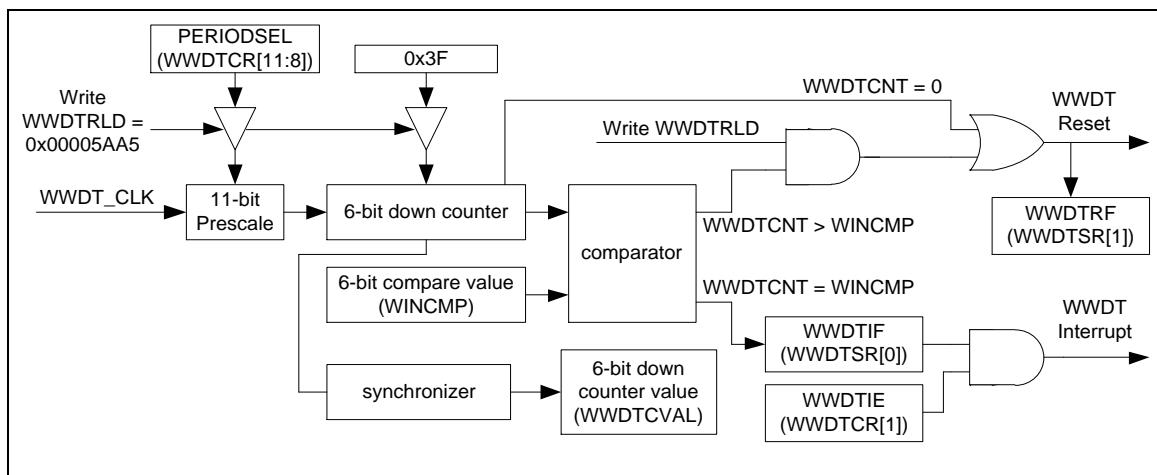


图 6-84 窗口看门狗定时器时钟框图



6.10.4 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。

6.10.5 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 (WWDTCTRL[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 (WWDT_CLK=10 KHz)	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 6-20 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 (WWDTCTRL[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(WWDTVAL) 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立刻产生，并导致芯片复位。

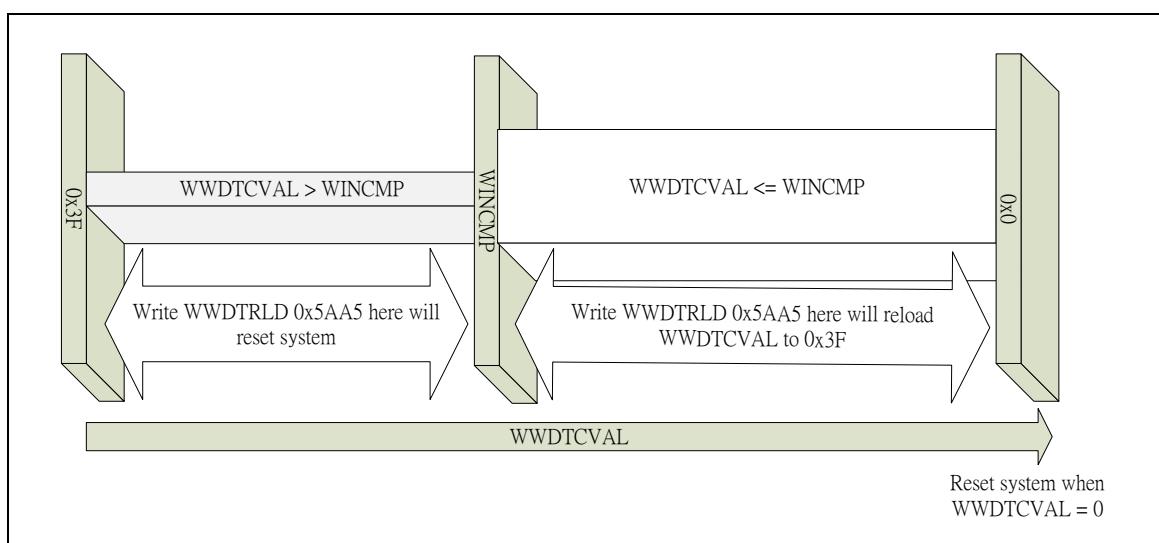


图 6-85 窗口看门狗定时器复位和重载过程

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
0000	1	0x3 ~ 0x3F
0001	2	0x2 ~ 0x3F
其它	其它	0x0 ~ 0x3F

表 6-21 WINCMP 寄存器设置限制



6.10.6 寄存器映射

R: 只读, **W:** 只写, **R/W:** 可读写

寄存器	偏移地址	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
WWDTCSR	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
WWDTCSR	WWDT_BA+0x0C	R	窗口看门狗定时器计数器值寄存器	0x0000_003F

6.10.7 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]	<p>WWDTRLD</p> <p>窗口看门狗定时器重载计数器寄存器</p> <p>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.</p> <p>注意:用户只能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。</p>



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被ICE暂停后，窗口看门狗向下计数器计数值将保持不变 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是否被ICE暂停。
[30:22]	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.
[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 $1 * 64 * \text{TWWDT}$. 0001 = 预分频为2; 最大定时溢出周期是 $2 * 64 * \text{TWWDT}$. 0010 = 预分频为4; 最大定时溢出周期是 $4 * 64 * \text{TWWDT}$. 0011 = 预分频为8; 最大定时溢出周期是 $8 * 64 * \text{TWWDT}$. 0100 = 预分频为16; 最大定时溢出周期是 $16 * 64 * \text{TWWDT}$. 0101 = 预分频为32; 最大定时溢出周期是 $32 * 64 * \text{TWWDT}$. 0110 = 预分频为64; 最大定时溢出周期是 $64 * 64 * \text{TWWDT}$. 0111 = 预分频为128; 最大定时溢出周期是 $128 * 64 * \text{TWWDT}$.

		1000 =预分频为192; 最大定时溢出周期是 $192 * 64 * \text{TWWDT}$. 1001 =预分频为256; 最大定时溢出周期是 $256 * 64 * \text{TWWDT}$. 1010 =预分频为384; 最大定时溢出周期是 $384 * 64 * \text{TWWDT}$. 1011 =预分频为512; 最大定时溢出周期是 $512 * 64 * \text{TWWDT}$. 1100 =预分频为768; 最大定时溢出周期是 $768 * 64 * \text{TWWDT}$. 1101 =预分频为1024; 最大定时溢出周期是 $1024 * 64 * \text{TWWDT}$. 1110 =预分频为1536; 最大定时溢出周期是 $1536 * 64 * \text{TWWDT}$. 1111 =预分频为2048; 最大定时溢出周期是 $2048 * 64 * \text{TWWDT}$.
[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	<p>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</p>
[0]	WWDTIF	<p>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</p>



窗口看门狗定时器计数器值寄存器 (WWDTCSR)

寄存器	偏移地址	读/写	描述	复位值
WWDTCSR	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		WWDTCSR					

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



6.11 UART 接口控制器 (UART)

6.11.1 概述

NuMicro™ NUC131系列提供多达 6 个通用异步串行接口(UART)。UART0/UART1/UART2支持16个字节FIFO，而UART3/UART4/UART5仅支持 1 个字节数据缓存。并且，只有UART0和UART1支持流量控制。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控制器还支持IrDA串行功能。UART0/UART1支持RS-485功能模式。UART0/UART1/UART2支持LIN主/从功能。

6.11.2 特性

- 全双工，异步通讯口
- 独立的接收/发送16/16字节FIFO（仅UART0/UART1/UART2支持），以及1/1字节数据缓存区（仅UART3/UART4/UART5支持）
- 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
- 可通过设置DLY (UA_TOR [15:8])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(UART0/UART1/UART2 支持)
 - 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(仅UART0和UART1支持)
 - 支持RS-485 9位模式

- 支持软硬件控制 RTS 管脚，用于控制 RS-485 传送方向

6.11.3 框图

串口时钟控制和内部模块框图分别如图 6-87 和 图 6-88 所示。

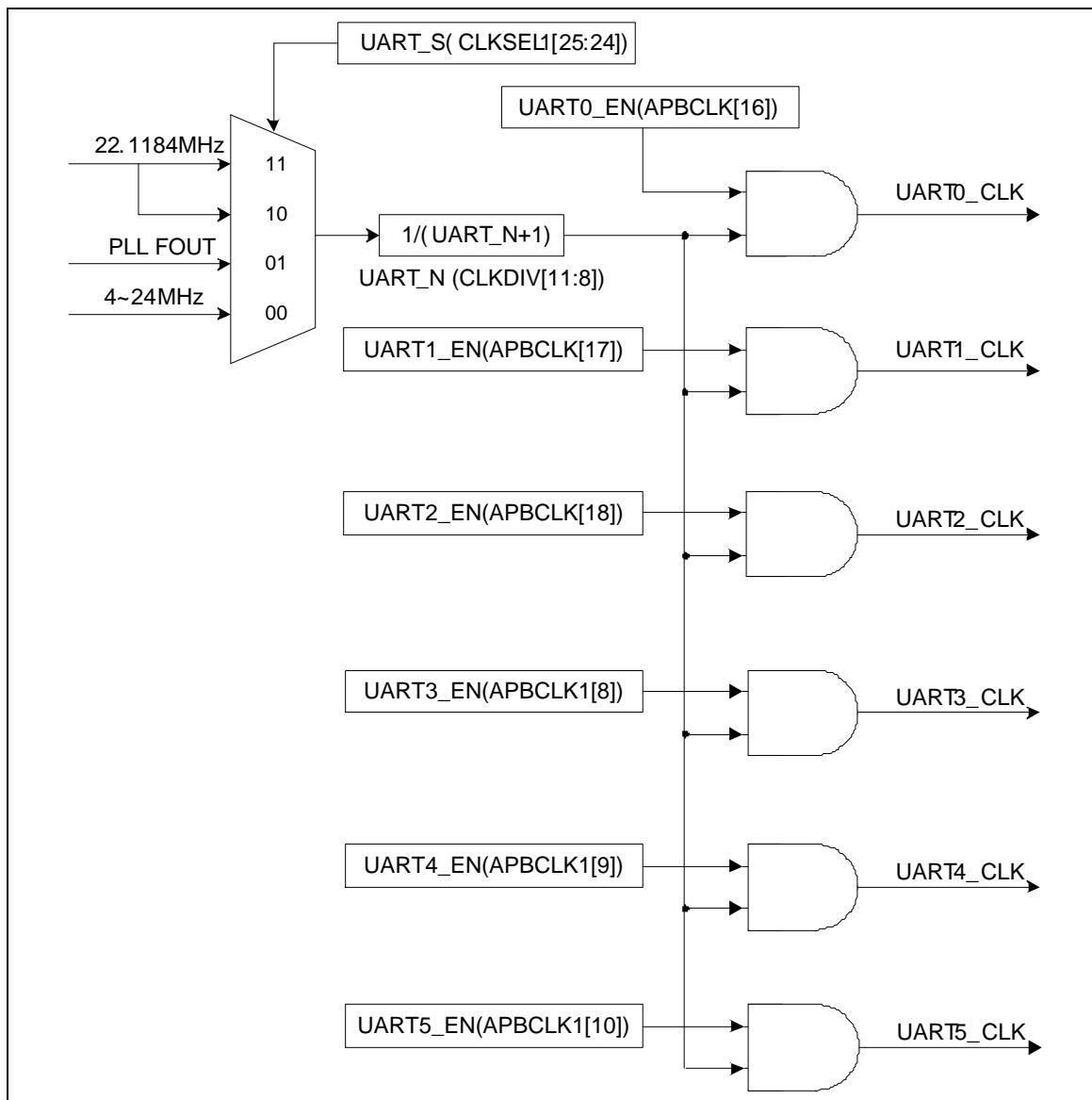


图 6-86 UART 串口时钟控制框图

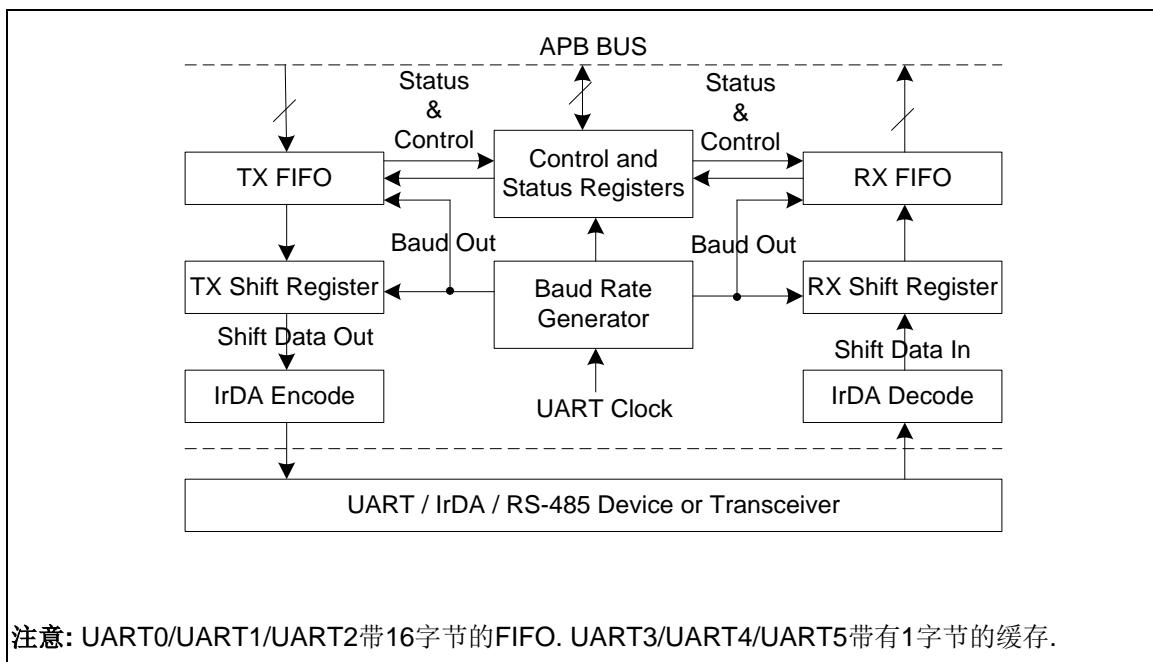


图 6-87 UART 串口模块框图

每个模块功能详细描述如下:

TX_FIFO

发送口带有一个16字节的FIFO缓冲区(UART0/UART1/UART2)或1个字节的缓存(UART3/UART4/UART5)以减少CPU中断的频率

RX_FIFO

接收口带有一个16字节的FIFO缓冲区(UART0/UART1/UART2)或1个字节的缓存(UART3/UART4/UART5)以减少CPU中断的频率

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.

该模块为IrDA编码控制模块

IrDA解码.



该模块为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)配置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, 用以设置产生相应中断。一共有10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), CTS唤醒中断(CTSWKIF), 数据唤醒中断(DATAWKIF)和自动波特率中断(ABRIF)。

6.11.4 基本配置

UART控制器功能脚在PB_MFP, PD_MFP, ALT_MFP, ALT_MFP 4 和ALT_MFP 5 寄存器中被配置。

UART 控制器时钟, 分别在 UART0_EN (APBCLK[16]), UART1_EN (APBCLK[17]), UART2_EN (APBCLK[18]), UART3_EN (APBCLK1[8]), UART4_EN (APBCLK1[9]) and UART5_EN (APBCLK1[10])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 6-22 串口接口控制脚

6.11.5 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

6.11.5.1 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	$\text{UART_CLK} / [16 * (\text{A}+2)]$.
1	1	0	B	A	$\text{UART_CLK} / [(B+1) * (\text{A}+2)]$, B必须 ≥ 8 .
2	1	1	无关	A	$\text{UART_CLK} / (\text{A}+2)$, 如果UART 外围时钟 $\leq 3 * \text{HCLK}$, A 必须 ≥ 9 . 如果UART 外围时钟 $> 3*\text{HCLK}$, A 必须 $\geq 3 * N - 1$. N为大于或等于 U A R T _CLK/HCLK的最小整数 例如: 如果 $3*\text{HCLK} < \text{UART_CLK} \leq 4*\text{HCLK}$, A必须 ≥ 11 . 如果 $4*\text{HCLK} < \text{UART_CLK} \leq 5*\text{HCLK}$, A必须 ≥ 14 .

表 6-23 UART 波特率公式

UART外围时钟 = 22.1184 MHz			
波特率	波特率	波特率	波特率
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 6-24 UART 控制器波特率参数设置表



UART 外围时钟 = 22.1184 MHz			
波特率	波特率	波特率	波特率
921600	不支持	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 6-25 UART 控制器波特率寄存器 (UA_BAUD) 设置表

自动波特率功能可以自动测量从UART RX管脚输入数据的波特率。当自动波特率测量结束，测量结果将会放置在BRD (UA_BAUD[15:0])中。DIV_X_EN (UA_BAUD[29]) 和 DIV_X_ONE (UA_BAUD[28])都被自动设置为1。在自动波特率检测帧中，UART RX数据从起始位到第一个上升沿的时间由**2 ABRDBITS**位时间设定。

从起始位到第一个上升沿之间的**2 ABRDBITS**位时间通过设定ABRDBITS (UA_ALT_CSR[20:19])计算得出。置位ABRDEN (UA_ALT_CSR[18])使能自动波特率检测功能。在初始阶段，UART RX保持为1。一旦检测到下降沿，为接收到起始位。自动波特率计数器被重启并开始计数。当检测到第一个上升沿时，自动波特率计数器将停止计数。然后，自动波特率计数器数值除以 ABRDBITS (UA_ALT_CSR[20:19]) 的结果将会自动载入到 BRD(UA_BAUD[15:0]) 中。ABRDEN (UA_ALT_CSR[18])会被清零。一旦自动波特率测量结束，ABRDIF (UA_FSR[1])会被置位。当自动波特率计数器溢出，ABRTOIF (UA_FSR[2]) 将会被置位。如果使能ABRIEN (UART_IER[18])，ABRDIF(UA_FSR[1]) 或 (UA_FSR[2])将会引发自动波特率中断ABRIF(UA_ALT_CSR[17])产生。

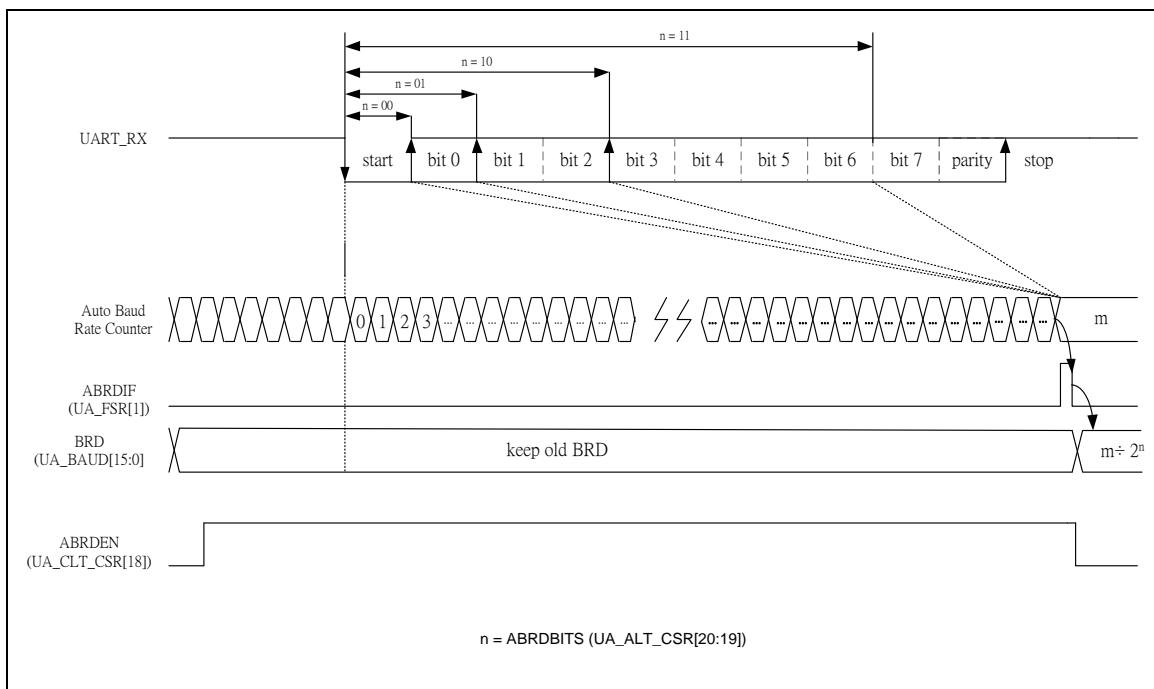


图 6-88 自动测量波特率

编程顺序范例：

1. 设置 ABRDBITS (UA_ALT_CSR[20:19]) 用于决定 UART RX 数据从起始到第一个上升沿时间用的 2 ABRDBITS 位时间
2. 置位 ABRIEN (UA_IER[18]) 用于使能自动波特率检测功能中断
3. 置位 ABRDEN (UA_ALT_CSR[18]) 用于使能自动波特率检测功能
4. 当 ABRDIF (UA_FSR[1]) 为 1，表示自动波特率测量结束
5. 执行 UART 发送和接收的动作
6. 当 ABRDTOIF (UA_FSR[2]) 为 1，表示自动波特率计数器溢出
7. 返回第二步

6.11.5.2 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。延时操作如图 6-89 所示

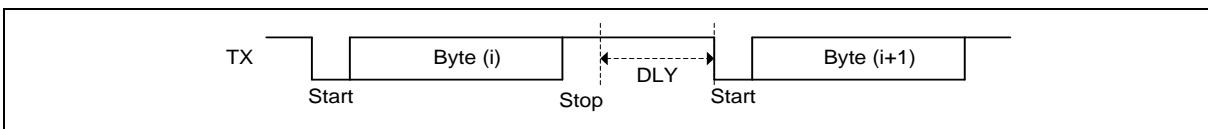


图 6-89 发送延时操作

6.11.5.3 UART 控制器 FIFO 控制和状态

UART0/UART1/UART2 内置一个 16 字节的发送 FIFO (TX_FIFO) 和一个 16 字节的接收 FIFO (RX_FIFO)，通讯中，使用这些收发 FIFO 可以减少对 CPU 的中断次数。UART3/UART4/UART5 仅有 1 个字节的发送缓存和 1 字节的接收缓存。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。

6.11.5.4 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。

6.11.5.5 UART控制器中断和状态

每个UART控制器支持10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状态中断(MODEM_INT) (UART0/UART1)
- 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) (仅UART0/UART1/UART2有效)
- CTS唤醒中断(CTSWKIF) (仅UART0/UART1有效)
- 数据唤醒中断(DATAWKIF)
- 自动波特率中断(ABRIF)

表6-25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 写“1”到 LIN_BKDET_F	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
nCTS 唤醒中断	N/A	WKCTSIEN	CTSWKIF	写‘1’到 CTSWKIF
数据唤醒中断	N/A	WKDATIEN	DATWKIF	写‘1’到 DATWKIF
自动波特率中断	N/A	ABRIEN	ABRIF = ABRDIF	写‘1’到 ABRDIF
			ABRIF = ABRDTOIF	写‘1’到 ABRDTOIF.

表 6-26 UART 控制器中断源

6.11.5.6 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。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位。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 6-27 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 6-28 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。

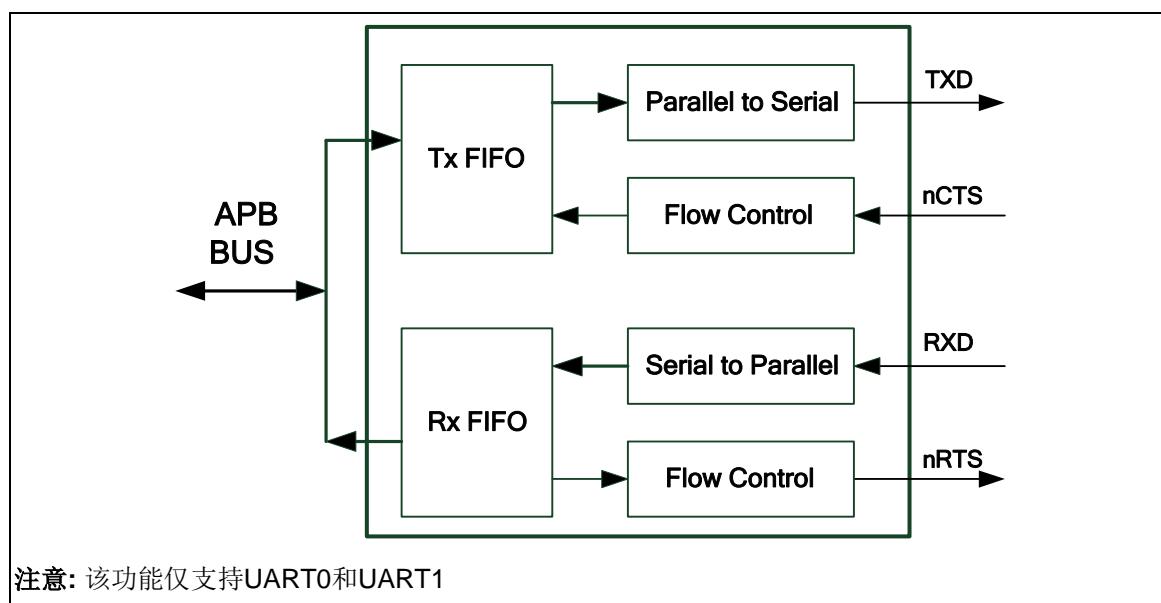


图 6-90 自动流控模块框图

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。

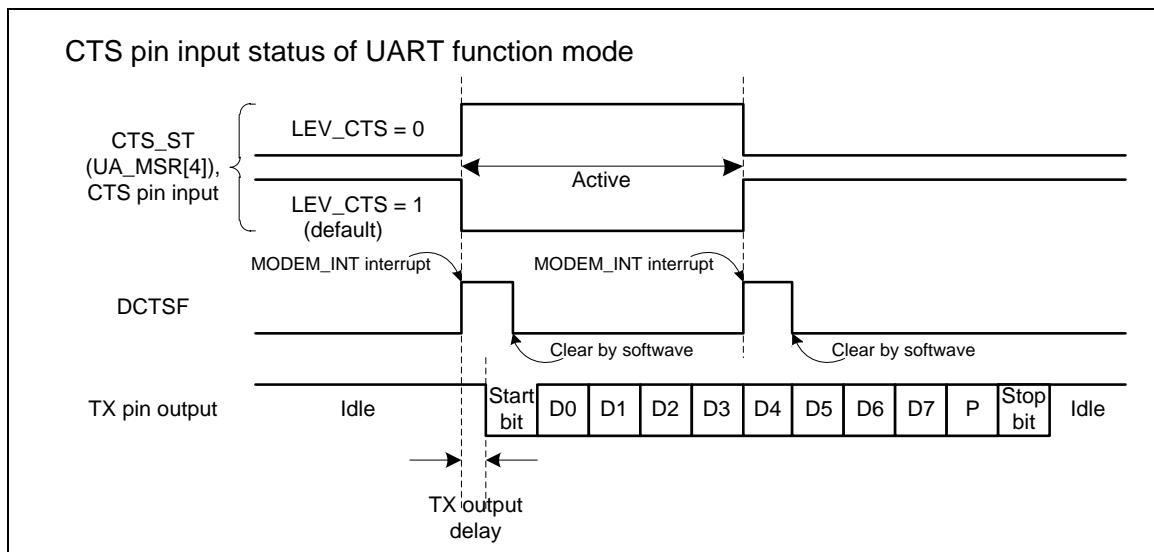


图 6-91 UART CTS 自动流控使能

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_RTLLEV(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。

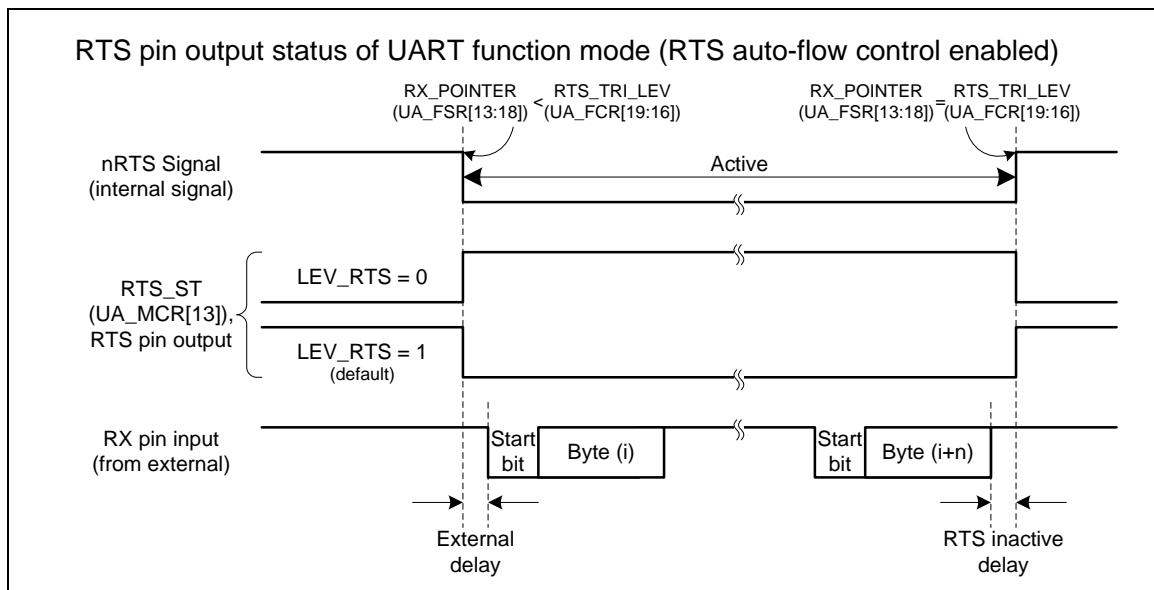


图 6-92 UART RTS 自动流控使能

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。

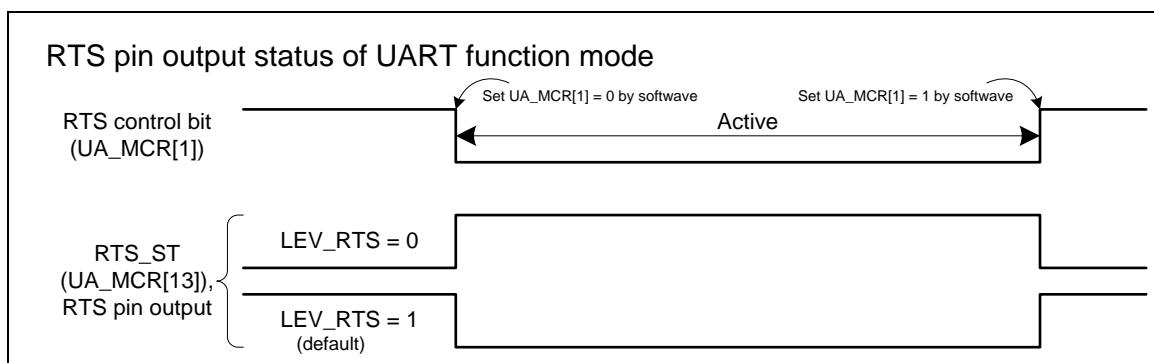


图 6-93 UART RTS 软件控制的流控

6.11.5.7 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：

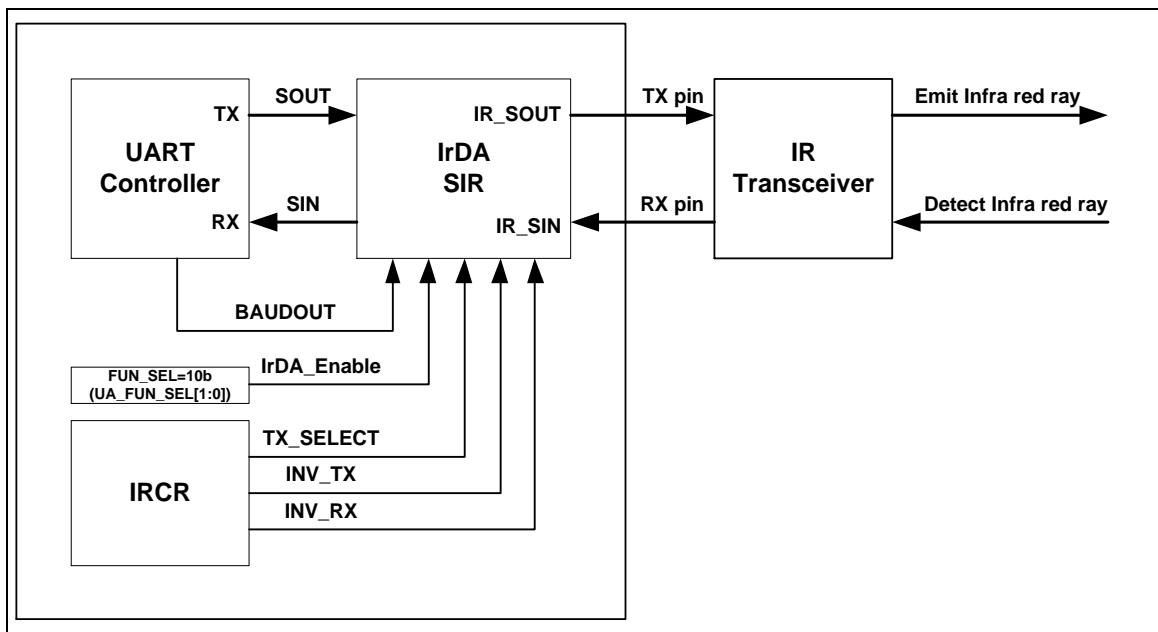


图 6-94 IrDA 控制模块框图

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：

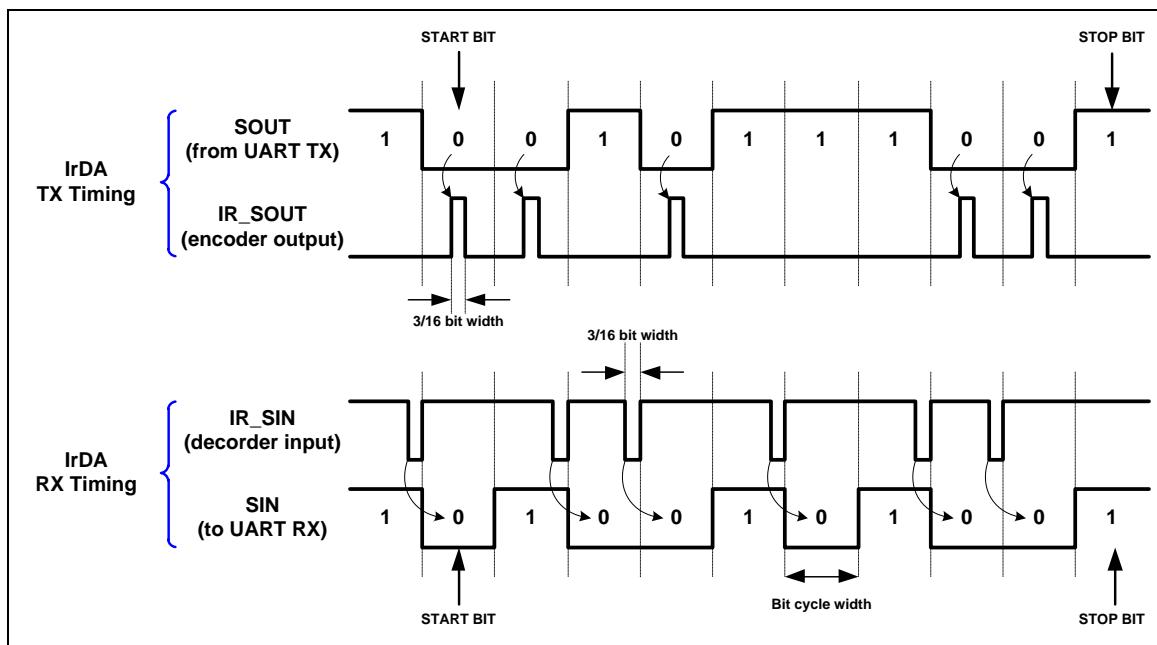


图 6-95 IrDA TX/RX 时序图

6.11.5.8 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。

6.11.5.8.1 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

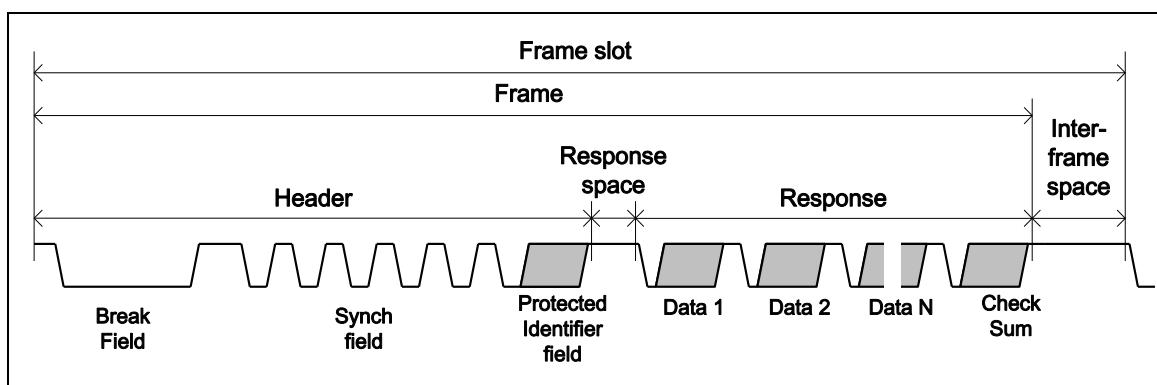


图 6-96 LIN 的帧结构

6.11.5.8.2 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：

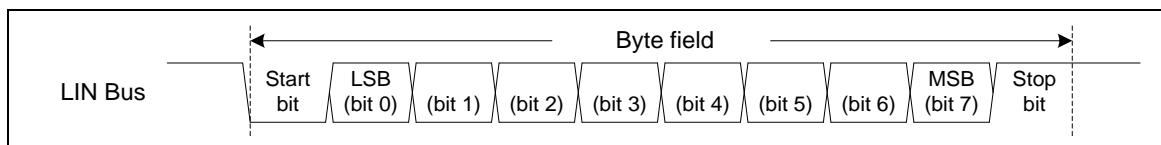


图 6-97 LIN 字节结构

6.11.5.8.3 LIN主机模式

UART0/UART1/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/UART2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，按顺序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 6-29 LIN 在主模式下的报头选择

当UART0/UART1/UART2 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])=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])为 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/UART2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。

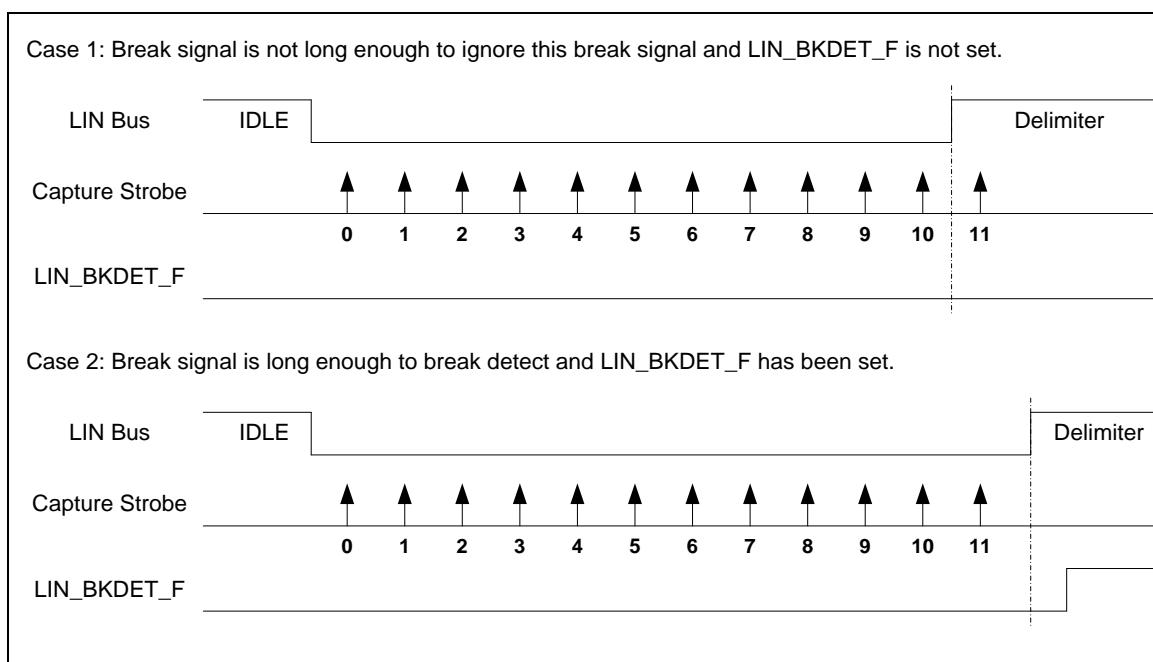


图 6-98 LIN 模式下 Break 检测

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

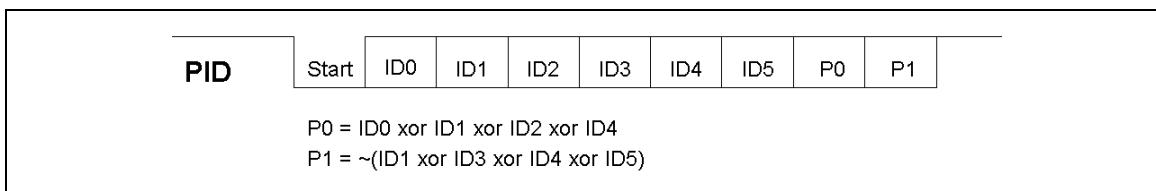


图 6-99 LIN 帧 ID 和奇偶校验格式

6.11.5.8.4 LIN从模式

UART0/UART1/UART2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

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: 建议校验字节发送之后, 将 LIN 从机节点设定为 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位 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位)溢出，报头错误标志LIN_HERR_F (UA_LIN_SR [1])将被置位。

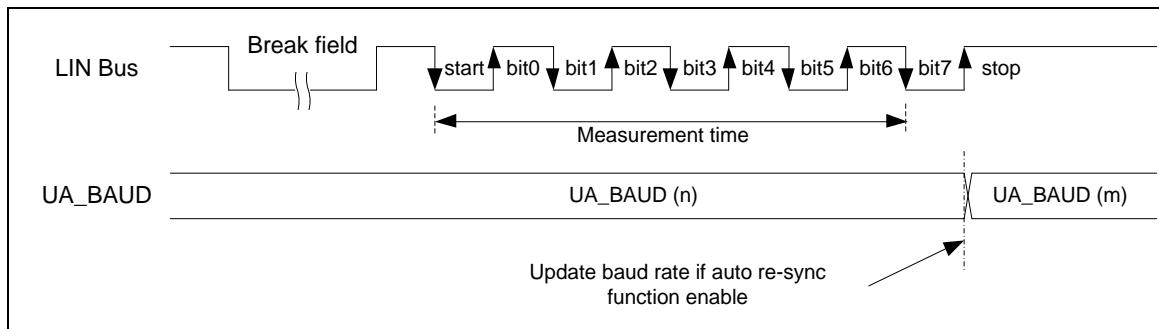


图 6-100 LIN 同步域的测量

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位 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)

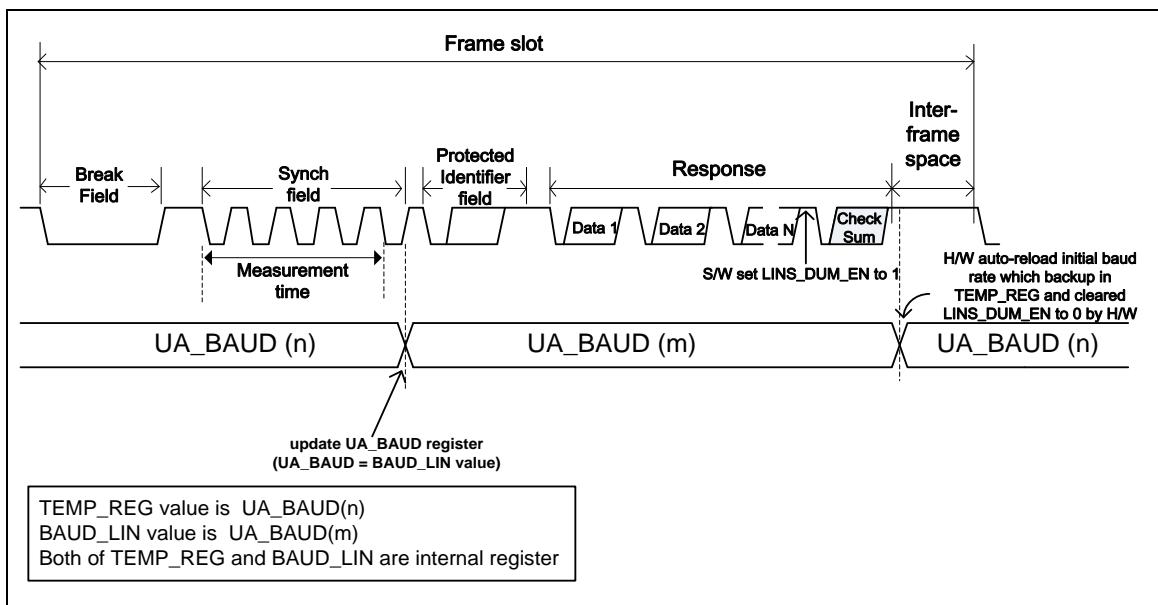


图 6-101 当LINS_DUM_EN (UA_LIN_CTL[3]) = 1 时自动重新同步模式下UA_BAUD 更新次序

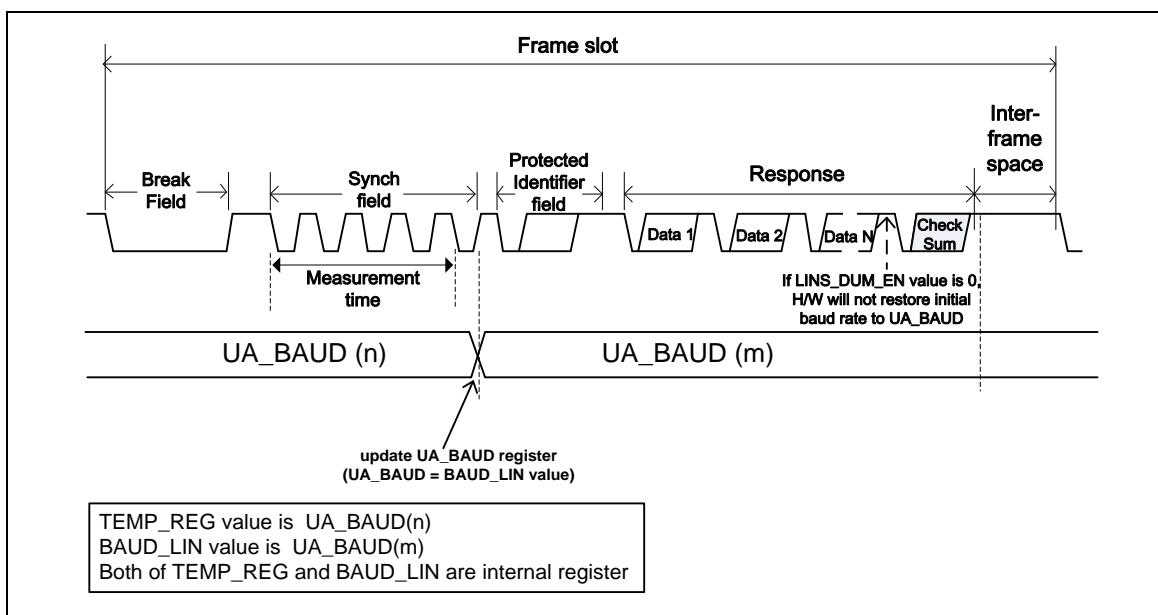


图 6-102 LINS_DUM_EN (UA_LIN_CTL[3])= 0 时自动重新同步模式下UA_BAUD的更新次序



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



6.11.5.9 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 (9th 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。



6.11.5.9.1 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。



6.11.5.9.2 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。

6.11.5.9.3 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。

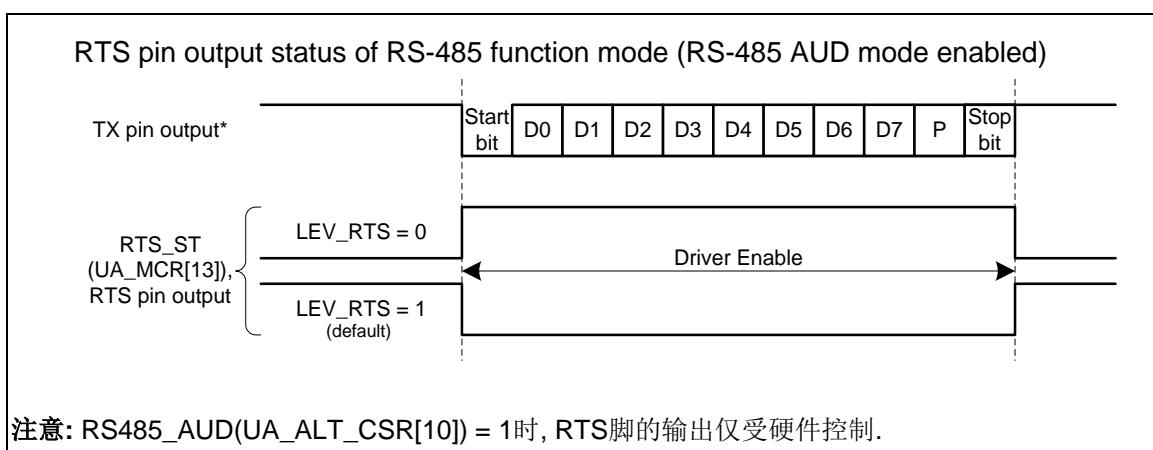


图 6-103 RS-485 自动方向模式下的 RS-485 RTS 驱动电平

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。用户可以通过读RTS_ST(UA_MCR[13])位来知道RTS脚上实际的逻辑电平。

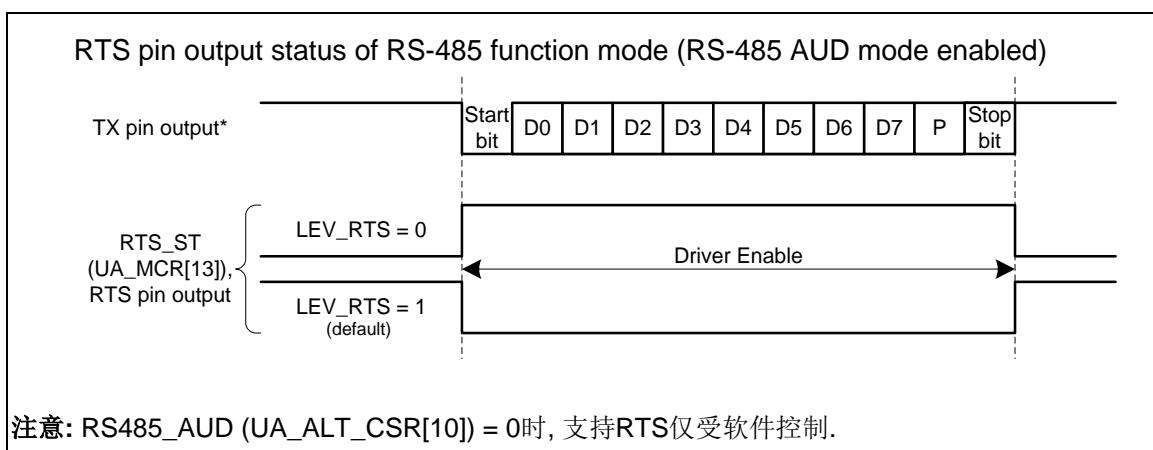


图 6-104 RS-485 RTS 软件控制下的驱动电平

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

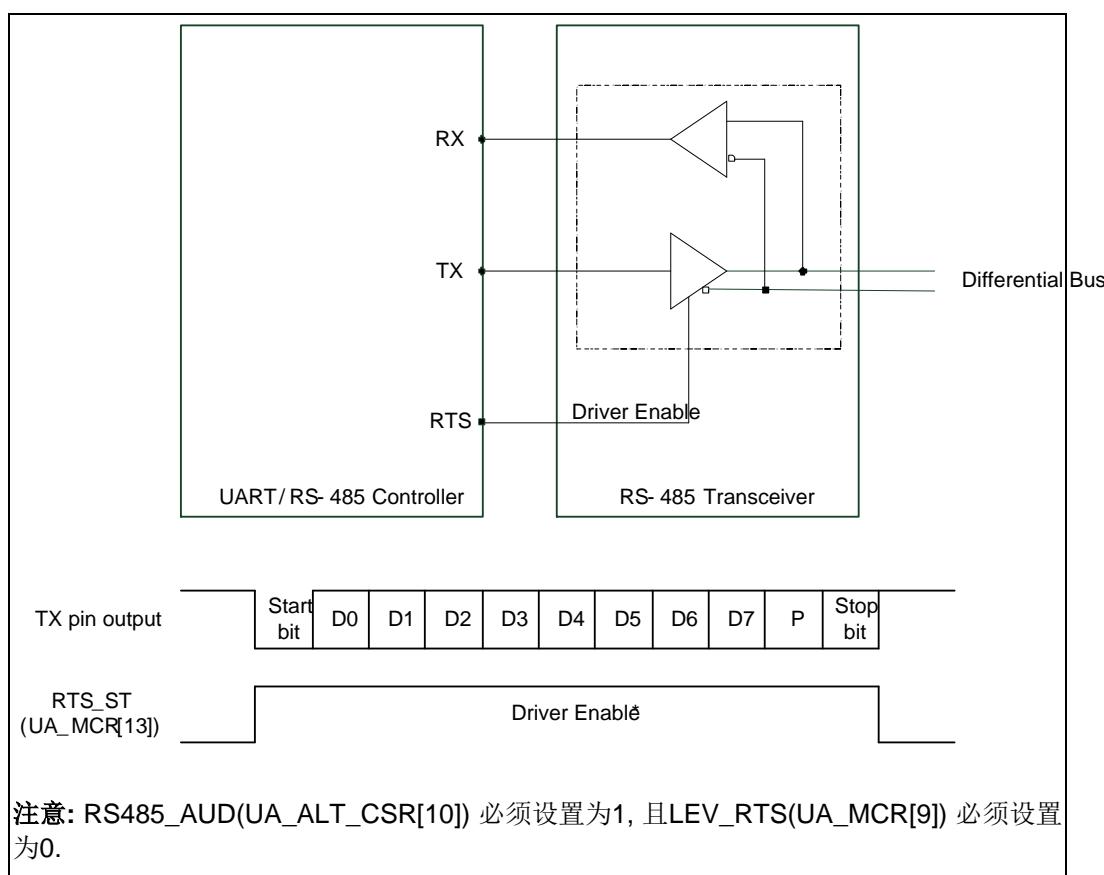


图 6-105 RS-485 帧结构



6.11.6 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
UART基址:				
UART0_BA = 0x4005_0000				
UART1_BA = 0x4015_0000				
UART2_BA = 0x4015_4000				
UART3_BA = 0x4005_4000				
UART4_BA = 0x4005_8000				
UART5_BA = 0x4015_8000				
UA_RBR x=0,1,2,3,4,5	UARTx_BA+0x00	R	UART 接收缓冲寄存器	未定义
UA_THR x=0,1,2,3,4,5	UARTx_BA+0x00	W	UART 发送保持寄存器	未定义
UA_IER x=0,1,2,3,4,5	UARTx_BA+0x04	R/W	UART 中断使能寄存器	0x0000_0000
UA_FCR x=0,1,2,3,4,5	UARTx_BA+0x08	R/W	UART FIFO 控制寄存器	0x0000_0101
UA_LCR x=0,1,2,3,4,5	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=0,1,2,3,4,5	UARTx_BA+0x18	R/W	UART FIFO状态寄存器	0x1040_4000
UA_ISR x=0,1,2,3,4,5	UARTx_BA+0x1C	R/W	UART Interrupt 状态寄存器	0x0000_0002
UA_TOR x=0,1,2,3,4,5	UARTx_BA+0x20	R/W	UART 定时溢出寄存器	0x0000_0000
UA_BAUD x=0,1,2,3,4,5	UARTx_BA+0x24	R/W	UART波特率分频寄存器	0x0F00_0000
UA_IRCR x=0,1,2,3,4,5	UARTx_BA+0x28	R/W	UART IrDA控制寄存器	0x0000_0040
UA_ALT_CSR x=0,1,2,3,4,5	UARTx_BA+0x2C	R/W	UART选择控制/状态寄存器	0x0000_000C

UA_FUN_SEL x=0,1,2,3,4,5	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



6.11.7 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=0,1,2,3,4,5	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=0,1,2,3,4,5	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=0,1,2,3,4,5	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					ABRIEN	Reserved	
15	14	13	12	11	10	9	8
保留		AUTO_CTS_EN	AUTO_RTS_EN	TIME_OUT_EN	WKDATIEN	保留	LIN_IEN
7	6	5	4	3	2	1	0
Reserved	WKCTSIEN	BUF_ERR_IE_N	TOUT_IEN	MODEM_IEN	RLS_IEN	THRE_IEN	RDA_IEN

位	描述	
[31:19]	保留	保留.
[18]	ABRIEN	自动波特率中断使能位 0 = 自动波特率中断禁止 1 = 自动波特率中断使能
[17:14]	保留	保留.
[13]	AUTO_CTS_EN	CTS自动流控使能位 (UART0/UART1通道有效) 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（UART将不会发送数据到外部设备直到CTS有效）。
[12]	AUTO_RTS_EN	RTS 自动流控使能位(UART0/UART1通道有效) 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_TRILEV (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]	WKDATIEN	输入数据唤醒中断使能控制 0 = 输入数据唤醒系统功能禁止。

		1 = 输入数据唤醒系统功能使能，当系统在Power-down模式时，输入数据可以将系统从Power-down模式唤醒。 注：当输入数据唤醒动作完成并且“系统时钟”工作稳定后，硬件将清除该位。
[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]	WKCTSIEN	nCTS 唤醒中断使能控制 0 = nCTS唤醒系统功能禁止. 1 = 唤醒系统功能使能，当系统在Power-down模式时，外部nCTS信号电平变化可以将系统从Power-down模式唤醒。
[5]	BUF_ERR_IEN	缓存错误中断使能 0 = BUF_ERR_INT 禁止 1 = BUF_ERR_INT 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 (UART0/UART1 通道有效) 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=0,1,2,3,4,5	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							
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	<p>RTS 自动流控触发阈值 (仅UART0/UART1通道有效)</p> <p>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. Others = 保留.</p> <p>注: 该区域用于自动 RTS 流控制</p>
[15:9]	保留	保留.
[8]	RX_DIS	<p>接收器禁止设置位</p> <p>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</p> <p>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</p>
[7:4]	RFITL	<p>RX FIFO 中断 (INT_RDA) 触发级别 (以下设置仅UART0/UART1/UART2通道有效)</p> <p>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)。</p> <p>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.</p>

		<p>其它 = 保留.</p> <p>RX FIFO 中断 (INT_RDA) 触发级别 (以下设置仅UART3/UART4/UART5通道有效)</p> <p>当接收缓存区内接收字节数等于 RFITL 后, RDA_IF 将被置位 (如果RDA_IEN (UA_IER[0]) 使能, 将产生中断)。</p> <p>0000 = RX FIFO 触发中断阈值为 1 byte.</p> <p>其它 = 保留.</p>
[3]	保留	保留.
[2]	TFR	<p>TX 软件复位</p> <p>当TFR置位, 发送 FIFO 内的所有数据和 TX 内部状态机将被清除。</p> <p>0 = 不影响.</p> <p>1 = 该位写 1 将复位 TX 内部状态机和指针</p> <p>注: 该位至少 3个 UART 时钟周期后会自动清零</p>
[1]	RFR	<p>RX 软件复位</p> <p>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。</p> <p>0 = 不影响.</p> <p>1 = 该位写 1 将复位 RX 内部状态机和指针.</p> <p>注: 该位至少 3个 UART 时钟周期后会自动清零。</p>
[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=0,1,2,3,4,5	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 0 = 禁用 Stick 奇偶校验 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。
[4]	EPE	Even 奇偶校验使能 0 = 逻辑 1 的奇数数目在每个字节中被发送和检验 1 = 逻辑 1 的偶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 0 = 无奇偶校验位 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
[2]	NSB	“停止位”个数 0= 当发生数据时，产生 1 “STOP bit” 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”
[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) (UART0/UART1 通道有效)

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) (UART0/UART1 通道有效) 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(UART0/UART1 通道有效) 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 6-928和图 6-93 注意2: RS-485 功能模式请参考图 6-103和 图 6-104
[8:2]	保留	保留.
[1]	RTS	RTS (请求发送) 信号控制(UART0/UART1 通道有效) 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) (UART0/UART1 通道有效)

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							
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	<p>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 6-91</p>
[7:5]	保留	保留.
[4]	CTS_ST	<p>CTS 脚状态(只读) 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。</p>
[3:1]	保留	保留.
[0]	DCTSF	<p>检测到 CTS 引脚电平改变标志 (只读) 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.</p>



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=0,1,2,3,4,5	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	ABRDTOIF	ABRDIF	RX_OVER_IF

位	描述	
[31:29]	保留	保留.
[28]	TE_FLAG	<p>发送空标志(只读)</p> <p>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_FLAG被硬件置1. (UART0/UART1/UART2)</p> <p>0 = TX FIFO 不为空. 1 = TX FIFO 为空.</p> <p>当TX 缓存 (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_FLAG被硬件置1. (UART3/UART4/UART5)</p> <p>0 = TX 缓存不为空. 1 = TX 缓存 为空.</p> <p>注意: 当TX FIFO/TX 缓存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。</p>
[27:25]	保留	保留.
[24]	TX_OVER_IF	<p>TX溢出错误中断标志 (只读)</p> <p>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. (UART0/UART1/UART2)</p> <p>0 = TX FIFO 未溢出 1 = TX FIFO 已溢出.</p> <p>如果TX缓存区已满, 如果再向UA_THR写入数据, 将会导致此位被置1. (UART3/UART4/UART5)</p> <p>0 = TX缓存 未溢出 1 = TX缓存 已溢出.</p>

		注意: 此位只读, 但可以写1清零。
[23]	TX_FULL	<p>发送FIFO满(只读)</p> <p>此位用于指示TX FIFO是否已满. (UART0/UART1/UART2)</p> <p>0 = TX FIFO 未满 1 = TX FIFO 已满.</p> <p>当TX FIFO Buffer中的字节数量等于16(UART0/UART1/UART2),此位将被置1.否则被硬件清零.</p> <p>此位用于指示TX 缓存区是否已满. (UART3/UART4/UART5)</p> <p>0 = TX 缓存 未满 1 = TX 缓存 已满.</p> <p>当TX 缓存中的字节数量等于1 (UART3/UART4/UART5),此位将被置1.否则被硬件清零.</p>
[22]	TX_EMPTY	<p>发送FIFO 空(只读)</p> <p>此位指示TX FIFO是否为空。 (UART0/UART1/UART2)</p> <p>0 = TX FIFO不为空. 1 = TX FIFO为空.</p> <p>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.当写入数据到THR中, (TX FIFO不为空), 此位被清零</p> <p>此位指示TX 缓存是否为空。 (UART3/UART4/UART5)</p> <p>0 = TX缓存不为空. 1 = TX 缓存为空.</p> <p>注意: 当TX 缓存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. 当写入数据到THR中, (TX 缓存不为空), 此位被清零</p>
[21:16]	TX_POINTER	<p>TX FIFO 指针(只读)</p> <p>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.</p> <p>TX_POINTER显示的最大值是15 (UART0/UART1/UART2)。当TX FIFO缓冲区所填充数据数量达到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15 (UART0/UART1/UART2)。</p> <p>TX_POINTER为0(UART3/UART4/UART5).</p> <p>当TX缓存区等于 1, 如果再接收到一个字节数据, TX_FULL将被置1, TX_POINTER显示为1。一旦TX缓存区被读取, TX_POINTER将被清零。</p>
[15]	RX_FULL	<p>接收 FIFO 满(只读)</p> <p>该位指示RX FIFO 是否已满(UART0/UART1/UART2)。</p> <p>0 = RX FIFO 未满 1 = RX FIFO 已满</p> <p>注意: 当RX FIFO缓冲区中的数据数量等于16(UART0/UART1/UART2)后, 此位将被置1, 否则被硬件清零。</p> <p>此位用于指示RX 缓存区是否已满. (UART3/UART4/UART5)</p> <p>0 = RX 缓存 未满</p>

		<p>1 = RX 缓存 已满. 当RX 缓存中的字节数量等于1 (UART3/UART4/UART5),此位将被置1.否则被硬件清零.</p>
[14]	RX_EMPTY	<p>接收 FIFO空(只读) 此位指示RX FIFO是否为空。 (UART0/UART1/UART2) 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。 此位指示RX 缓存是否为空。 (UART3/UART4/UART5) 0 =RX缓存不为空. 1 = RX 缓存为空. 注意: 当RX 缓存中最后一个字节被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。</p>
[13:8]	RX_POINTER	<p>RX FIFO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15 (UART0/UART1/UART2)。当RX FIFO的数据达到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15 (UART0/UART1/UART2)。 RX 缓存区的数据等于1时, 如果再接收到一个字节数据, RX_FULL位将被置1, RX_POINTER显示为1。一旦RX缓存区数据被读取, RX_POINTER被清除为0。</p>
[7]	保留	保留.
[6]	BIF	<p>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</p>
[5]	FEF	<p>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 =没有帧错误产生 1 =有帧错误产生. 注: 该位只读, 但是可以写 1 清零。</p>
[4]	PEF	<p>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。</p>

[3]	RS485_ADD_DETF	<p>RS-485 地址数据检测标志(只读) (仅UART0/UART1有效)</p> <p>0 = 接收到的数据没有地址位标记 (bit 9 ='1').</p> <p>1 = 接收到的数据有地址位标记 (bit 9 ='0').</p> <p>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</p> <p>注意2: 该位只读，但是可以写1清零。</p>
[2]	ABRDTOIF	<p>自动波特率时间溢出中断(只读)</p> <p>0 = 自动波特率计数器没有溢出.</p> <p>1 = 自动波特率计数器溢出.</p> <p>注意1: 在自动波特率检测模式下，当自动波特率计数器溢出时，该位将被置为1。</p> <p>注意2: 该位虽为只读位，但可以通过写1清除该位。</p>
[1]	ABRDIF	<p>自动波特率检测中断(只读)</p> <p>0 = 自动波特率检测还没有完成.</p> <p>1 = 自动波特率检测已经完成</p> <p>当自动波特率检测完成时该位将被置为“1”.</p> <p>注: 该位虽为只读位，但可以通过写1清除该位。</p>
[0]	RX_OVER_IF	<p>RX 溢出错误标志(只读)</p> <p>当RX FIFO溢出时被置1</p> <p>如果接收到的数据数量大于 RX_FIFO (UA_RBR) 16 字节 (UART0/UART1/UART2) , 该位将被置位.</p> <p>0 = RX FIFO未溢出.</p> <p>1 = RX FIFO溢出</p> <p>如果接收到的数据数量大于1个字节(UART3/UART4/UART5), 该位将被置位.</p> <p>0 = RX 缓存未溢出.</p> <p>1 = RX 缓存溢出</p> <p>注意: 该位只读，但是可以写1清零</p>



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=0,1,2,3,4,5	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							
23	22	21	20	19	18	17	16
保留						DATWKIF	CTSWKIF
15	14	13	12	11	10	9	8
LIN_INT	保留	BUF_ERR_IN_T	TOUT_INT	MODEM_INT	RLS_INT	THRE_INT	RDA_INT
7	6	5	4	3	2	1	0
LIN_IF	WKIF	BUF_ERR_IF	TOUT_IF	MODEM_IF	RLS_IF	THRE_IF	RDA_IF

位	描述	
[31:18]	保留	保留.
[17]	DATWKIF	<p>数据唤醒中断标志 (只读) 当数据传入将芯片从POWER-DOWN状态唤醒时，该位将被置位 0 = 芯片保持在POWER-DOWN状态 1 = 数据输入将芯片从POWER-DOWN状态唤醒 注意1：如果WKDATIEN (UA_IER[10])为使能状态，将会产生唤醒中断 注意2：该位为只读位，但可以写‘1’清零</p>
[16]	CTSWKIF	<p>nCTS唤醒中断标志 (只读) 0 = 芯片保持在POWER-DOWN状态 1 = nCTS将芯片从POWER-DOWN状态唤醒 注意1：如果WKCTSIEN (UA_IER[6])为使能状态，将会产生唤醒中断 注意2：该位为只读位，但可以写‘1’清零</p>
[15]	LIN_INT	<p>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.</p>
[14]	保留	保留.
[13]	BUF_ERR_INT	<p>Buffer错误中断标志(只读) 如果BUF_ERR_IEN(UA_IER[5] 和 BUF_ERR_IF(UA_ISR[5]))都被置1，该位置 1 0 =没有buffer错误中断产生.</p>

		1 =有buffer错误中断产生
[12]	TOUT_INT	<p>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.</p>
[11]	MODEM_INT	<p>MODEM状态中断标志(只读) (UART0/UART1通道有效) 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..</p>
[10]	RLS_INT	<p>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.</p>
[9]	THRE_INT	<p>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</p>
[8]	RDA_INT	<p>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.</p>
[7]	LIN_IF	<p>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 and 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_IDPERR_F (UA_LIN_SR[2]) 和 LINS_HERR_F(UA_LIN_SR[1])都被清0, 该位清0。</p>
[6]	WKIF	<p>UART唤醒标志 (只读) 当DATWKIF (UART_INTSTS[17]) 或 CTSWKIF(UART_INTSTS[16])为1时, 该位将被置位。 0 = 没有DATWKIF 和 CTSWKIF产生 1 = DATWKIF 或 CTSWKIF 为1 注: 该位为只读。当写1到DATWKIF (UART_INTSTS[17]) 和CTSWKIF (UART_INTSTS[17]), 将DATWKIF (UART_INTSTS[17]) 和CTSWKIF (UART_INTSTS[16])清零时, 该位将被清零</p>



[5]	BUF_ERR_IF	Buffer 错误中断标志(只读) 当TX FIFO (UART0/UART1/UART2) / TX 缓存(UART3/UART4/UART5)或RX FIFO(UART0/UART1/UART2) / RX 缓存 (UART3/UART4/UART5)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(UART0/UART1/UART2) / RX 缓存 (UART3/UART4/UART5) 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, 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 (UART0/UART1/UART2) / TX 缓存 (UART3/UART4/UART5)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)

	<p>当 RX FIFO (UART0/UART1/UART2) / RX 缓存 (UART3/UART4/UART5) 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。</p> <p>0 = 没有 RDA 中断标志产生。 1 = 有 RDA 中断标志产生。</p> <p>注: 该位只读, 当 RX FIFO 未读数据低于阈值(RFITL(UA_FCR[7:4]))时, 该位将被清除</p>
--	--



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=0,1,2,3,4,5	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 (UART0/UART1/UART2) / RX 缓存 (UART3/UART4/UART5)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 (UART0/UART1/UART2) / RX 缓存 (UART3/UART4/UART5)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=0,1,2,3,4,5	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 6-13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 6-23 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(IRCn) (UART0/UART1/UART2有效)

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=0,1,2,3,4,5	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			ABRDBITS		ABRDEN	ABRIF	Reserved
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 (UART0/UART1有效) 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
[23:21]	保留	保留.
[20:19]	ABRDBITS	自动波特率检测位长 00 = 1位时长, 从起始位到第一个 上升沿。 输入数据格式应该为0x01。 01 = 2位时长, 从起始位到第一个 上升沿。 输入数据格式应该为0x02。 10 = 4位时长, 从起始位到第一个 上升沿。 输入数据格式应该为0x08。 11 = 8位时长, 从起始位到第一个 上升沿。 输入数据格式应该为0x80。 注意: 计算的位数包括起始位。
[18]	ABRDEN	自动波特率检测使能控制 0 = 自动波特率检测功能禁止。 1 = 自动波特率检测功能使能 自动检测结束后, 该位将被自动清零。
[17]	ABRIF	自动波特率中断标志 (只读) 当自动波特率检测结束, 或者自动波特率计数器发生溢出, 如果ABRIEN(UART_IEN [18])为1, 那么自动波特率中断将会产生。 注: 该位为只读位, 但可以通过写“1”到ABRDTOIF (UA_FSR[2]) 和 ABRDIF(UA_FSR[1])将该位清零。
[16]	保留	保留.
[15]	RS485_ADD_EN	RS-485 地址检测使能位 (UART0/UART1有效)

		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) (UART0/UART1有效) 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) (UART0/UART1有效) 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) (UART0/UART1有效) 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 Break 模式使能 (UART0/UART1/UART2有效) 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 (UART0/UART1/UART2有效) 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 (UART0/UART1/UART2有效) 该域表示一个 4 位 LIN TX break 域计数 注1： 该break 域长度为 UA_LIN_BKFL + 1 注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=0,1,2,3,4,5	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	

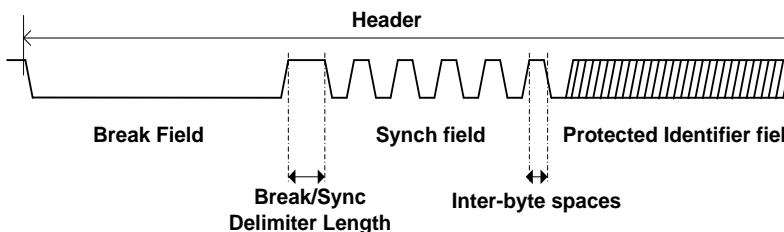
位	描述	
[31:2]	保留	保留.
[1:0]	FUN_SEL	功能选择使能位 00 = UART功能使能. 01 = LIN 功能使能 (UART0/UART1/UART2有效) 10 = IrDA功能使能 11 = RS-485功能使能. (UART0/UART1有效)

UART LIN控制寄存器(UA_LIN_CTL) (UART0/UART1/UART2有效)

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_ARSEN	LINS_HDET_EN	LINS_EN

位	描述										
[31:24]	<p>LIN PID寄存器</p> <p>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。</p> <p>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</p> <table border="1" style="margin-left: 20px;"> <tr> <td>PID</td> <td>Start</td> <td>ID0</td> <td>ID1</td> <td>ID2</td> <td>ID3</td> <td>ID4</td> <td>ID5</td> <td>P0</td> <td>P1</td> </tr> </table> <p>P0 = ID0 xor ID1 xor ID2 xor ID4 P1 = ~(ID1 xor ID3 xor ID4 xor ID5)</p> <p>注1: 用户可以填任何 8位 值到该域，位 24 表示 ID0 (LSB 优先)</p> <p>注2: 该域可以用在 LIN 主机模式或从机模式</p>	PID	Start	ID0	ID1	ID2	ID3	ID4	ID5	P0	P1
PID	Start	ID0	ID1	ID2	ID3	ID4	ID5	P0	P1		
[23:22]	<p>LIN帧头部选择</p> <p>00 = LIN帧头部包括“break 域”</p> <p>01 = LIN 帧头部包括“break 域”和“同步域”。</p> <p>10 = LIN 帧头部 包括“break域”，“同步域”域“帧ID域”。</p> <p>11 = 保留</p> <p>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))。</p>										
[21:20]	<p>LIN Break/同步分隔符长度</p> <p>00 = LIN break/同步分隔符长度为1 bit时间.</p> <p>10 = LIN break/同步分隔符长度为2 bit时间.</p>										

		<p>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.</p>  <p>Header Break Field Sync field Inter-byte spaces Protected Identifier field Break/Sync Delimiter Length</p> <p>注: 该位用于LIN 主机发送帧头部域</p>
[19:16]	LIN_BKFL	<p>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).</p>
[15:13]	保留	保留.
[12]	BIT_ERR_EN	<p>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</p>
[11]	LIN_RX_DIS	<p>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)</p>
[10]	LIN_BKDET_EN	<p>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.</p>
[9]	LIN_IDPEN	<p>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。</p>
[8]	LIN_SHD	LIN TX 发送帧头使能位

		<p>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]).</p> <p>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</p> <p>0 = LIN TX 帧头发送禁止.</p> <p>1 = LIN TX 帧头发送使能.</p> <p>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</p> <p>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.</p>
[7:5]	保留	保留.
[4]	LIN_MUTE_EN	<p>LIN Mute 模式使能位</p> <p>0 = LIN Mute 模式禁止</p> <p>1 = LIN Mute 模式使能.</p> <p>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</p>
[3]	LINS_DUM_EN	<p>LIN从机分频器更新方式使能</p> <p>0 = UA_BAUD 被软件更新(如果同时没有自动重同步发生)</p> <p>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.</p> <p>注意1: 该位仅在 LIN 从模式有效 (LINS_EN (UA_LIN_CTL[0]) = 1)</p> <p>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)</p> <p>注意3: 该域的控制和相互作用在6.11.5.8.4节中解释(带自动重同步的从机模式)</p>
[2]	LINS_ARS_EN	<p>LIN 从机自动重同步模式使能位</p> <p>0 = LIN从机自动重同步模式禁止</p> <p>1 = LIN从机自动重同步模式使能.</p> <p>注意1: 该位仅在 LIN 从机模式(LINS_EN (UA_LIN_CTL[0]) = 1) 有效.</p> <p>注意2: 当工作在自动重同步模式，波特率的设置必须是模式(BAUD_M1 (UA_BAUD [29]) 和 BAUD_M0 (UA_BAUD [28]))必须是 1).</p> <p>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)</p>
[1]	LINS_HDET_EN	<p>LIN从机报头侦测使能位</p> <p>0 = LIN 从机报头检测禁止.</p> <p>1 = LIN 从机报头检测使能.</p> <p>注意1: 该位仅在 LIN 从机模式有效 (LINS_EN (UA_LIN_CTL[0]) = 1)</p> <p>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</p>
[0]	LINS_EN	<p>LIN从模式使能位</p> <p>0 = LIN 从模式禁止</p> <p>1 = LIN 从模式使能</p>



UART LIN状态寄存器(UA_LIN_SR) (UART0/UART1/UART2有效)

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	<p>位错误侦测状态标志 (只读).</p> <p>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。</p> <p>当发生位错误, 如果LIN_IEN (UA_IER[8]) = 1, 会产生一个中断</p> <p>注意1: 该位只读, 向该位写1清0</p> <p>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)</p>
[8]	LIN_BKDET_F	<p>LIN Break 检测标志(只读)</p> <p>当一个break 侦测到, 该位由硬件置位, 通过软件写1清0</p> <p>0 = LIN break 没有被检测到</p> <p>1 = LIN break 被检测到.</p> <p>注意1: 该位只读, 向该位写1清0.</p> <p>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)</p>
[7:4]	保留	保留.
[3]	LINS_SYNC_F	<p>LIN 从机同步域.</p> <p>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</p> <p>0 = 当前字符不在 LIN同步状态.</p> <p>1 = 当前字符在 LIN 同步状态.</p> <p>注意1: 该位仅在 LIN 从机模式有效 (LINS_EN = 1)</p>

		<p>注意2: 该位只读, 向该位写1清0</p> <p>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</p>
[2]	LINS_IDPERR_F	<p>LIN 从机 ID 奇偶错误标志 (只读)</p> <p>当接收帧ID奇偶校验错误, 该位由硬件置1</p> <p>0 = 无效.</p> <p>1 = 收到的帧ID奇偶校验错误.</p> <p>注意1: 该位只读, 向该位写1清0..</p> <p>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]))</p>
[1]	LINS_HERR_F	<p>LIN从机报头错误标志 (只读).</p> <p>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”</p> <p>0 = LIN未检测到报头错误</p> <p>1 = LIN检测到报头错误.</p> <p>注意1: 该位只读, 向该位写1清0</p> <p>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]))</p>
[0]	LINS_HDET_F	<p>LIN 从机报头侦测标志 (只读)</p> <p>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</p> <p>0 = LIN 报头没有被侦测到.</p> <p>1 = LIN 报头被侦测到(break + 同步+ 帧ID)</p> <p>注意1: 该位只读, 向该位写1清0</p> <p>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]))</p> <p>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_HDET_F 会被置位, 无论帧ID是否正确</p>



6.12 I²C 总线控制器 (I²C)

6.12.1 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。

6.12.2 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

6.12.3 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

6.12.4 框图

I²C基本配置如下:

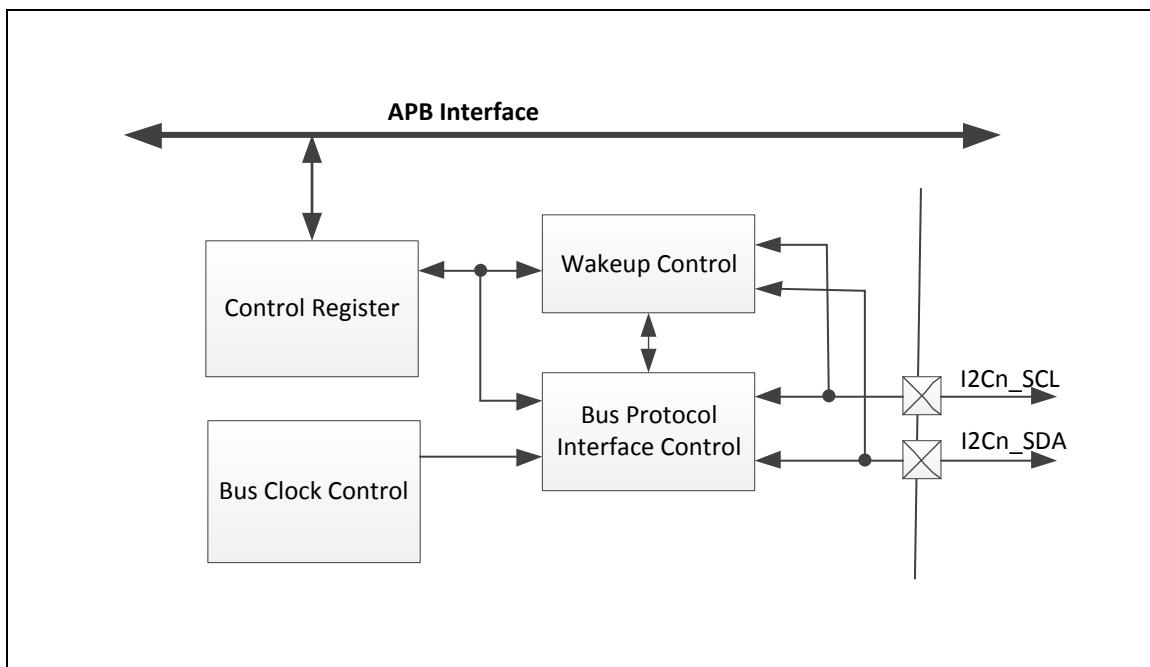


图 6-106 I²C 控制器模块框图

6.12.5 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。

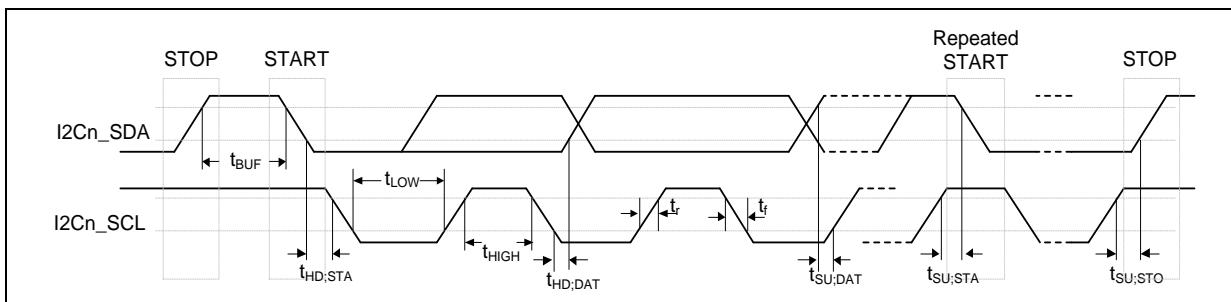


图 6-107 I²C 总线时序

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，因为这两个管脚为开漏脚。

6.12.5.1 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)

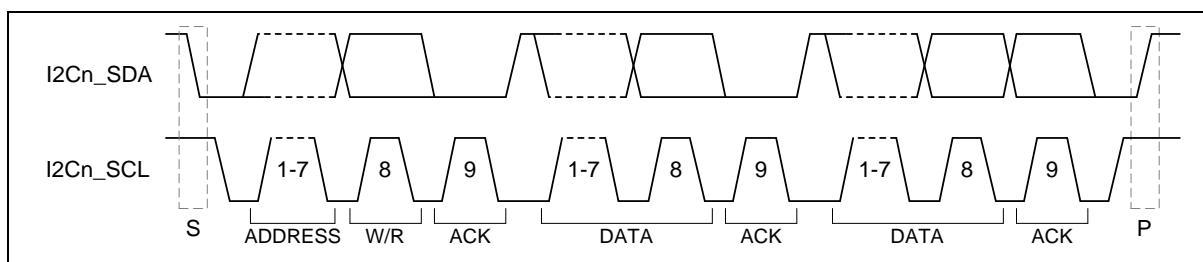


图 6-108 I²C 协议



6.12.5.1.1 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（I₂C_n_SDA 与 I₂C_n_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，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。

6.12.5.1.2 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，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，起始和停止信号的波形。

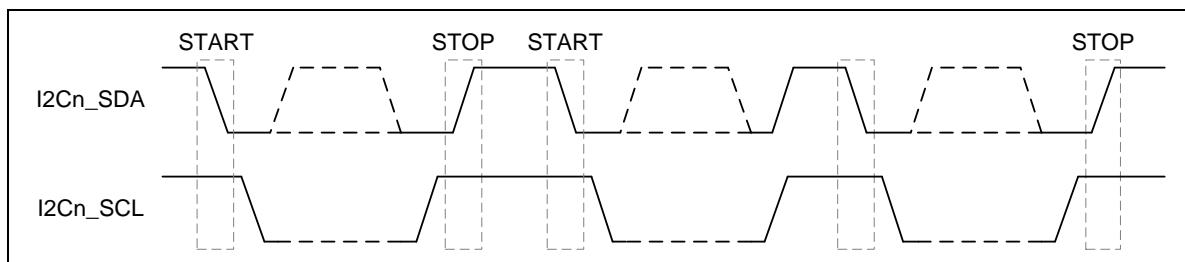


图 6-109 起始 (START) 和停止 (STOP) 信号



6.12.5.1.3 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。

6.12.5.1.4 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。

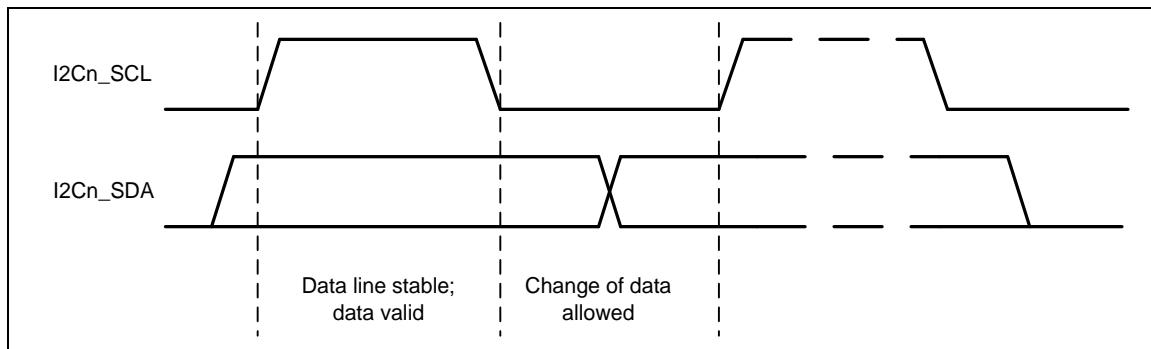


图 6-110 总线上的位传输

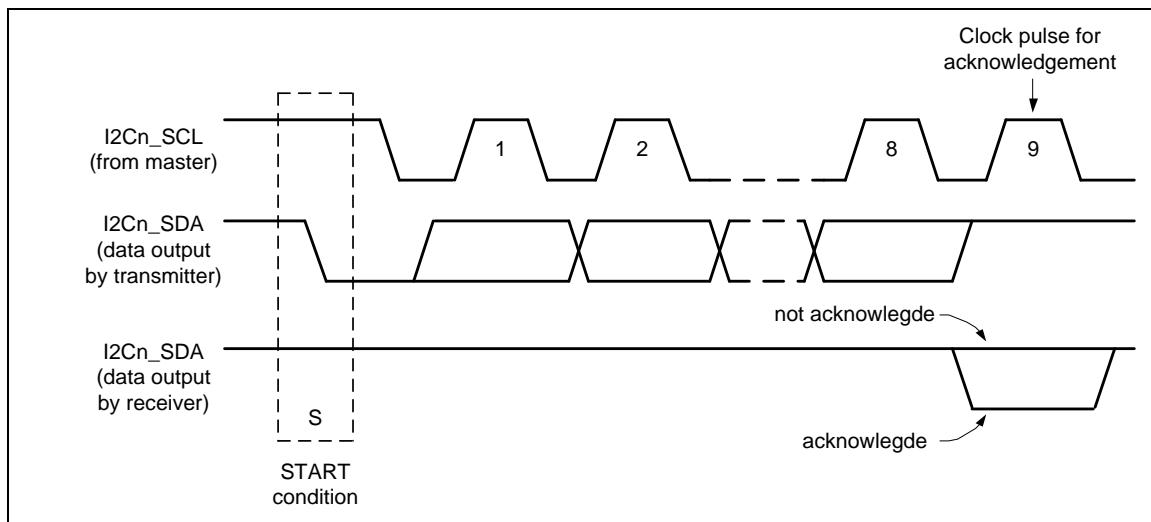


图 6-111 总线上的应答信号

6.12.5.1.5 I²C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位地址和1位写指令,表示主机想要传送数据给从机。从机回应答给主机之后,主机继续传输数据。

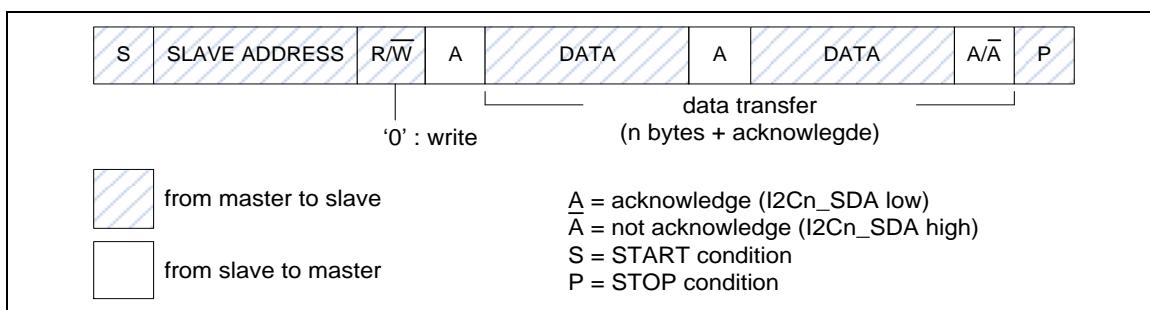


图 6-112 主机向从机传输数据

下图表示主机向从机读取数据。主机发7位地址寻址和1位读指示, 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。

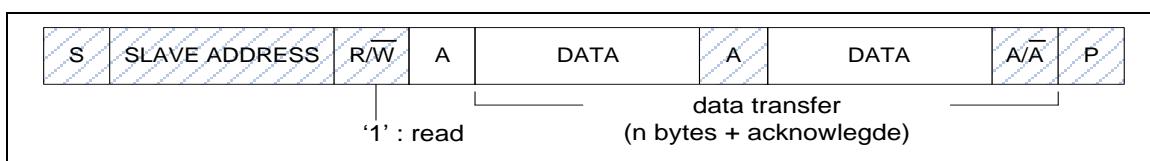


图 6-113 主机向从机读取数据

6.12.5.2 操作模式

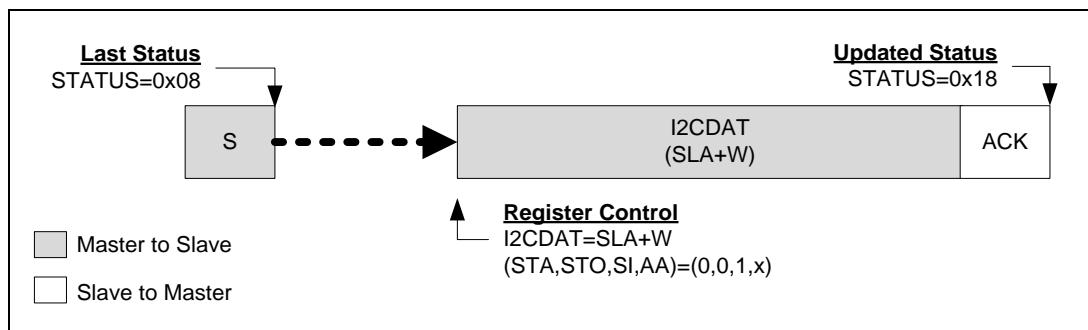
片上I²C端口支持五种操作模式：主机传送，主机接收，从机传送，从机接收和广播呼叫模式。

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 6-114 根据 I²C 当前状态控制总线

6.12.5.2.1 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 6-115）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 6-116）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。

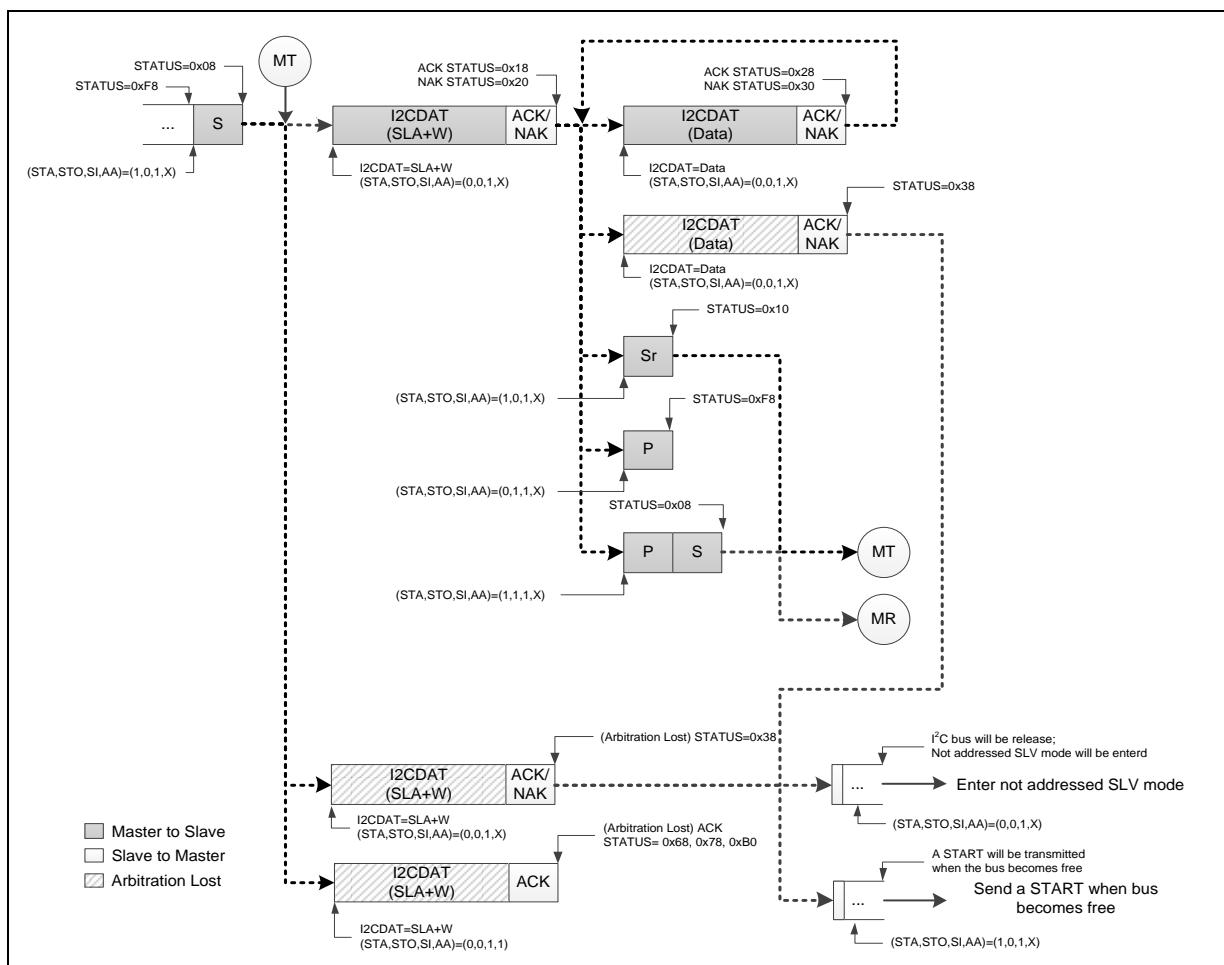


图 6-115 主机传输模式流程控制

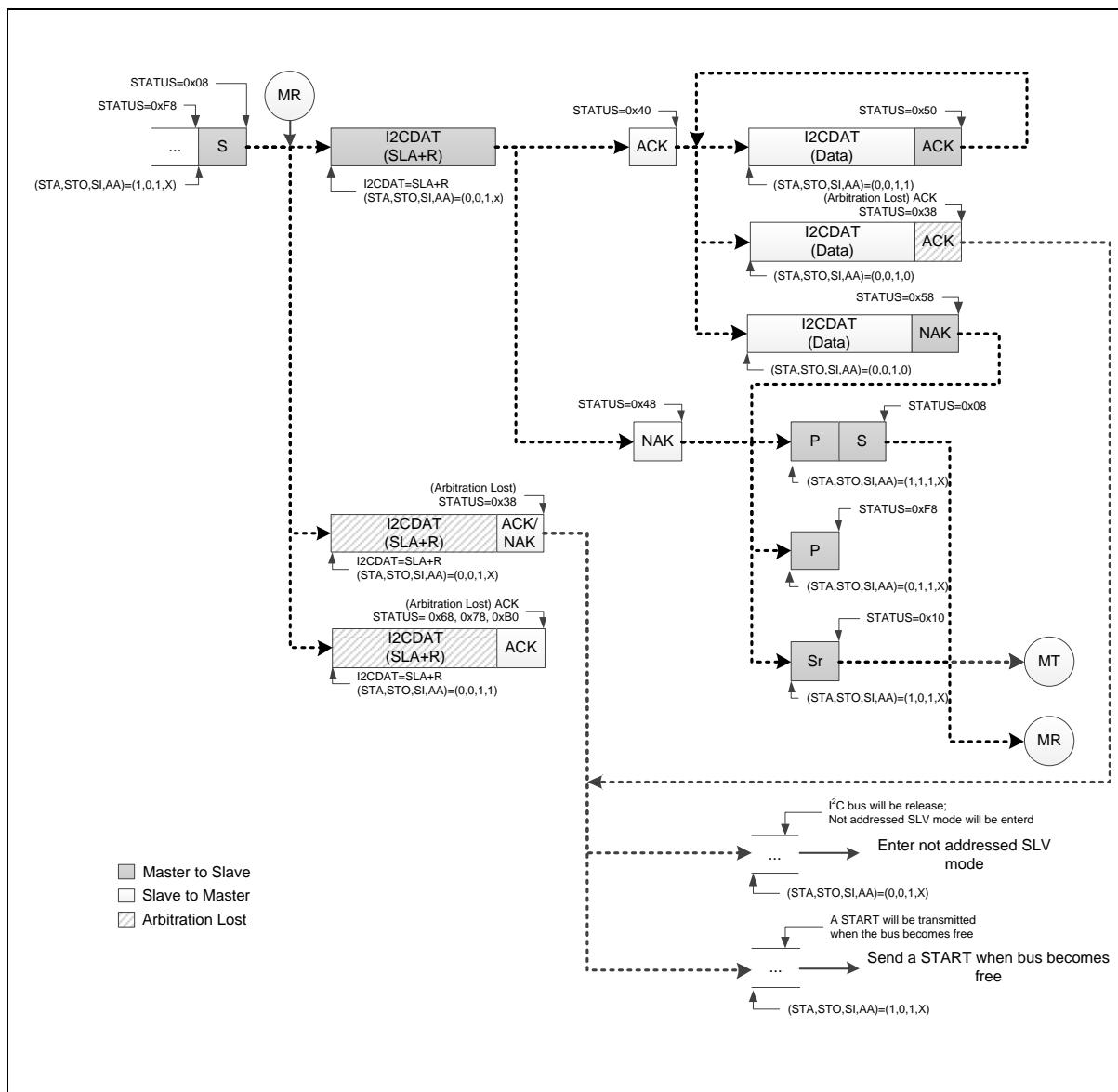


图 6-116 主机接收模式控制流程

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。

6.12.5.2.2 从机模式

复位后默认情况下，I²C不会被寻址，并且不会识别I²C总线上的地址。用户可以通过设置从机地址I2CADDRx 和(STA, STO, SI, AA) = (0, 0, 1, 1)来让I²C识别主机发送的地址。图 6-117所示，从机模式的所有流程。用户需要遵循（图 6-117）的流程来实现他们的I²C协议。

如果在主机模式仲裁丢失，I²C端口立即切换到从机模式并且在同一串行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：I²C通讯期间，当从机模式的SI标志被写‘1’(清除)，I2Cn_SCL clock将释放。

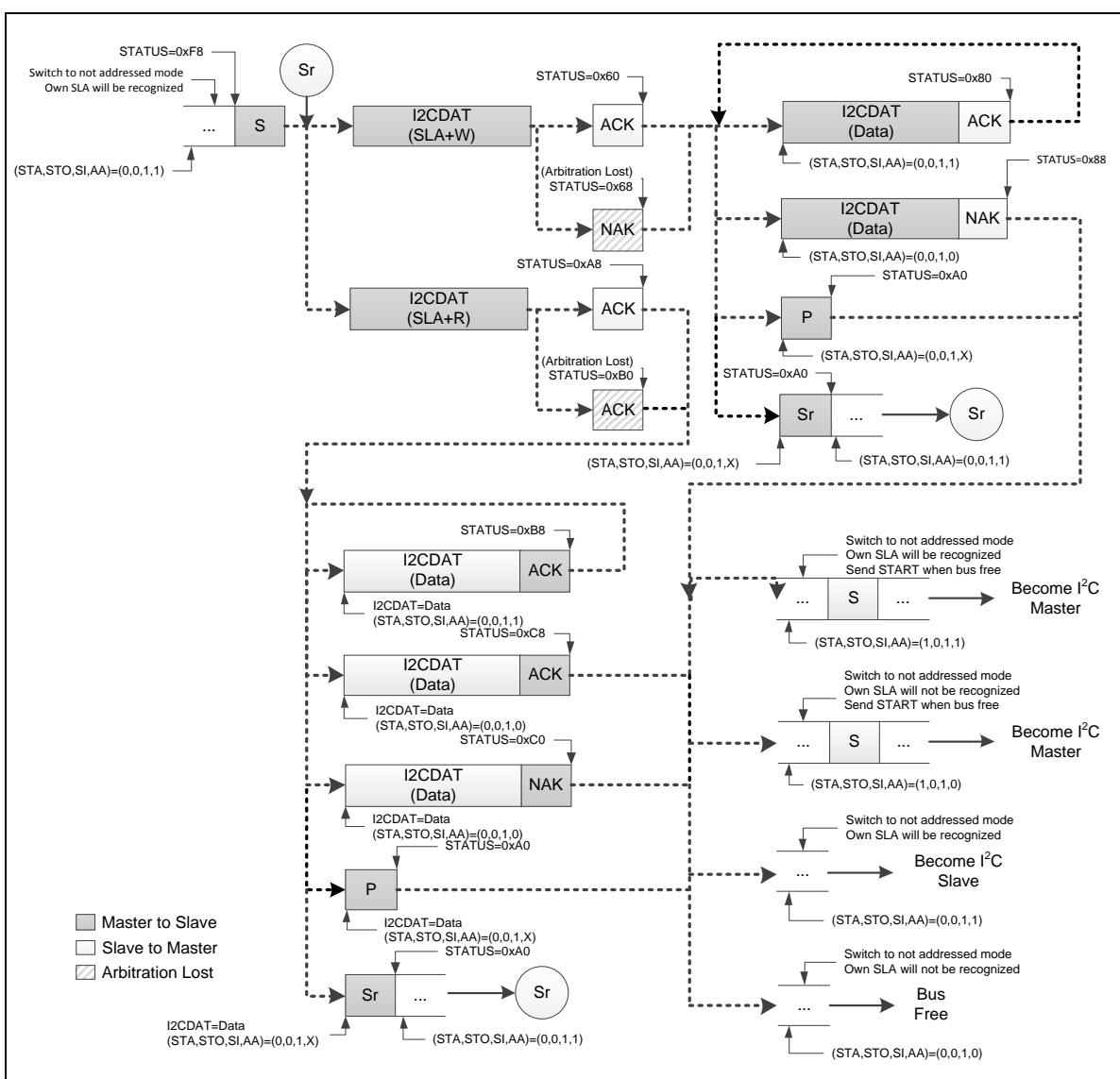


图 6-117 从机模式控制流程

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。

6.12.5.2.3 广播呼叫模式(GC)

如果 GC 位(I²CADDRn [0]) 被设, I²C 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 I²C 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。

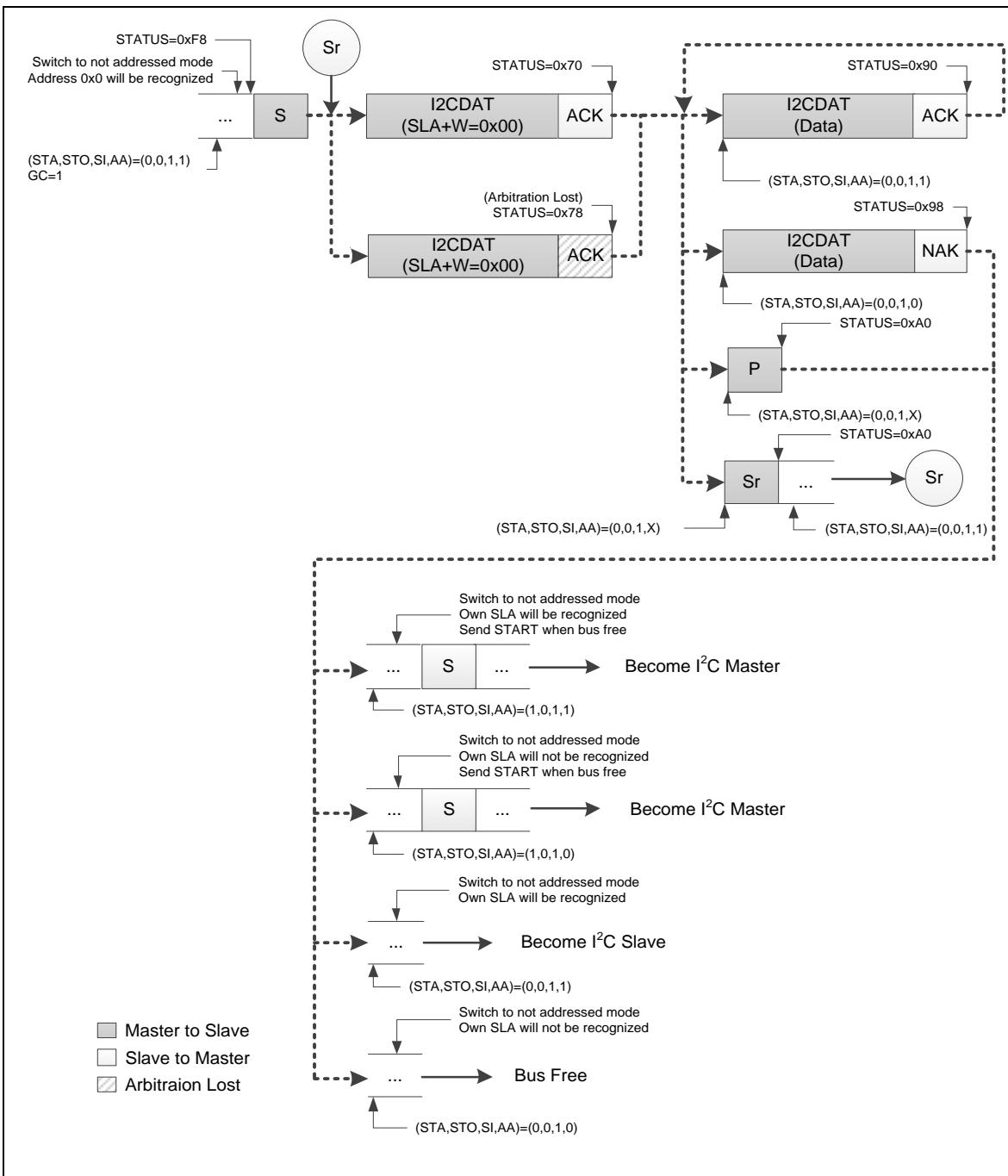


图 6-118 广播呼叫模式 (GC)



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。

6.12.5.2.4 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I2Cn_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。

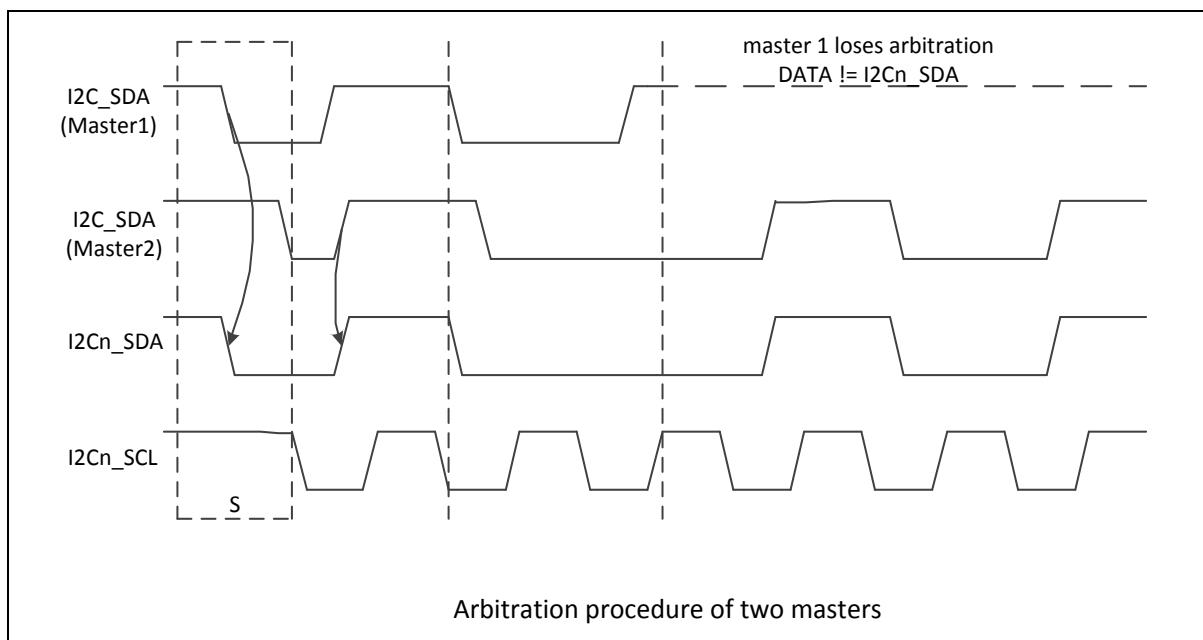


图 6-119 仲裁丢失

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。

6.12.5.3 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)。



6.12.5.3.1 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没意义。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



6.12.5.3.2 从机地址掩码寄存器(I2CADM)

I²C总线控制器带有四个地址掩码寄存器I2CADM_n (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

6.12.5.3.3 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I²C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]

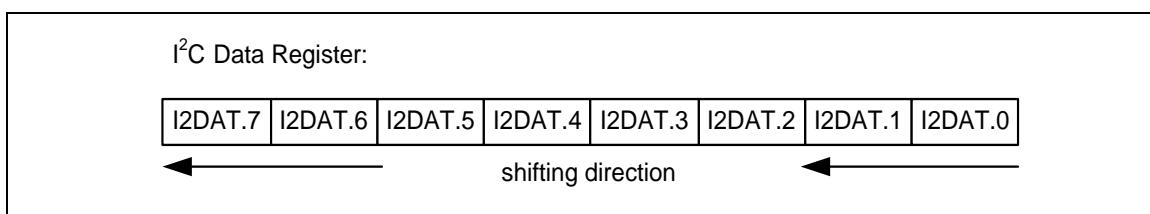


图 6-120 I²C 数据移位方向

6.12.5.3.4 控制寄存器 (I2CON)

CPU 可以直接读或写寄存器 I2CON，当 I²C 端口通过设置 ENS1 (I2CON [6]) 为高使能，内部状态由 I2CON 和 I²C 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，I²C 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，I²C 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。



6.12.5.3.5 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 [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I²C格式帧。总线错误会发生在在一个地址字节、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I²C总线不能识别停止条件。

主机模式		从机模式	
状态码	描述	状态码	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	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 6-30 I²C 状态码描述



6.12.5.3.6 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。

6.12.5.3.7 超时计数器寄存器 (I2CTOC)

MCU 提供一个14位超时计数器来处理当 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 (I2CTOC[0]) = 1 并要求 CPU 产生 I²C 中断或者清除 ENTI (I2CTOC[2]) 为 0 关闭计数功能。当超时计数器使能，对 SI (I2CON[3]) 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_TO[0]) 为0。

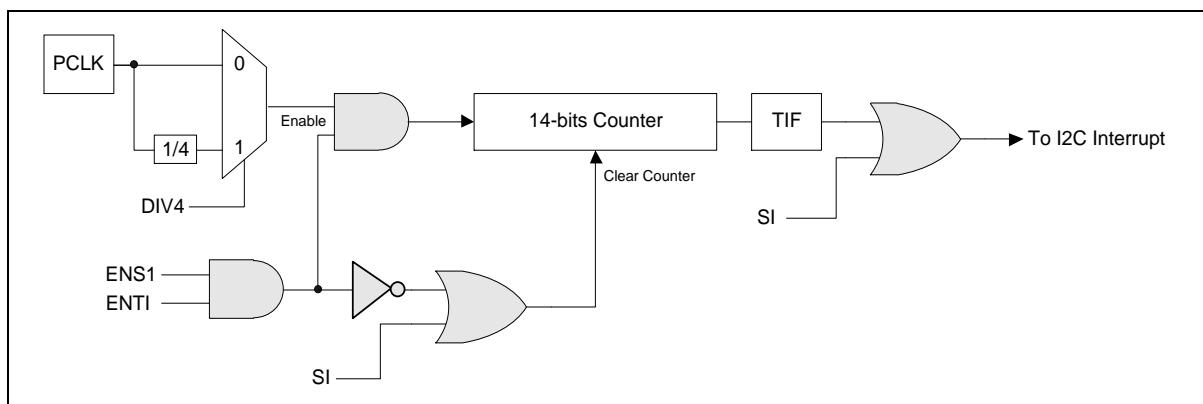


图 6-121 I²C 超时计数器框图



6.12.5.3.8 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中的一个发生地址匹配时唤醒，此时下一个数据将被作废。

6.12.5.3.9 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。

6.12.6 EEPROM随机读取例子

下面是配置I2C0相关寄存器的流程用于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

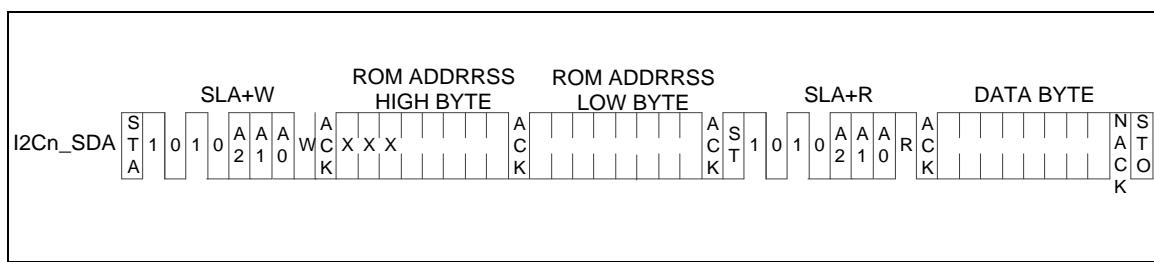


图 6-122 EEPROM 随机读取

下图显示如何使用I²C控制器来实现EEPROM随机读取协议。

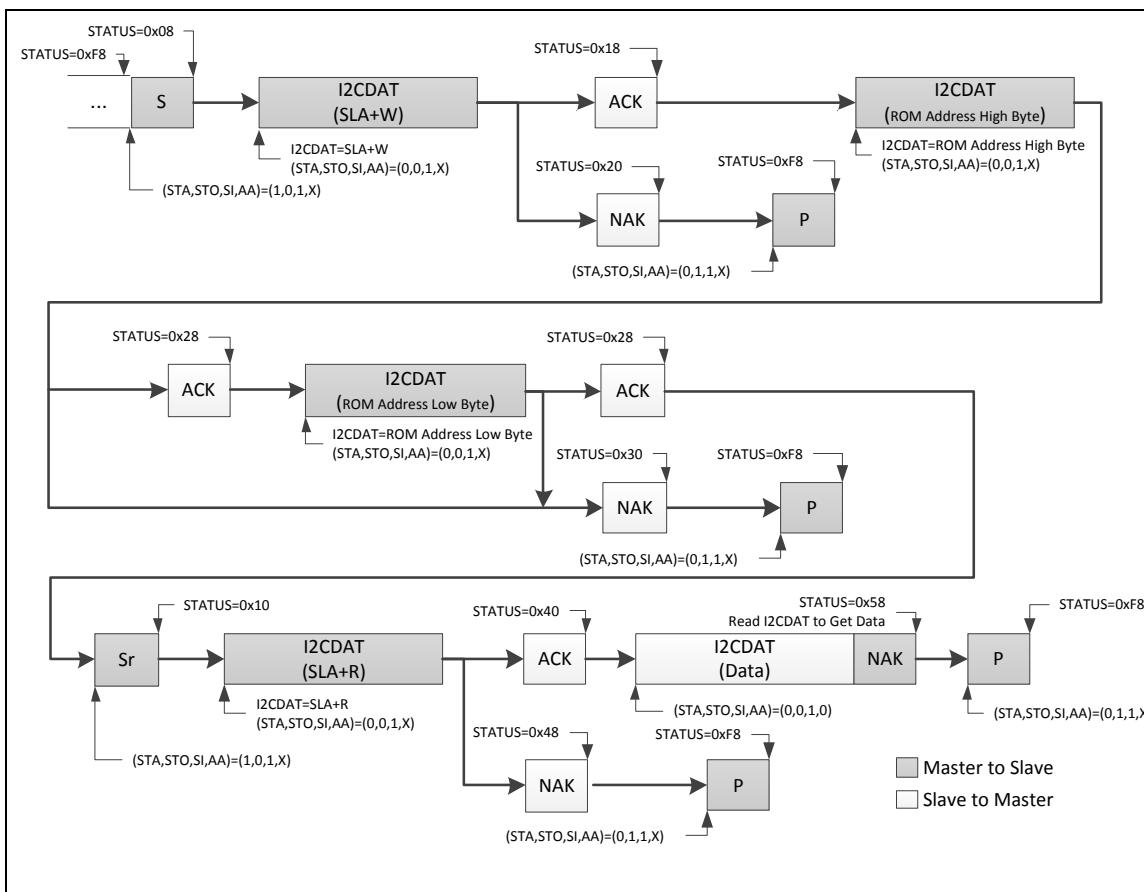


图 6-123 EEPROM 随机读协议

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EEPROM，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



6.12.7 寄存器映射

R: 只读, **W:** 只写, **R/W:** 读/写

寄存器	偏移量	R/W	描述	复位值
I²C 基地址:				
I2C0_BA = 0x4002_0000				
I2C1_BA = 0x4012_0000				
I2CON n=0,1	I2Cn_BA+0x00	R/W	I ² C控制寄存器	0x0000_0000
I2CADDR0 n=0,1	I2Cn_BA+0x04	R/W	I ² C从机地址寄存器0	0x0000_0000
I2CDAT n=0,1	I2Cn_BA+0x08	R/W	I ² C数据寄存器	0x0000_0000
I2CSTATUS n=0,1	I2Cn_BA+0x0C	R	I ² C状态寄存器	0x0000_00F8
I2CLK n=0,1	I2Cn_BA+0x10	R/W	I ² C时钟分频寄存器	0x0000_0000
I2CTOC n=0,1	I2Cn_BA+0x14	R/W	I ² C超时控制寄存器	0x0000_0000
I2CADDR1 n=0,1	I2Cn_BA+0x18	R/W	I ² C从机地址寄存器1	0x0000_0000
I2CADDR2 n=0,1	I2Cn_BA+0x1C	R/W	I ² C从机地址寄存器2	0x0000_0000
I2CADDR3 n=0,1	I2Cn_BA+0x20	R/W	I ² C从机地址寄存器3	0x0000_0000
I2CADM0 n=0,1	I2Cn_BA+0x24	R/W	I ² C从机地址掩码寄存器0	0x0000_0000
I2CADM1 n=0,1	I2Cn_BA+0x28	R/W	I ² C从机地址掩码寄存器1	0x0000_0000
I2CADM2 n=0,1	I2Cn_BA+0x2C	R/W	I ² C从机地址掩码寄存器2	0x0000_0000
I2CADM3 n=0,1	I2Cn_BA+0x30	R/W	I ² C从机地址掩码寄存器3	0x0000_0000
I2CWKUPCON n=0,1	I2Cn_BA+0x3C	R/W	I ² C 唤醒控制寄存器	0x0000_0000
I2CWKUPSTS n=0,1	I2Cn_BA+0x40	R/W	I ² C唤醒状态寄存器	0x0000_0000



6.12.8 寄存器描述

I²C 控制缓存器 (I2CON)

寄存器	偏移量	R/W	描述	复位值
I2CON n=0,1	I2Cn_BA+0x00	R/W	I ² C 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
EI	ENS1	STA	STO	SI	AA	Reserved	

位	描述	
[31:8]	Reserved	保留
[7]	EI	使能中断 1 = 使能 I ² C 中断 0 = 禁用 I ² C 中断
[6]	ENS1	I²C 控制器使能位 1 = 使能 0 = 禁用 设置使能 I ² C 串行控制器功能。当 ENS1=1, I ² C 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 I ² C 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, I ² C 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 I ² C 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 I ² C 从机模式, 设置 STO 复位 I ² C 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 I ² C 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 I ² C 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	偏移量	R/W	描述	复位值
I2CDAT n=0,1	I2Cn_BA+0x08	R/W	I ² C 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							

位	描述	
[31:8]	Reserved	保留
[7:0]	I2CDAT	I²C 数据寄存器 该区域 8 位 I ² C 串行端口传输数据

I²C 状态寄存器 (I2CSTATUS)

寄存器	偏移量	R/W	描述	复位值
I2CSTATUS n=0,1	I2Cn_BA+0x0C	R	I ² C 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							

位	描述	
[31:8]	Reserved	保留
[7:0]	I2CSTATUS	<p>I²C 状态寄存器</p> <p>共有26个可能状态码。</p> <p>当I2CSTATUS值是0xF8没有串行中断请求。</p> <p>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。</p> <p>此外，0x00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。不正确的位比如是在串行传输地址字节、数据字节或应答位期间。</p>



I²C 时钟分频寄存器 (I2CLK)

寄存器	偏移量	R/W	描述	复位值
I2CLK n=0,1	I2Cn_BA+0x10	R/W	I ² C 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							

位	描述	
[31:8]	Reserved	保留
[7:0]	I2CLK	<p>I²C 时钟分频寄存器</p> <p>I²C 数据波特率 = (系统时钟) / (4x (I2CLK+1)).</p> <p>注意: I2CLK 的最小值是 4.</p>



I²C 超时计数器寄存器 (I2CTOC)

寄存器	偏移量	R/W	描述	复位值
I2CTOC n=0,1	I2Cn_BA+0x14	R/W	I ² C 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	<p>超时计数器使能/禁用 1 = 使能 0 = 禁用</p> <p>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。</p>
[1]	DIV4	<p>超时定时器输入时钟除以 4 1 = 使能 0 = 禁用</p> <p>该位使能后，超时时间扩大 4 倍。</p>
[0]	TIF	<p>超时溢出标志 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。</p> <p>注意：写 1 清除该位。</p>



I²C 从机地址寄存器 (I2CADDRx)

寄存器	偏移量	R/W	描述	复位值
I2CADDR0 n=0,1	I2Cn_BA+0x04	R/W	I ² C 从机地址寄存器0	0x0000_0000
I2CADDR1 n=0,1	I2Cn_BA+0x18	R/W	I ² C 从机地址寄存器1	0x0000_0000
I2CADDR2 n=0,1	I2Cn_BA+0x1C	R/W	I ² C 从机地址寄存器2	0x0000_0000
I2CADDR3 n=0,1	I2Cn_BA+0x20	R/W	I ² C 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							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，I ² C 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	偏移量	R/W	描述	复位值
I2CADM0 n=0,1	I2Cn_BA+0x24	R/W	I ² C 从机地址掩码寄存器0	0x0000_0000
I2CADM1 n=0,1	I2Cn_BA+0x28	R/W	I ² C 从机地址掩码寄存器1	0x0000_0000
I2CADM2 n=0,1	I2Cn_BA+0x2C	R/W	I ² C 从机地址掩码寄存器2	0x0000_0000
I2CADM3 n=0,1	I2Cn_BA+0x30	R/W	I ² C 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							Reserved

位	描述	
[31:8]	Reserved	保留
[7:1]	I2CADM	<p>I²C 地址掩码寄存器</p> <p>1 = 使能掩码（接收到的相应地址位不予辨识）</p> <p>0 = 禁用掩码（接收到的相应地址必须完全符合地址寄存器）</p> <p>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。</p>
[0]	Reserved	保留



I²C 唤醒控制寄存器(I2CWKUPCON)

寄存器	偏移量	R/W	描述	复位值
I2CWKUPCON n=0,1	I2Cn_BA+0x3C	R/W	I ² C 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	I ² C唤醒功能使能 1 = I ² C 唤醒功能使能. 0 = I ² C 唤醒功能禁止.



I²C 唤醒状态寄存器 (I2CWKUPSTS)

寄存器	偏移量	R/W	描述	复位值
I2CWKUPSTS n=0,1	I2Cn_BA+0x40	R/W	I ² C 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 I ² C唤醒标志 0 =芯片不是在掉电模式下由I ² C中断唤醒 1 =芯片是在掉电模式下由I ² C中断唤醒 注: 该位软件写1清零



6.13 串行外围设备接口 (SPI)

6.13.1 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。NuMicro™ NUC131 系列带一套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。该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，以适用于一些特殊的应用

6.13.2 特性

- 一组 SPI 控制器
- 支持主机和从机工作模式
- 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
- 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
- 支持三线，没有从机选择信号的双向接口

6.13.3 框图

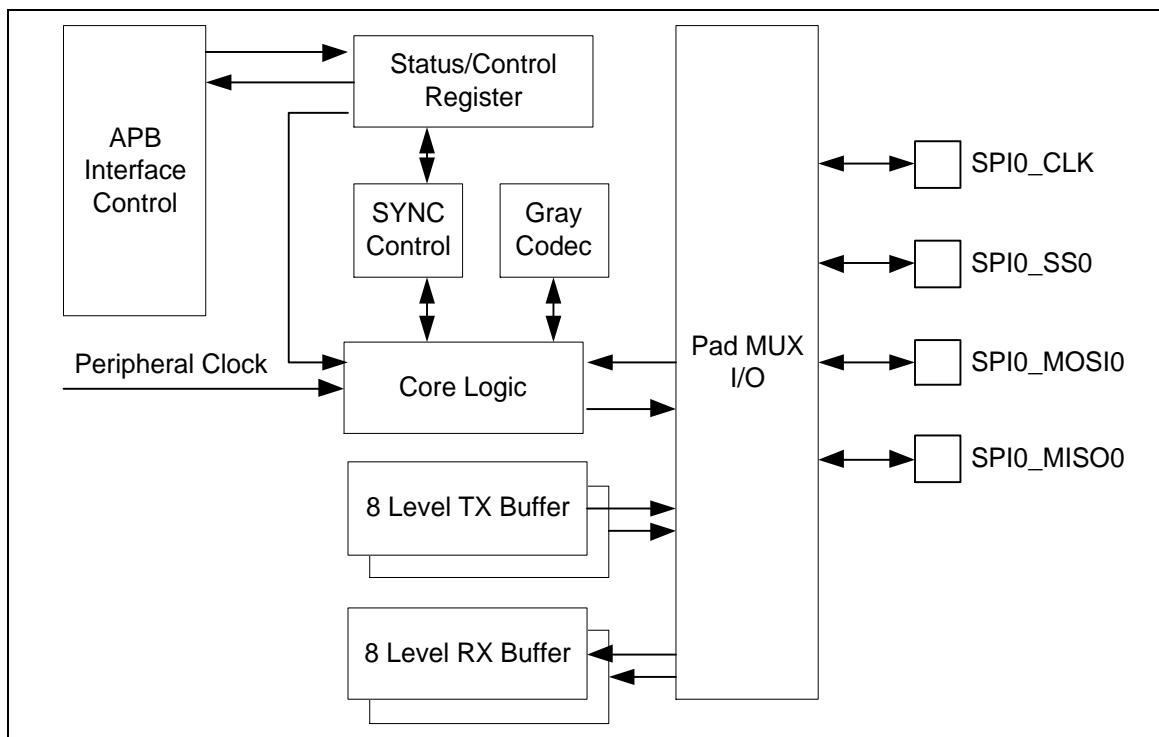


图 6-124 SPI 框图

6.13.4 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 and GPC_MFP 配置.
- 通过寄存器CLKSEL1的bit4 SPI0_S 选择SPI0 的时钟源.
- 通过寄存器APBCLK 的bit12 SPI0_EN使能SPI0的时钟.
- 通过寄存器IPRSC2的bit12位SPI0_RST复位SPI0控制器.

6.13.5 功能描述

6.13.5.1 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总线时钟就是SPI0_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SPI0_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_CTRL2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SPI_DIVIDER[7:0]位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。通常情况下，SPI总线时钟表示为SPI时钟。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_CTRL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：

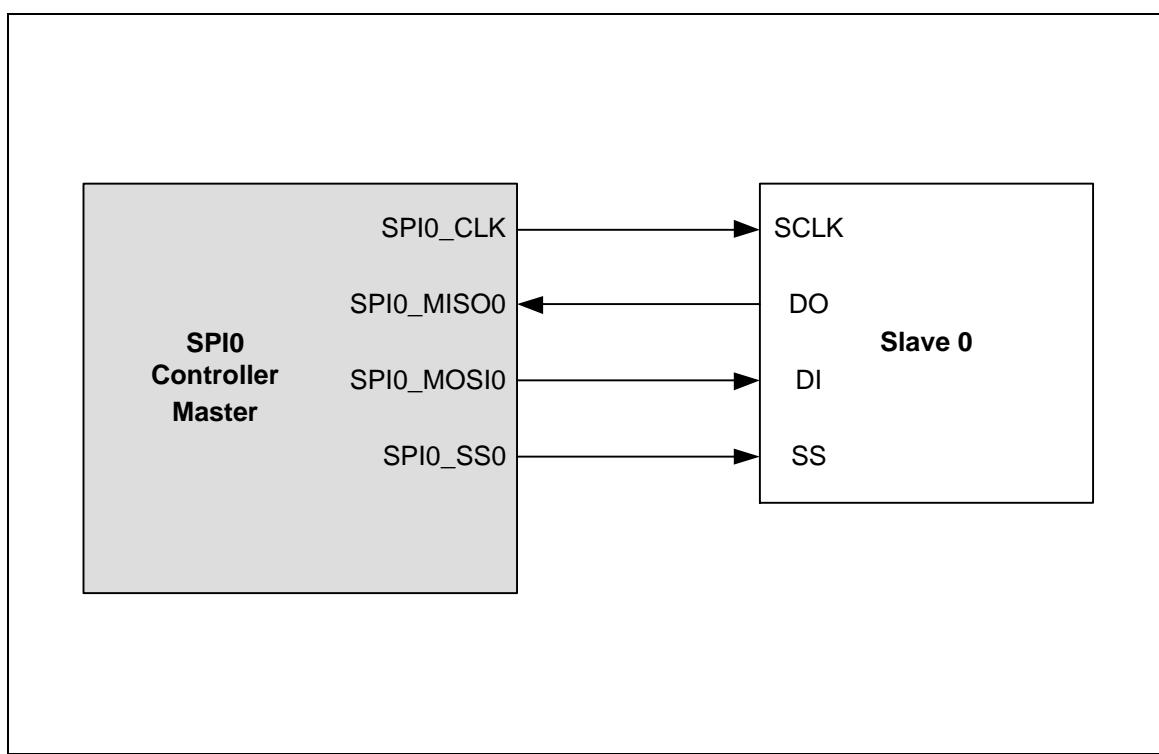


图 6-125 SPI 主机模式应用框图

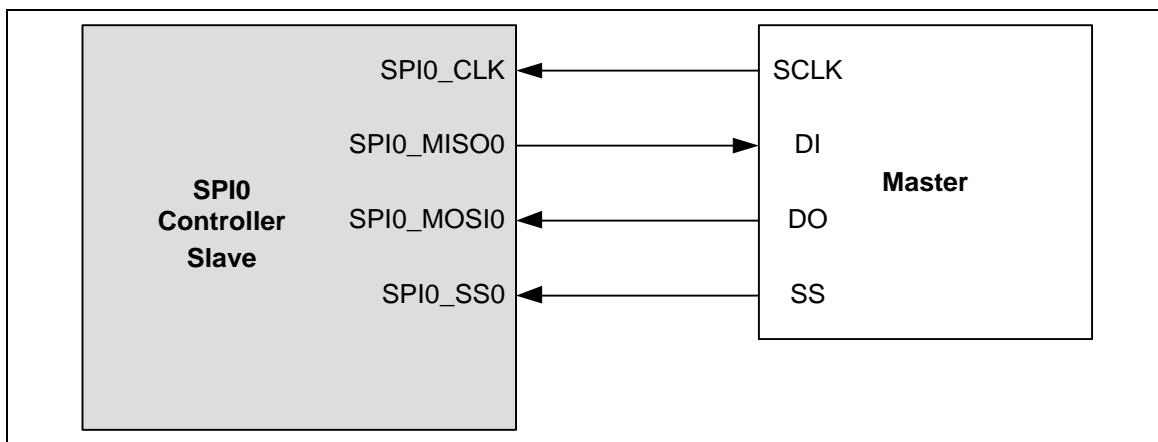


图 6-126 SPI 从机模式应用框图

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，SPI0_CLK输出为空闲高电平状态，否则如果CLKP=0，SPI0_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_CCTRL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。

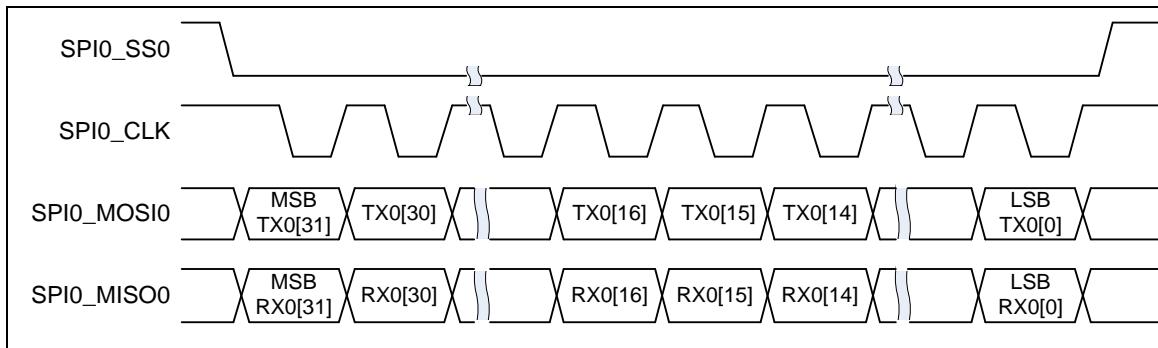


图 6-127 一次传输32位长度 (主模式下)

LSB/MSB优先

寄存器SPI_CCTRL的第10位LSB定义了一个事务中位的传输序列。如果设定 SPI_CCTRL 寄存器的 bit10位LSB位为1，传输序列为LSB优先。否则如果LSB位为0，传输序列是MSB优先。

发送边沿

寄存器SPI_CCTRL的第2位TX_NEG定义了数据在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定义了数据在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SPI0_SS0来驱动片外从机设备。在从机模式，片外主机设备通过SPI0_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SPI0_SS0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(SPI_CNTRL[7:3])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(SPI_SSR[5])状态位来检查数据是否已经全部传完。

6.13.5.2 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]是否使能，将从机选择信号输出到SPI0_SS0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(SPI_SSR[3])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，在两个连续的事务

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。

6.13.5.3 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(SPI_CNTRL[7:3])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 (SPI_DIVIDER[7:0])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(SPI_DIVIDER[23:16])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(SPI_DIVIDER[7:0])的设置；如果值是11，时钟周期取决于寄存器DIVIDER2(SPI_DIVIDER[23:16])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(SPI_DIVIDER[7:0])设置和寄存器DIVIDER2(SPI_DIVIDER[23:16])设置之间的时序关系如下图所示。

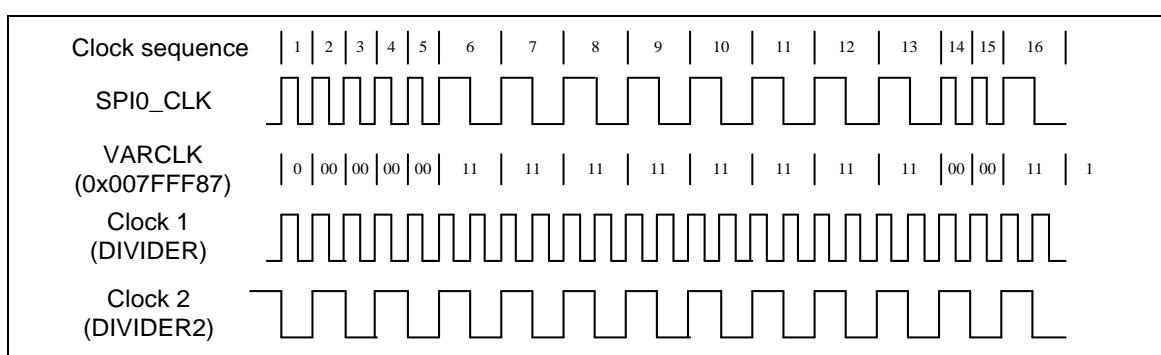


图 6-128 可变总线时钟频率

6.13.5.4 字节重排序功能

当设置为 MSB 优先($\text{SPI_CNTRL}[10] = 0$)且使能 REORDER($\text{SPI_CNTRL}[19]$)，在 TX_BIT_LEN = 0，32位模式时($\text{TX_BIT_LEN}(\text{SPI_CNTRL}[7:3]) = 0$)，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($\text{SPI_CNTRL}[7:3]$)被置位设为 24位模式，存储在 TX 缓存与 RX 缓存的数据将重新按 [未知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 $\text{TX_BIT_LEN}(\text{SPI_CNTRL}[7:3])$ 为 16, 24, 和 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。

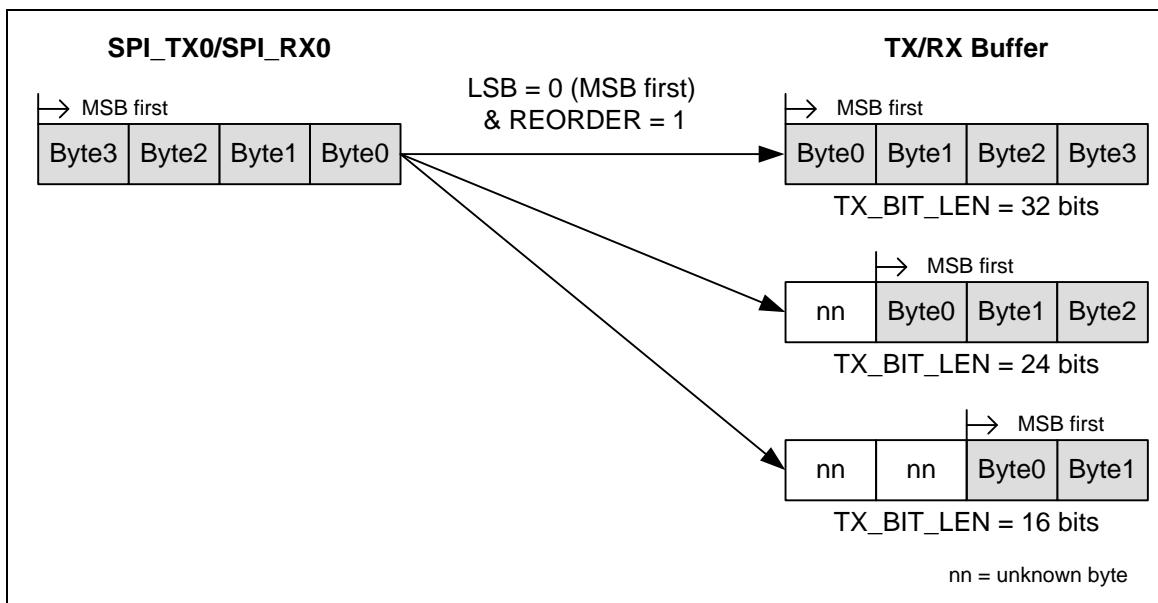


图 6-129 字节重排序功能

6.13.5.5 字节休眠功能

主机模式下，如果 SPI_CNTRL[19] (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(SPI_CNTRL[15:12])来设置。

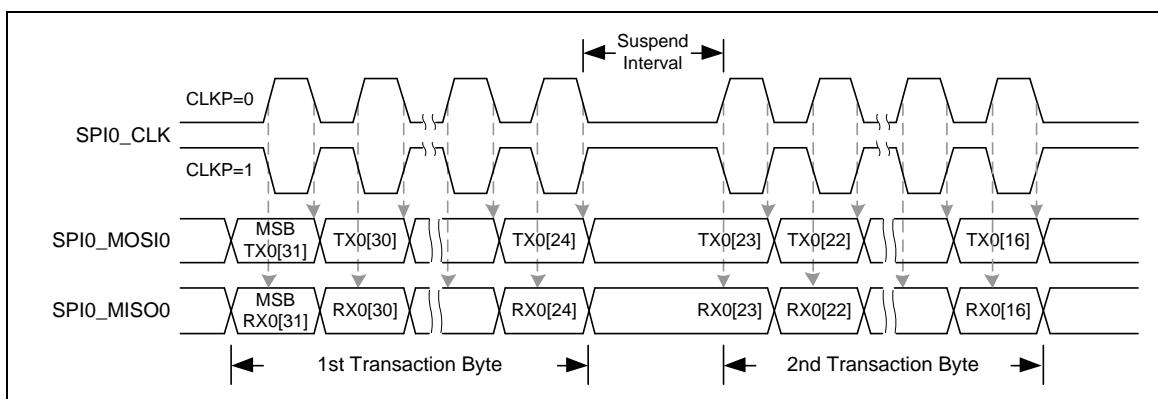


图 6-130 字节休眠时序波形(主模式)

6.13.5.6 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控制器可以在没有从机选择信号下工作。NOSLVSEL(SPI_CNTRL2[8])位仅在从机模式下有效。在与一个主机通讯时，仅需要三个管脚，SPI0_CLK, SPI0_MISO0 和 SPI0_MOSI0。SPI0_SS脚可以配置成一个GPIO。当NOSLVSEL(SPI_CNTRL2[8])位被设为 1时，在 GO_BUSY(SPI_CNTRL[0])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TX_BIT_LEN(SPI_CNTRL[7:3])长度定义时，单元传输中断标志IF(SPI_CNTRL[16])将会置位。

注意:在从机3-线模式下，SS_LTRIG, SPI_SSR[4]，必须设为 1。

6.13.5.7 双 I/O 模式

当设置DUAL_IO_EN bit (SPI_CNTRL2[13]) 为1时，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 (SPI_CNTRL2[12])位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 位被清0，控制器会从外部设备读数据。该功能支持8, 16, 24, 和 32位长度传输。

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(SPI_CNTRL2[13]) 和 DUAL_IO_DIR(SPI_CNTRL2[12])都设置成1， SPI0_MOSI0 是偶数位数据输出， SPI0_MISO0是奇数位数据输出。如果DUAL_IO_EN(SPI_CNTRL2[13])置1， DUAL_IO_DIR(SPI_CNTRL2[12])清0， SPI0_MISO0和 SPI0_MOSI0都会设置为数据输入口。

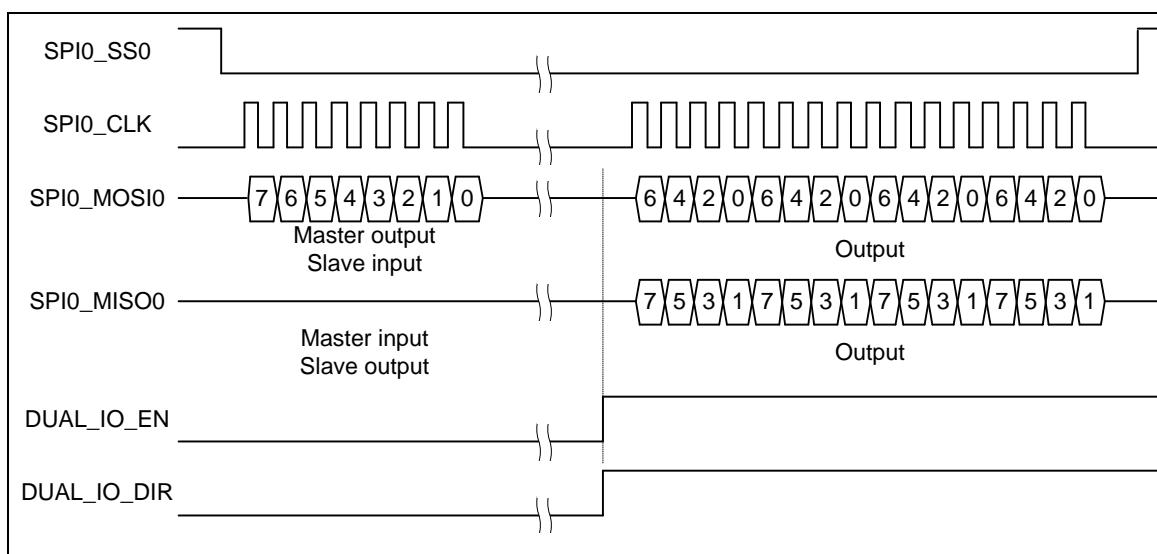


图 6-131 双输出模式的位序列

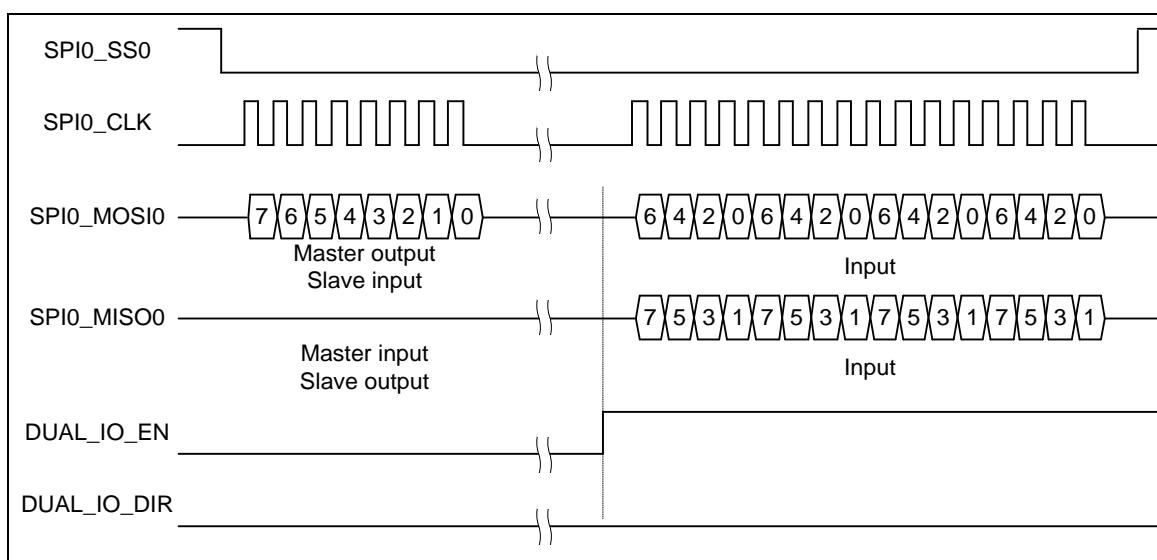


图 6-132 双输入模式的位序列

6.13.5.8 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8个32位发送和接收FIFO缓存。

发送FIFO缓存是一个8层深度，32位宽，先进先出的寄存器缓存。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(SPI_STATUS[27])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，那么8层发送FIFO缓存为空，TX_EMPTY(SPI_STATUS[26])位被置1。注意最后一笔传输还在进行时，TX_EMPTY(SPI_STATUS[26])标志已被置1。主机模式下，软件应同时检查GO_BUSY(SPI_CNTRL[0])位和TX_EMPTY(SPI_STATUS[26])位确认SPI是否为空闲状态。

接收FIFO缓存也是一个8层深度，32位宽，先进先出的寄存器缓存。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。FIFO还带有一些相关状态位，像RX_EMPTY(SPI_STATUS[24]) 和 RX_FULL RX_FULL (SPI_STATUS[25])，用来表明当前FIFO缓存的状态。

FIFO模式下，发送和接收阀值可以通过软件设置TX_THRESHOLD(SPI_FIFO_CTL[30:28])和RX_THRESHOLD(SPI_FIFO_CTL[26:24])来设定。当存储在发送FIFO缓存的有效数据计数小于或等于TX_THRESHOLD设定，TX_INTSTS(SPI_STATUS[4])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(SPI_STATUS[0])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 寄存器的内容不会被软件修改，除非 FIFO位被清0使FIFO模式禁止。

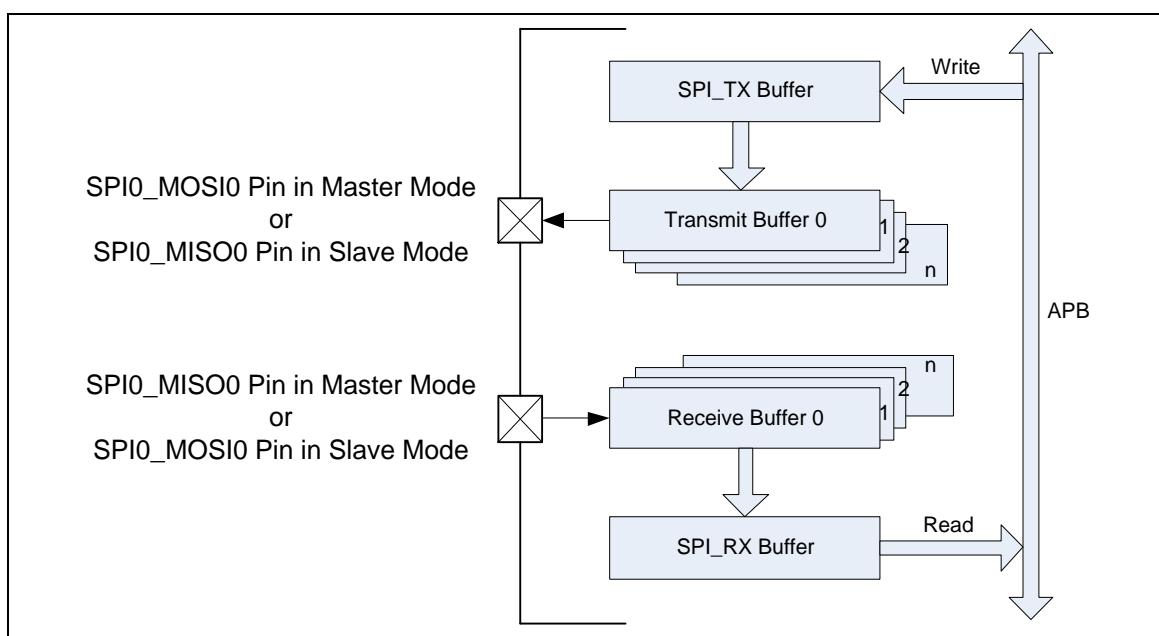


图 6-133 FIFO 模式框图

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 (SPI_STATUS[26])标志将会被清0。只要发送 FIFO 缓存非空，发送操作立即开始。用户立即可以写下

一个要发送的数据。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 (SPI_STATUS[27]) 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 SPI0_MISO0 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(SPI_STATUS[24]) 将会被清除为 0。只要 RX_EMPTY(SPI_STATUS[24]) 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(SPI_STATUS[25]) 标志将被设置为 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。

从机模式下，当 FIFO 位被设置为 1，GO_BUSY 位将被硬件自动设置为 1。

在从机发送操作中，当软件写数据到 SPI_TX0 寄存器，数据将被加载到发送 FIFO 缓存，且 TX_EMPTY(SPI_STATUS[26]) 标志将被清 0。当从设备从主机接收到时钟信号，发送操作将开始。只要 TX_FULL(SPI_STATUS[27]) 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 (SPI_STATUS[26]) 标志将被设置为 1。

在从机接收操作中，串行数据从 SPI0_MOSI0 管脚被接收，并被存储到接收 FIFO 缓存寄存器。接收机制与主机模式接收操作类似。

6.13.5.9 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 (SPI_CNTRL2[10]) 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(SPI_CNTRL2[9]) 位来中止当前传输。如果软件设置 SLV_ABORT(SPI_CNTRL2[9]) 位为 1，单元传输中断标志 IF 将会被置位。

- 接收 FIFO 超时中断

FIFO 模式下，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

FIFO 模式下，如果发送 FIFO 缓存的有效数据计数小于或等于 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

FIFO 模式下，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。

6.13.6 时序图

从机选择信号的有效状态可以由 SS_LVL (SPI_SSR[2]) 位和 SS_LTRIG (SPI_SSR[4]) 位来定义。串行时钟 (SPI0_CLK) 的空闲状态可以通过 CLKP 位 (SPI_CCTRL[11]) 配置为高电平或低电平。传输字段长度在 TX_BIT_LEN (SPI_CCTRL[7:3]) 中定义，发送/接收数据是以 MSB 或 LSB 优先由 LSB 位 (SPI_CCTRL[10]) 定义。用户可以通过设置 TX_NEG/RX_NEG (SPI_CCTRL[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。

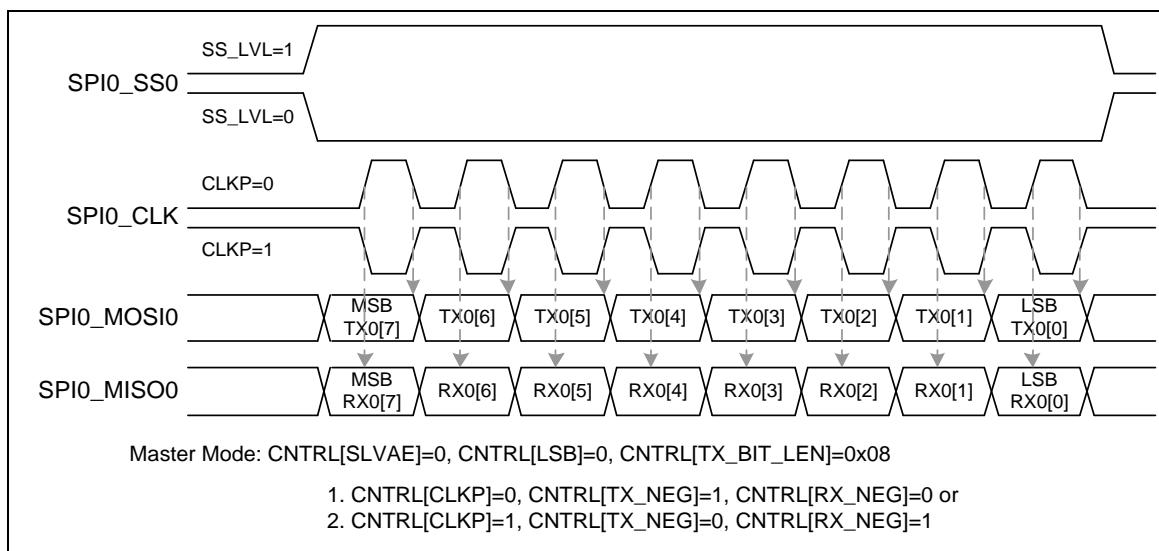


图 6-134 SPI 主机模式下的时序

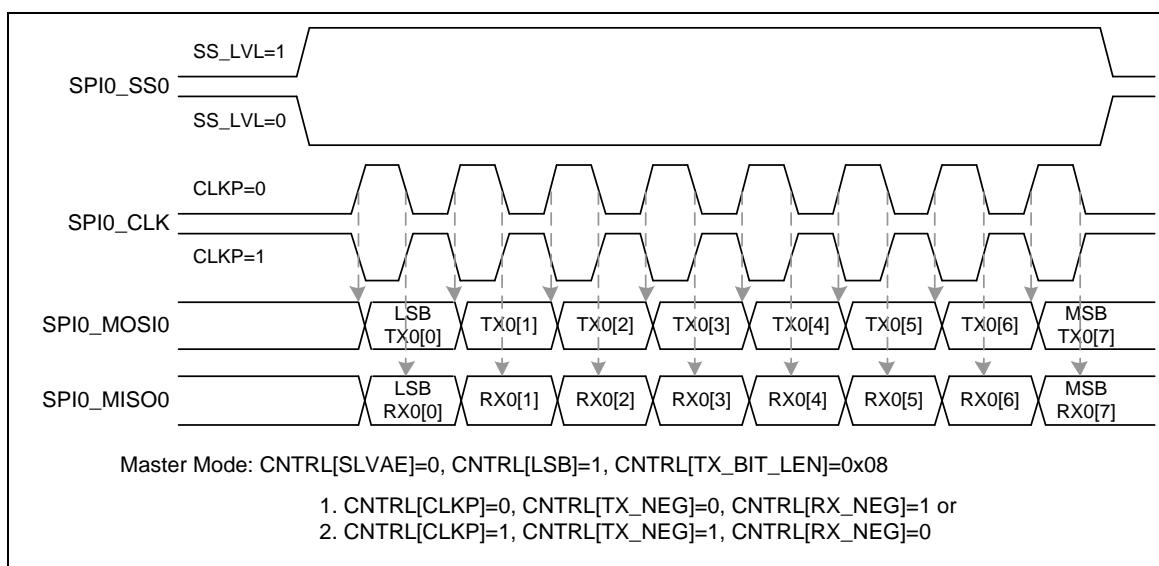


图 6-135 SPI 主机模式下的时序(交替SPI时钟相位)

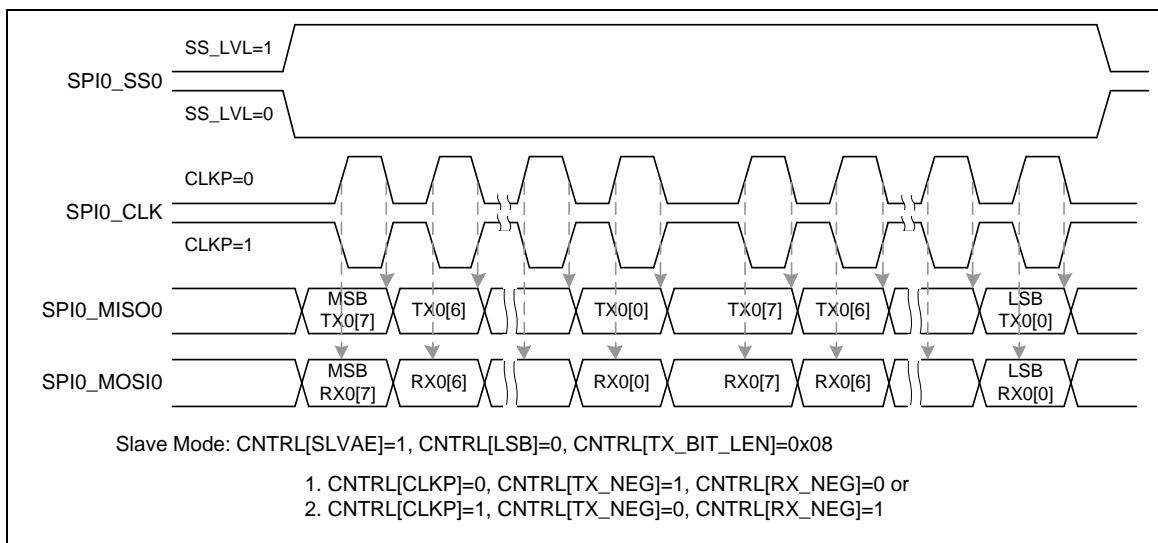


图 6-136 SPI 从机模式下的时序

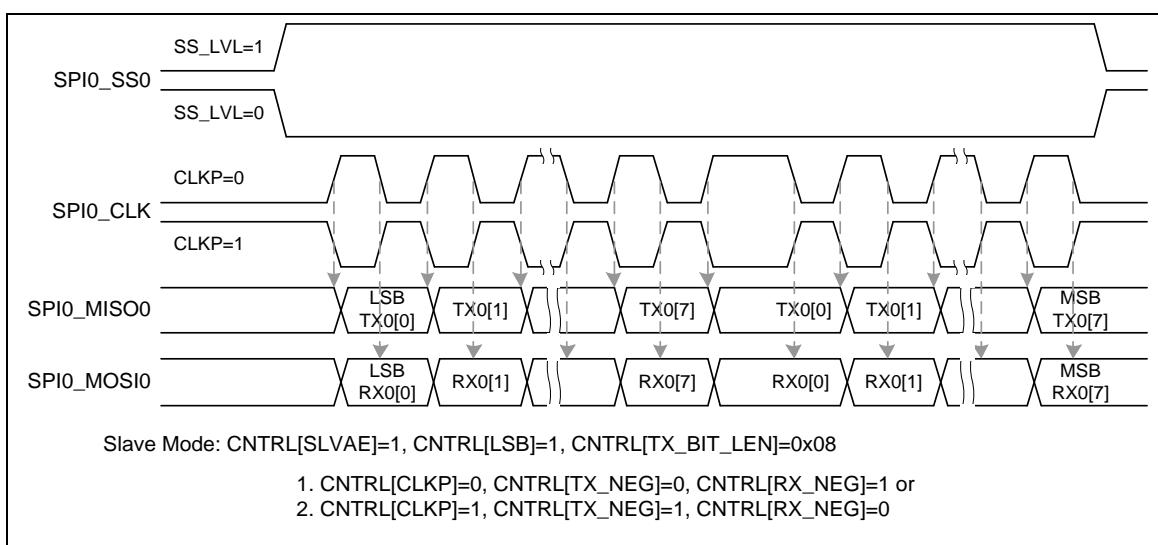


图 6-137 SPI 从机模式下的时序(交替SPI时钟相位)



6.13.7 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。
- SPI0_CLK 空闲模式为低电平状态
- 每次只发送/接收一个字节
- 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]) 和从机选择电平触发位 SS_LTRIG (SPI_SSR[4]) 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 (SPI_CNTRL[18] = 0) 位将 SPI 控制器设为主机设备。
 2. 通过设置 CLKP (SPI_CNTRL[11] = 0) 位将串行时钟的空闲状态设为低电平。
 3. 通过设置 TX_NEG (SPI_CNTRL[2] = 1) 位选择数据在串行时钟的下降沿传输。.
 4. 通过设置 RX_NEG (SPI_CNTRL[1] = 0) 位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 被使能）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。外设主设备通过SPI 接口与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 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片外主机，则将所要发送的数据写入到 SPI_TX0 寄存器。
- 4) 如果 SPI 从机只是要从片外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 被使能）或轮询检测 GO_BUSY 位直到被硬件自动清零。
- 7) 从寄存器 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。



6.13.8 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				
SPI_CNTRL	SPI0_BA+0x00	R/W	控制和状态寄存器	0x0500_3004
SPI_DIVIDER	SPI0_BA+0x04	R/W	时钟分频寄存器	0x0000_0000
SPI_SSR	SPI0_BA+0x08	R/W	从机选择寄存器	0x0000_0000
SPI_RX0	SPI0_BA+0x10	R	数据接收寄存器0	0x0000_0000
SPI_RX1	SPI0_BA+0x14	R	数据接收寄存器1	0x0000_0000
SPI_TX0	SPI0_BA+0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1	SPI0_BA+0x24	W	数据发送寄存器1	0x0000_0000
SPI_VARCLK	SPI0_BA+0x34	R/W	可调时钟类型寄存器	0x007F_FF87
SPI_CNTRL2	SPI0_BA+0x3C	R/W	控制和状态寄存器 2	0x0000_1000
SPI_FIFO_CTL	SPI0_BA+0x40	R/W	SPI FIFO 控制寄存器	0x4400_0000
SPI_STATUS	SPI0_BA+0x44	R/W	SPI 状态寄存器	0x0500_0000



6.13.9 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	SPI0_BA+0x00	R/W	控制与状态寄存器				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	Reserved	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]	Reserved	保留.
[21]	FIFO	<p>FIFO 模式使能位 1 = 使能 FIFO 模式 0 = 禁用 FIFO 模式</p> <p>注意: 在使能 FIFO 模式前, 其他相关的设定必须事先设定.</p> <ol style="list-style-type: none"> 在主机模式, 如果 FIFO 模式被使能, 在数据被写入发送 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。 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	<p>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.</p> <p>注意: 字节重排序仅在 TX_BIT_LEN 被定义为 16 位, 24 位和 32 位时有效. 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。 1. 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。</p>
[18]	SLAVE	<p>从机模式使能位 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.</p>
[17]	IE	<p>中断使能位 1 = 使能 SPI 中断 0 = 禁用 SPI 中断</p>
[16]	IF	<p>中断标志位 1 = 表示传输完成 0 = 表示没有传输还完成</p> <p>注意: 该位写 1 清零。</p>
[15:12]	SP_CYCLE	<p>休眠间隔(仅主模式) 该四位提供一个在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。休眠间隔是指前一个事务的最后一个时钟边缘和后一个事务的第一个时钟边缘之间的间隔。默认值是 0x3. 休眠间隔长度根据如下公式获得: $(SP_CYCLE[3:0] + 0.5) * \text{SPICLK 时钟周期}$</p> <p>例: $SP_CYCLE = 0x0 \dots 0.5 \text{ SPICLK 时钟周期}$ $SP_CYCLE = 0x1 \dots 1.5 \text{ SPICLK 时钟周期}$</p>

		<p>.....</p> <p>SP_CYCLE = 0xE ... 14.5 SPICLK 时钟周期</p> <p>SP_CYCLE = 0xF ... 15.5 SPICLK 时钟周期</p> <p>如果可变时钟功能被使能，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 $(6.5 + SP_CYCLE) * SPICLK clock cycle$</p>
[11]	CLKP	<p>时钟极性</p> <p>1 = SPICLK 空闲状态的电平默认为高</p> <p>0 = SPICLK 空闲状态的电平默认为低.</p>
[10]	LSB	<p>LSB 优先</p> <p>1 = LSB, SPI_RX0/1 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.</p> <p>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。</p>
[9:8]	Reserved	保留.
[7:3]	TX_BIT_LEN	<p>传输位长</p> <p>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。</p> <p>TX_BIT_LEN = 0x08 8 位.</p> <p>TX_BIT_LEN = 0x09 9 位.</p> <p>.....</p> <p>TX_BIT_LEN = 0x1F 31 位.</p> <p>TX_BIT_LEN = 0x00 32 位.</p>
[2]	TX_NEG	<p>在下降沿发送</p> <p>1 = 在 SPICLK 的下降沿发送数据信号改变</p> <p>0 = 在 SPICLK 的上升沿发送数据信号改变</p>
[1]	RX_NEG	<p>在下降沿接收</p> <p>1 = 在 SPICLK 的下降沿锁存数据输入信号.</p> <p>0 = 在 SPICLK 的上升沿锁存数据输入信号</p>
[0]	GO_BUSY	<p>SPI 传输控制位和忙状态</p> <p>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</p> <p>0 = 停止数据传输.</p> <p>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。</p> <p>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。</p> <p>注意：</p> <p>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。</p>

SPI 分频寄存器 (SPI_DIVIDER)

寄存器	偏移地址	R/W	描述	复位值
SPI_DIVIDER	SPI0_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							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DIVIDER							

位	描述	
[31:24]	Reserved	保留.
[23:16]	DIVIDER2	<p>时钟分频 2 寄存器(仅主模式) 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：</p> $f_{clock2} = \frac{f_{spi_eclk}}{(DIVIDER2 + 1) * 2}$ <p>当 VARCLK_EN 被清为零时，该设置无意义。</p>
[15:8]	Reserved	保留.
[7:0]	DIVIDER	<p>时钟分频器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_CNRLL2[31], 为 0</p> $f_{spi_eclk} = \frac{f_{system_clock}}{(DIVIDER + 1) * 2}$ <p>如果BCn 为 1，</p> $f_{spi_eclk} = \frac{f_{spi_clock_src}}{(DIVIDER + 1)}$ <p>这里</p> $f_{spi_clock_src}$ 是 SPI 引擎时钟源，在 CLKSEL1 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	SPI0_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	Reserved	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 (仅从模式) 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 (仅主模式) 1 = 如果该位被置位, SPI0_SPISS0 信号自动产生。这表示当通过设置 GO_BUSY 位开始发送/接收时, SPI_SSR[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 SPI_SSR[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 (SPI0_SPISS0) 的有效状态。 1 = 从机选择信号 SPI0_SPISS0 在高电平/上升沿有效 0 = 从机选择信号 SPI0_SPISS0 在低电平/下降沿有效
[1]	Reserved 保留.
[0]	SSR 从机选择控制位 (仅主模式)



如果 AUTOSS 位被清零，对该位写1将会激活SPI0_SPISS0线，写 0 则为非激活状态。

如果 AUTOSS 位置 1，写 0 到该位将会保持SPI0_SPISS0线为非激活状态；写 1 将会使 SPI0_SPISS0 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。 SPI0_SPISS0 的激活状态类型由 SS_LVL 指定。

注意： SPI0_SPISS0 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	SPI0_BA+0x10	R	数据接收寄存器0	0x0000_0000
SPI_RX1	SPI0_BA+0x14	R	数据接收寄存器 1	0x0000_0000

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

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	SPI0_BA+0x20	W	数据发送寄存器 0	0x0000_0000
SPI_TX1	SPI0_BA+0x24	W	数据发送寄存器 1	0x0000_0000

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

位	描述
[31:0]	<p>TX</p> <p>数据发送寄存器</p> <p>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_CTRL 寄存器内定义的长度决定。</p> <p>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。</p> <p>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。</p> <p>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</p>

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	SPI0_BA+0x34	R/W	可调时钟类型寄存器	0x007F_FF87

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

位	描述	
[31:0]	VARCLK	<p>可调时钟类型</p> <p>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。</p> <p>更多详细细节请参考可调串行时钟频率章节。</p>



SPI 控制和状态寄存器2 (SPI_CTRL2)

寄存器	偏移地址	R/W	描述	复位值
SPI_CTRL2	SPI0_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 模式使能位 1 = 双 I/O 模式使能。 0 = 双 I/O 模式禁止
[12]	DUAL_IO_DIR	双 I/O 模式方向选择位 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，传输是否已经开始。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位 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-线模式使能位 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 SPI0_CLK, SPI0_MISO0 和 SPI_SPI0_MOSI0。. 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	SPI0_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中断使能位 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	发送阈值中断使能位 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	SPI0_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



		<p>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。</p> <p>注意: 向该位写1清0</p>
[19:17]	Reserved	保留.
[16]	IF	<p>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</p> <p>注意: 向该位写1清0.</p>
[15:12]	RX_FIFO_COUNT	<p>接收 FIFO 数据计数 (只读) 该域表明接收 FIFO 缓存有效数据计数.</p>
[11]	SLV_START_INT_STS	<p>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。</p>
[10:5]	Reserved	保留.
[4]	TX_INTSTS	<p>发送 FIFO 阈值中断状态 (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。</p> <p>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。</p>
[3]	Reserved	保留.
[2]	RX_OVERRUN	<p>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。</p> <p>注意: 向该位写1清0</p>
[1]	Reserved	保留.
[0]	RX_INTSTS	<p>接收 FIFO 阈值中断状态(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。</p> <p>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</p>

6.14 控制器局域网(CAN)

6.14.1 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 6-138）组成。CAN内核通信符合CAN协议规范2.0A和2.0B。位速率最高可达 1Mbit/s。为与物理层相连，还需另外外接收发器硬件。

关于CAN网络的通讯，各个报文对象是可以配置的。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。

6.14.2 特性

- 支持 CAN 协议规范 2.0A 和 2.0 B
- 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

6.14.3 框图

C_CAN与AMBA APB总线相接。下图为C_CAN的框图

- **CAN内核**

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示符掩码

- **寄存器**

所有寄存器用于控制和配置C_CAN

- **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接口与来自ARM的AMBA APB16位总线相接。

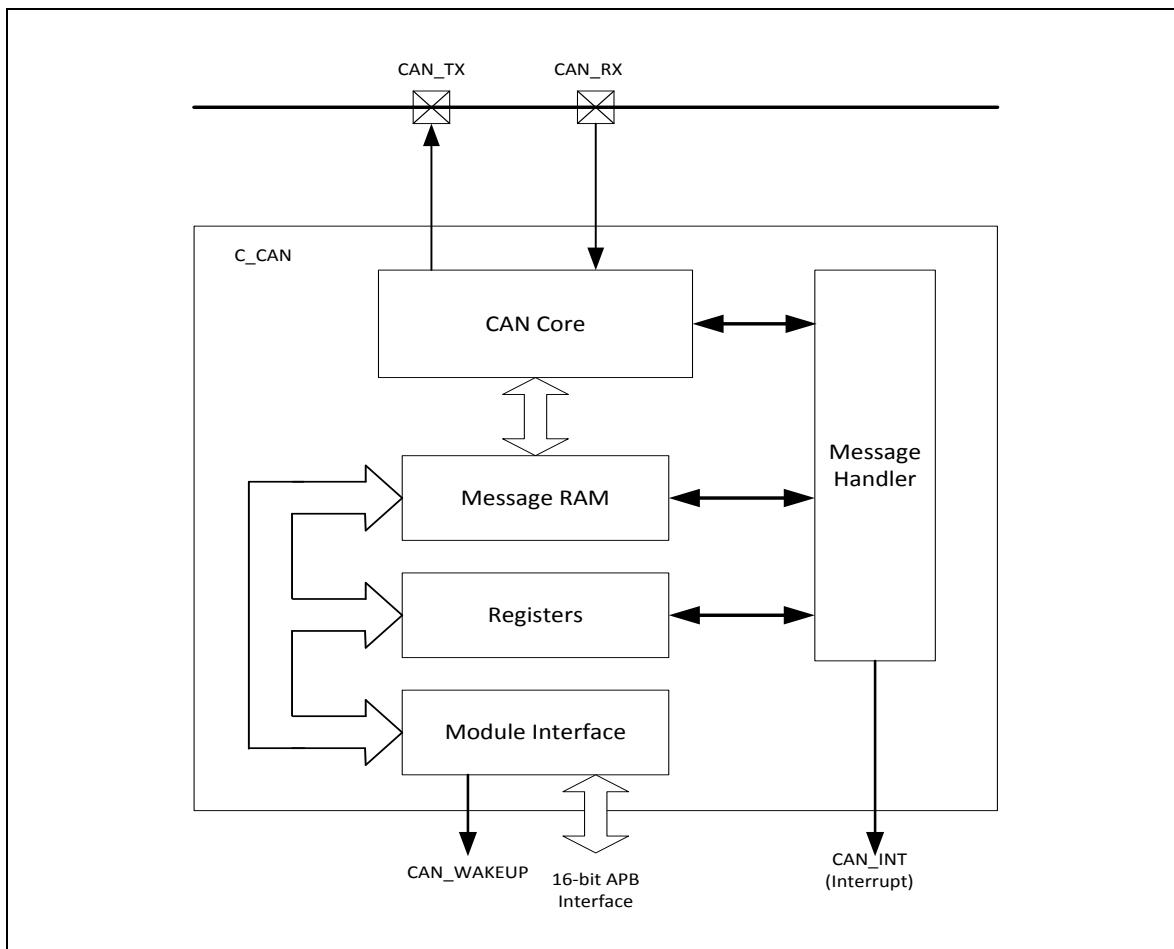


图 6-138 CAN 外设框图

6.14.4 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])).
- 重启CAN控制器(相关寄存器控制位为CAN0_RST (IPRSTC2[24])).



6.14.5 功能描述

6.14.5.1 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，或置位 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 6.5.7.15：配置位时序）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，当配置完成后，需再次设置 MsgVal 位

6.14.5.2 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”位的仲裁位可能会在报文对象中覆盖。

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。

6.14.5.3 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

6.14.6 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(CAN_TEST[7])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。

6.14.6.1 静默模式

通过设置测试寄存器中的Silent位(CAN_TEST[3])为1，CAN内核可被设为静默模式。在静默模式中，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，所以静默模式可以用来分析CAN总线的运输状况。下图为静默模式下，CAN_TX和CAN_RX与CAN内核的信号连接。

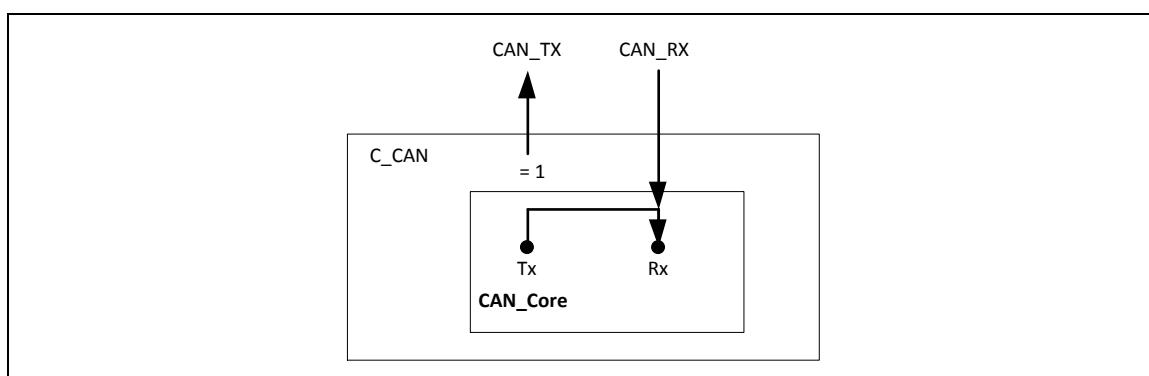


图 6-139 CAN 内核静默模式

6.14.6.2 环回模式

通过设定测试寄存器的LBack位(CAN_TEST[4])为1，CAN内核可被设为环回模式。在环回模式下，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）。下图为环回模式下，CAN_TX和CAN_RX与CAN内核的信号连接。

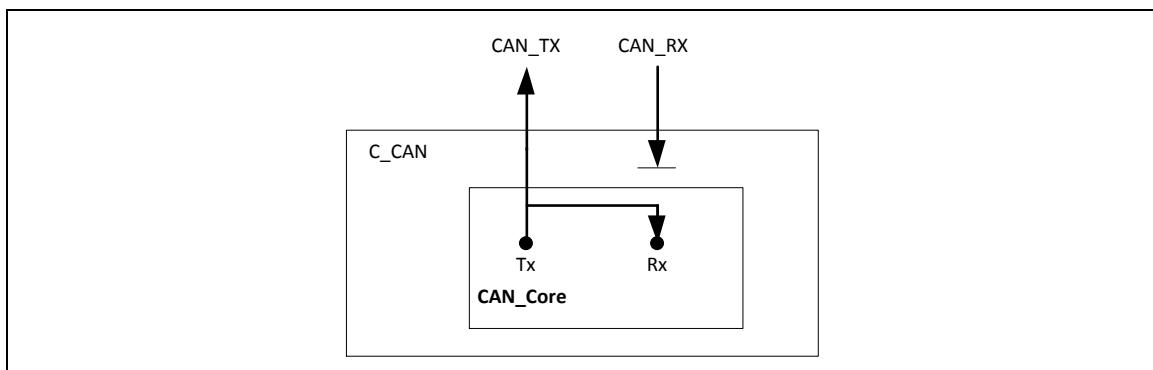


图 6-140 CAN 内核环回模式

该模式用于自检。为了不受外部的干扰，在环回模式下，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。

6.14.6.3 环回模式和静默模式的组合

通过设定LBack (CAN_TEST[4])和Silent (CAN_TEST[3])位同时为1，还可以LBack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。下图为环回模式和静默模式整合下，CAN_TX和CAN_RX与CAN内核的信号连接。

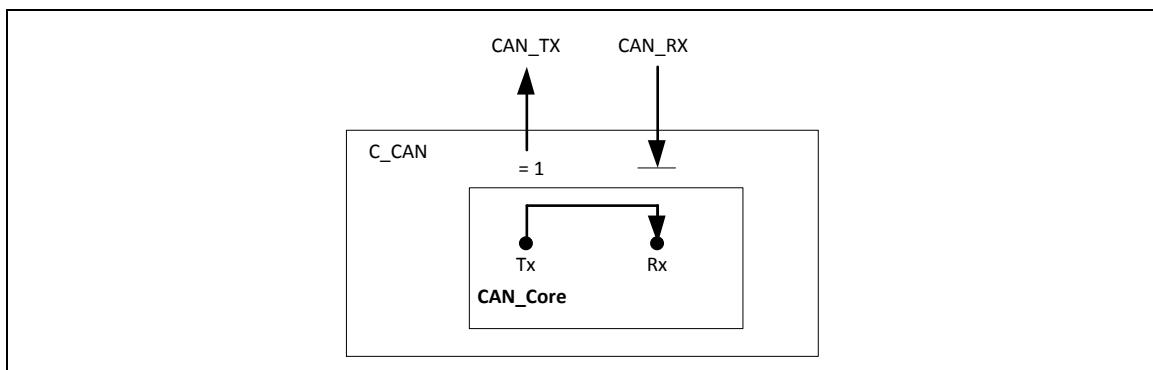


图 6-141 CAN 内核环回模式和静默模式的组合

6.14.6.4 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内核工作在基本模式。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。

在基本模式下，所有与报文对象相关的控制和状态位以及IFn命令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’

6.14.6.5 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 (CAN_TEST[6])和Tx0(CAN_TEST[5])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（环回模式，静默模式，或基本模式）时，CAN_TX 必须使用它的默认功能。

6.14.7 CAN通信

6.14.7.1 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。



6.14.7.2 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

6.14.7.3 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 Busy位(CAN_IFn_CREQ[15])为‘1’。在传输完成后，Busy位会被再次清除（见下图）。

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。

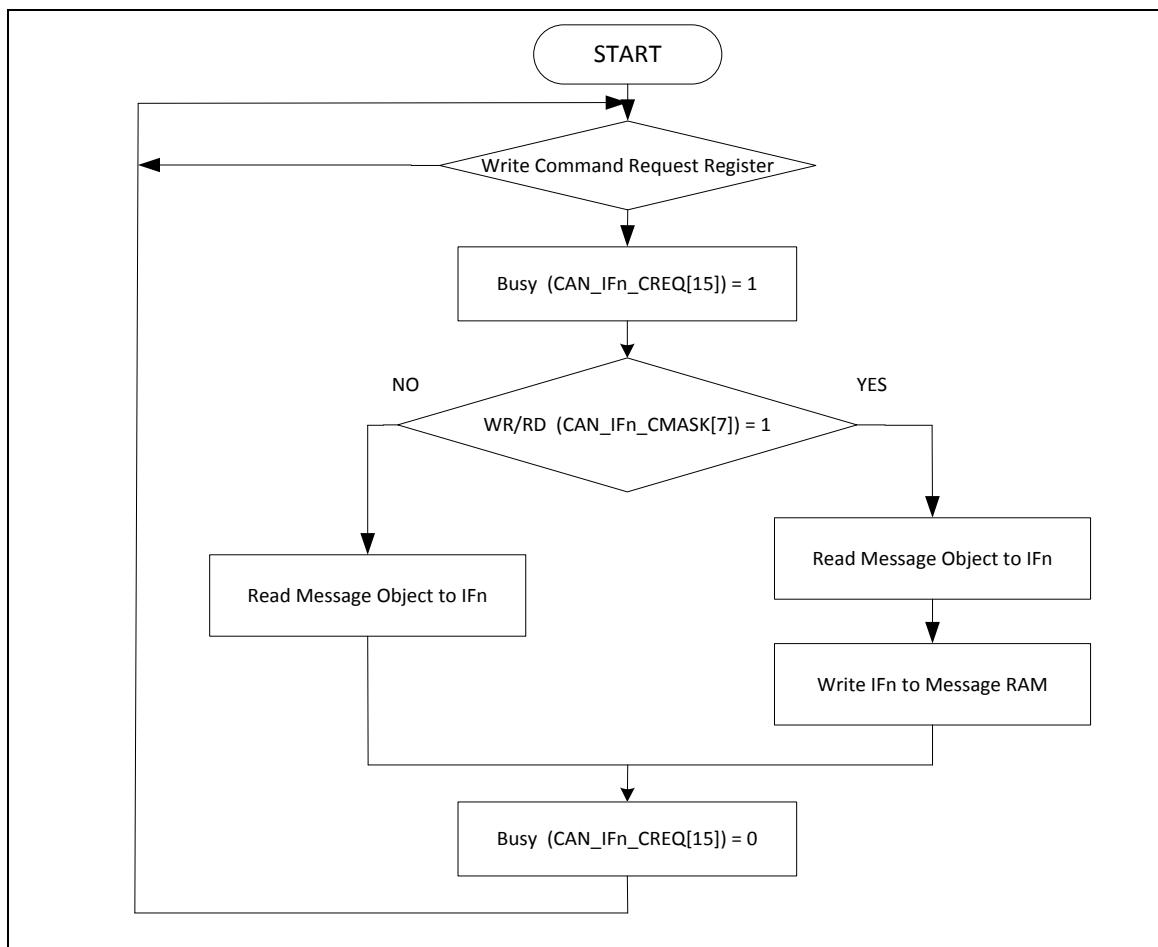


图 6-142 IFn 寄存器和报文 RAM 间的数据传输

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

6.14.7.4 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。

成功传送后或者从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。



6.14.7.5 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，接收过滤单元装载CAN内核移位寄存器的仲裁位。步骤为：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中的各自报文对象。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。如果仲裁掩码寄存器使用的话，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，IntPhd 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' (方向为传送), RmtEn (CAN_IFn_MCON[9]) = '1'和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' (方向为传送), RmtEn = '0'和 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' (方向为传送), RmtEn = '0'和 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

6.14.7.6 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。

6.14.7.7 配置传送对象

下表列出如何初始化一个发送对象。

Ms	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxE	TxE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

表 6-31 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TxE(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])

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。

6.14.7.8 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。

6.14.7.9 配置接收对象

下表表明怎么初始化接收对象。

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxE	TxE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

表 6-32 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。

6.14.7.10 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。

6.14.7.11 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6.5.7.9: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。因为报文对象内含的优先级，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。

6.14.7.12 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。

6.14.7.13 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在复位之前一直反映其状态。

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。

下图表示应用软件如何处理一组和FIFO缓存相连的报文对象。

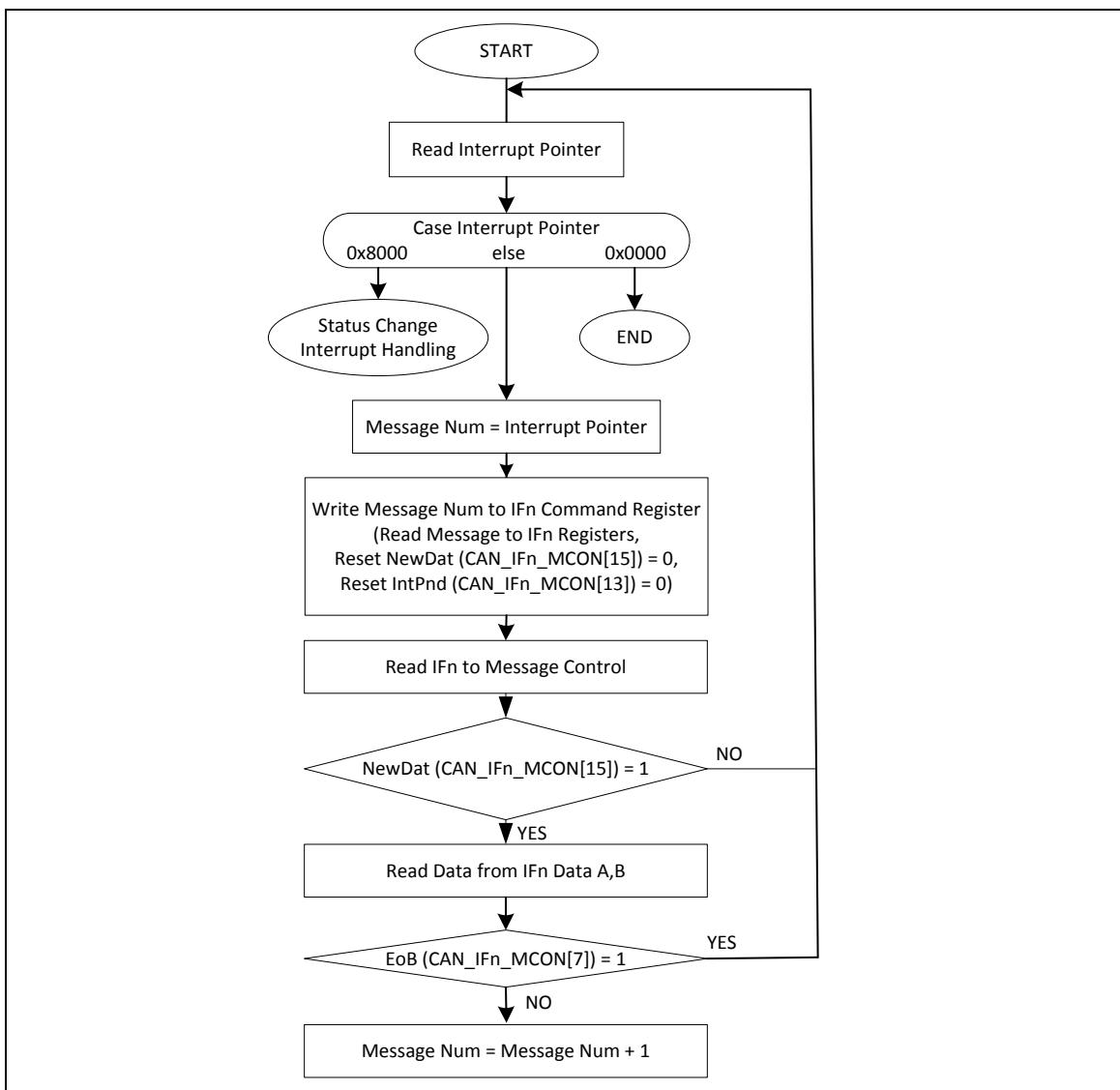


图 6-143 应用软件处理 FIFO 缓存

6.14.7.14 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，IntId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（相应的中断被复位）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。IntId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（位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IntId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 (CIRIntPnd(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。

6.14.7.15 配置位时序

CAN位时序配置的小错误不会导致通讯立即失败，但是会显着降低CAN网络的整体性能。

很多情况下，CAN位同步可以修补CAN位时序的错误配置，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。

6.14.7.16 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。位时序的定时参数（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（见下图）。每段由一个特定的、可编程的时间片组成（见下表）。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BRP位 (CAN_BTIME[5:0]) 定义 : $t_q = \text{BRP} / f_{APB}$.

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。

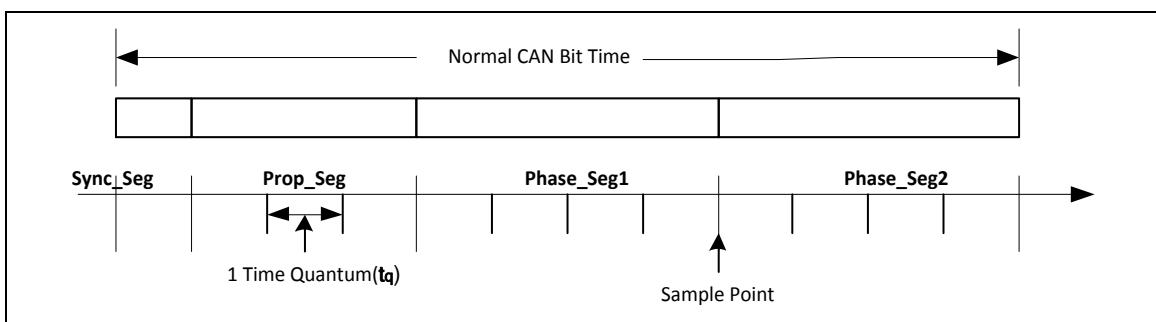


图 6-144 位时序

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..] 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 6-33 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误差范围必须考虑。

6.14.7.17 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。下图的示例表示连两个CAN节点间的相位移动

和传播时间。

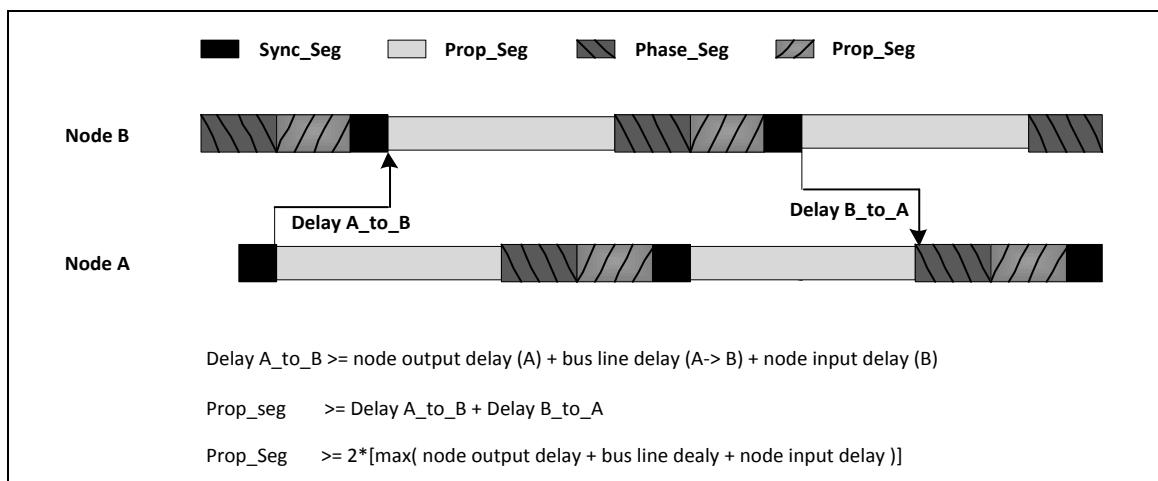


图 6-145 传播时间段

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节点的位时间段须参照A移位。B节点发送的是一个高优先级的标识符，所以在节点B传送一个显性位，而节点A发送一个隐性位时，节点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)后到达节点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数字低通滤波器，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。

6.14.7.18 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，否则被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，否则被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。

在两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。

下图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。

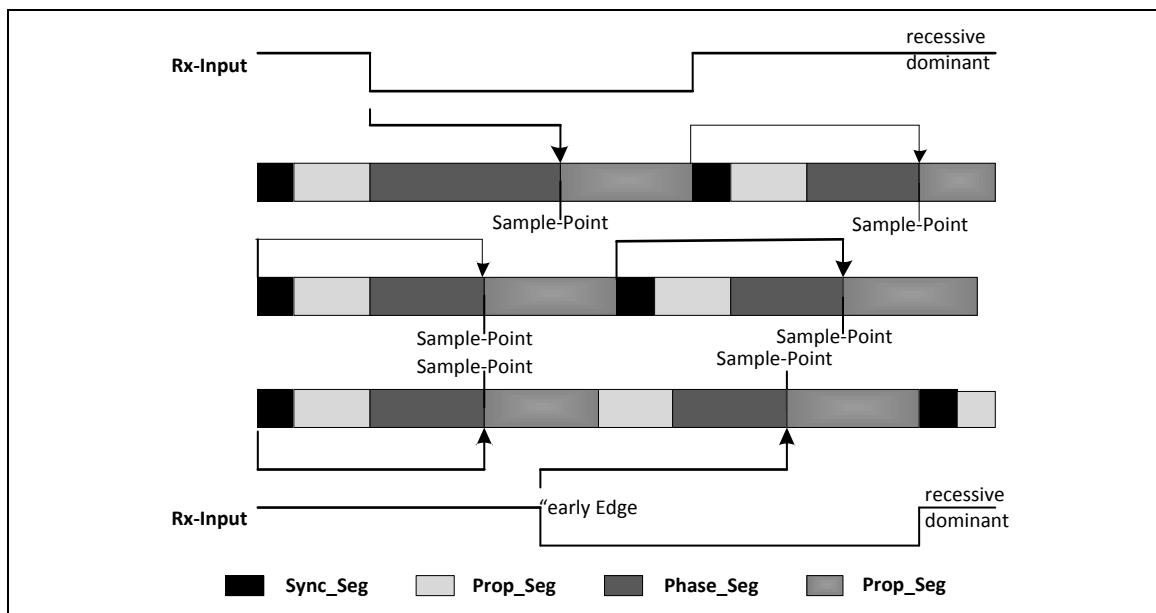


图 6-146 同步在“late”和“early”边沿

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。

下图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。

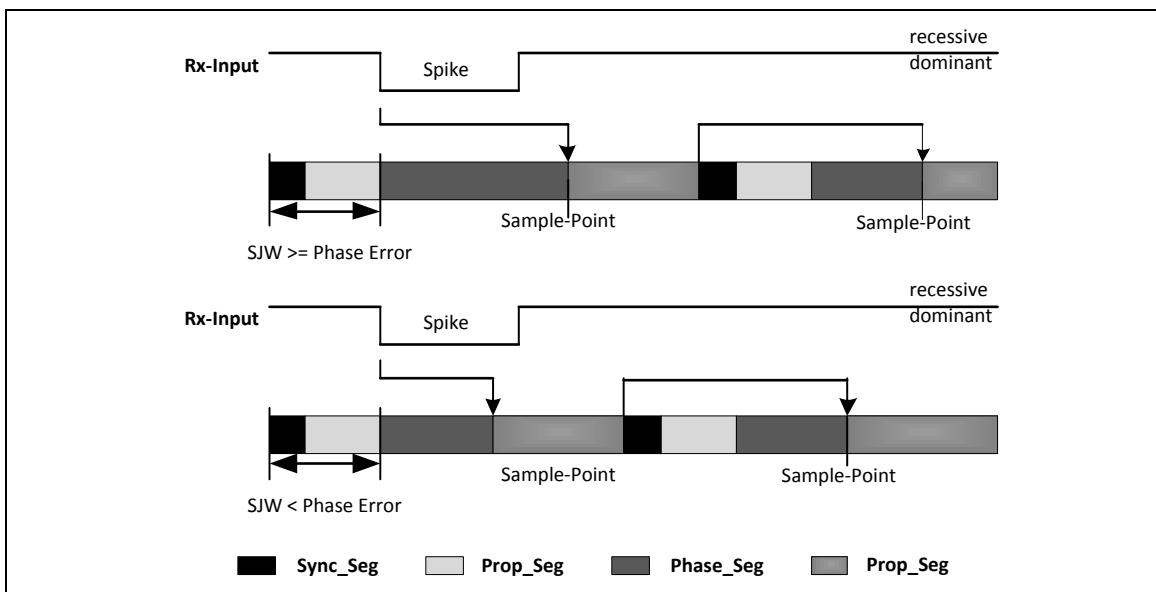


图 6-147 过滤短显性毛刺

6.14.7.19 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），振荡器容差范围没有更新。

振荡器频率 f_{osc} 基于标称频率 f_{nom} 的容差范围 df 的公式为：

$$(1 - d_f) \cdot f_{nom} \leq f_{osc} \leq (1 + d_f) \cdot f_{nom}$$

该值取决于Phase_Seg1, Phase_Seg2, SJW和位时间。最大容错值 df 由两个条件决定（两个都必须满足）：

$$\min (\text{Phase_Seg1}, \text{Phase_Seg2}) \\ \text{I: } d_f \leq \frac{2 * (13 * \text{bit_time} - \text{Phase_Seg2})}{\text{SJW}}$$

$$\text{II: } d_f \leq \frac{20 * \text{bit_time}}{\text{SJW}}$$

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容差为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。

6.14.7.20 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 （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(CAN_BTIME[7:6])和BRP(CAN_BTIME[5:0])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

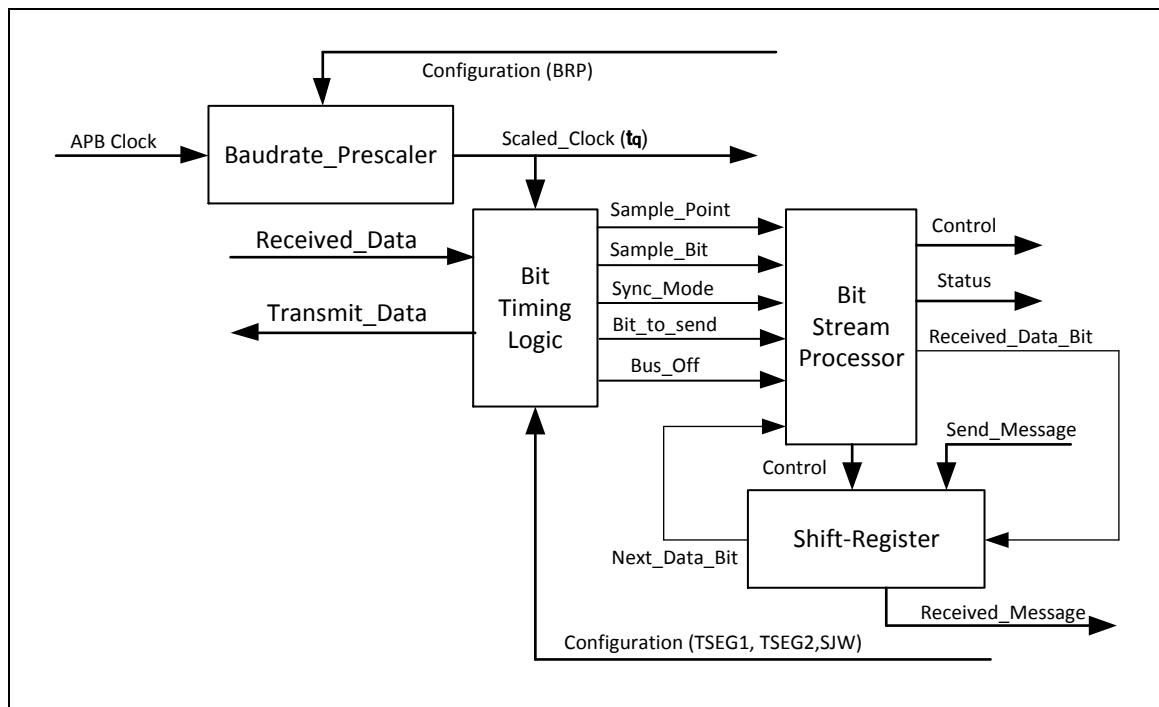


图 6-148 CAN 内核的协议控制器结构

位时序寄存器中的数据是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, and 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时序逻辑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。

6.14.7.21 计算位时序参数

通常，位时序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 $t_q = (\text{Baud Rate Prescaler})/f_{\text{apb_clk}}$ 。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容差范围由章节“振荡器容差范围”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容差范围的配置。

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。

最低容差范围的节点限制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位时序配置。写入位时序寄存器的配置结果为：

(Phase_Seg2-1) & (Phase_Seg1+Prop_Seg-1) & (SynchronisationJumpWidth-1) & (Prescaler-1)



高波特率的位时序示例：

该例中，APB_CLK的时钟为10MHz，BRP (CAN_BTIME[5:0])为0，位速率为1 Mbit/s。

T _q	100	ns	= t _{APB_CLK}
总线驱动延时	50	ns	
接收电路延时	30	ns	
总线（40米）延时	220	ns	
t _{Prop}	600	ns	= 6 • t _q
t _{SJW}	100	ns	= 1 • t _q
t _{TSeg1}	700	ns	= t _{Prop} + t _{SJW}
t _{TSeg2}	200	ns	= 信息处理时间 + 1 • t _q
t _{Sync-Seg}	100	ns	= 1 • t _q
位时间	1000	ns	= t _{Sync-Seg} + t _{TSeg1} + t _{TSeg2}

$$\begin{aligned}
 \text{APB_CLK容差 } 0.39\% &= \frac{\text{Min}(PB1, PB2)}{2 \times 13 \times (\text{bit time} - PB2))} \\
 &= \frac{0.1\mu s}{2 \times 13 \times (1\mu s - 0.2\mu s))
 \end{aligned}$$

该例中，CAN_BTIME 寄存器被配置成 0x1600.



低波特率的位时序示例

该例中，APB_CLK的时钟为2MHz，BRP (CAN_BTME[5:0])为1，位速率为100 Kbit/s。

t_q	1	us	$= 2 \cdot t_{APB_CLK}$
总线驱动延时	200	ns	
接收电路延时	80	ns	
总线（40米）延时	220	ns	
t_{Prop}	1	us	$= 1 \cdot t_q$
t_{SJW}	4	us	$= 4 \cdot t_q$
t_{TSeg1}	5	us	$= t_{Prop} + t_{SJW}$
t_{TSeg2}	4	us	$= \text{信息处理时间} + 3 \cdot t_q$
$t_{Sync-Seg}$ 1us	1	us	$= 1 \cdot t_q$
位时间	10	us	$= t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$

$$\begin{aligned}
 APB_CLK容差 & 1.58 \% = \frac{\text{Min}(PB1, PB2)}{2 \times 13 \times (\text{bit time} - PB2))} \\
 & = \frac{4\mu s}{2 \times 13 \times (10\mu s - 4\mu s))
 \end{aligned}$$

该例中，CAN_BTME寄存器被配置成0x34C1.

6.14.8 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。直到应用软件复位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寄存器表中各个位的功能

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									Test	CCE	DAR	Res	EIE	IE	Init	
04h	CAN_STATUS									BOff	EWarn	EPass	RxOk	TxOk	SIE	LEC	
08h	CAN_ERR	R P														TEC7-0	
0Ch	CAN_BTIME	Res		TSeg2		TSeg1				SJW						BRP	
10h	CAN_IIDR				IntId15-8											IntId7-0	
14h	CAN_TEST					保留				RX	Tx1	Tx0	LBack	Silent	Basic	保留	
18h	CAN_BRPE					保留										BRPE	
20h	CAN_IF1_CREQ	Bus y				保留										Message Number	
24h	CAN_IF1_CMASK					保留				WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	
28h	CAN_IF1_MASK1									Msk15-0						Data B	
2Ch	CAN_IF1_MASK2	MXt d	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	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0				
38h	CAN_IF1_MCO_N	NewDat	MsgLst	IntPnd	UMask	TxE	RxE	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_CRE_Q	Busy	保留								Message Number										
84h	CAN_IF2_CMA_SK	保留								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B				
88h	CAN_IF2_MAS_K1	Msk15-0																			
8Ch	CAN_IF2_MAS_K2	MXt_d	MDir	Res.	Msk28-16																
90h	CAN_IF2_ARB_1	ID15-0																			
94h	CAN_IF2_ARB_2	MsgVal	Xtd	Dir	ID28-16																
98h	CAN_IF2_MCO_N	NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst	EoB	保留				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(2)
A4h	CAN_IF2_DAT_B1																Data(4)
A8h	CAN_IF2_DAT_B2																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																WAKU_P_EN
16Ch	CAN_WU_STA_TUS																WAKU_P_STS
170h	CAN_RAM_CE_N																RAM_CEN
Others	Reserved																保留

表 6-34 CAN 寄存器表对应各个位的功能

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



6.14.9 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。

6.14.10 寄存器映射

R: 只读, **W:** 只写, **R/W:** 可读可写

寄存器	偏移量	R/W	描述	复位值
CAN 基地址:				
CAN0_BA = 0x4018_0000				
CAN_CON	CAN0_BA+0x00	R/W	控制寄存器	0x0000_0001
CAN_STATUS	CAN0_BA+0x04	R/W	状态寄存器	0x0000_0000
CAN_ERR	CAN0_BA+0x08	R	错误计数寄存器	0x0000_0000
CAN_BTIME	CAN0_BA+0x0C	R/W	位时序寄存器	0x0000_2301
CAN_IIDR	CAN0_BA+0x10	R	中断标识符寄存器	0x0000_0000
CAN_TEST	CAN0_BA+0x14	R/W	测试寄存器 (寄存器映射备注1)	0x0000_0080
CAN_BRPE	CAN0_BA+0x18	R/W	波特率预分频扩充寄存器	0x0000_0000
CAN_IF1_CREQ	CAN0_BA+0x20	R/W	IF1命令请求寄存器 (寄存器映像备注2)	0x0000_0001
CAN_IF2_CREQ	CAN0_BA+0x80	R/W	IF2命令请求寄存器 (寄存器映像备注2)	0x0000_0001
CAN_IF1_CMASK	CAN0_BA+0x24	R/W	IF1命令掩码寄存器	0x0000_0000
CAN_IF2_CMASK	CAN0_BA+0x84	R/W	IF2命令掩码寄存器	0x0000_0000
CAN_IF1_MASK1	CAN0_BA+0x28	R/W	IF1掩码1寄存器	0x0000_FFFF
CAN_IF2_MASK1	CAN0_BA+0x88	R/W	IF2掩码1寄存器	0x0000_FFFF
CAN_IF1_MASK2	CAN0_BA+0x2C	R/W	IF1掩码2寄存器	0x0000_FFFF
CAN_IF2_MASK2	CAN0_BA+0x8C	R/W	IF2掩码2寄存器	0x0000_FFFF
CAN_IF1_ARB1	CAN0_BA+0x30	R/W	IF1仲裁1寄存器	0x0000_0000
CAN_IF2_ARB1	CAN0_BA+0x90	R/W	IF2仲裁1寄存器	0x0000_0000
CAN_IF1_ARB2	CAN0_BA+0x34	R/W	IF1仲裁2寄存器	0x0000_0000
CAN_IF2_ARB2	CAN0_BA+0x94	R/W	IF2仲裁2寄存器	0x0000_0000
CAN_IF1_MCON	CAN0_BA+0x38	R/W	IF1报文控制寄存器	0x0000_0000

CAN_IF2_MCON	CAN0_BA+0x98	R/W	IF2报文控制寄存器	0x0000_0000
CAN_IF1_DAT_A1	CAN0_BA+0x3C	R/W	IF1 数据A1 寄存器 (寄存器映射备注3)	0x0000_0000
CAN_IF1_DAT_A2	CAN0_BA+0x40	R/W	IF1 数据A2 寄存器 (寄存器映射备注3)	0x0000_0000
CAN_IF1_DAT_B1	CAN0_BA+0x44	R/W	IF1 数据B1 寄存器 (寄存器映射备注3)	0x0000_0000
CAN_IF1_DAT_B2	CAN0_BA+0x48	R/W	IF1 数据B2 寄存器 (寄存器映射备注3)	0x0000_0000
CAN_IF2_DAT_A1	CAN0_BA+0x9C	R/W	IF2 数据A1 寄存器 (寄存器映射备注3)	0x0000_0000
CAN_IF2_DAT_A2	CAN0_BA+0xA0	R/W	IF2 数据A2 寄存器 (寄存器映射备注3)	0x0000_0000
CAN_IF2_DAT_B1	CAN0_BA+0xA4	R/W	IF2 数据B1 寄存器 (寄存器映射备注3)	0x0000_0000
CAN_IF2_DAT_B2	CAN0_BA+0xA8	R/W	IF2 数据B2 寄存器 (寄存器映射备注3)	0x0000_0000
CAN_TXREQ1	CAN0_BA+0x100	R	发送请求寄存器1	0x0000_0000
CAN_TXREQ2	CAN0_BA+0x104	R	发送请求寄存器2	0x0000_0000
CAN_NDAT1	CAN0_BA+0x120	R	新数据寄存器1	0x0000_0000
CAN_NDAT2	CAN0_BA+0x124	R	新数据寄存器2	0x0000_0000
CAN_IPND1	CAN0_BA+0x140	R	中断挂起寄存器1	0x0000_0000
CAN_IPND2	CAN0_BA+0x144	R	中断挂起寄存器2	0x0000_0000
CAN_MVLD1	CAN0_BA+0x160	R	报文有效寄存器1	0x0000_0000
CAN_MVLD2	CAN0_BA+0x164	R	报文有效寄存器2	0x0000_0000
CAN_WU_EN	CAN0_BA+0x168	R/W	唤醒使能寄存器	0x0000_0000
CAN_WU_STATUS	CAN0_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



CAN控制寄存器(CAN_CON)

寄存器	偏移量	R/W	描述	复位值
CAN_CON	CAN0_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	测试模式使能位 0 = 正常操作模式 1 = 测试模式
[6]	CCE	配置改变使能位 0 = 禁止写访问到位时序寄存器 1 = 允许写访问到位时序寄存器(CAN_BTME)(当Init(CAN_CON[0]) = 1)
[5]	DAR	禁止自动重传位 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	错误中断使能位 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	状态改变中断使能位 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	模块中断使能位 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住, 从而监控busoff修复序列的进程。



CAN状态寄存器 (CAN_STATUS)

寄存器	偏移量	R/W	描述	复位值
CAN_STATUS	CAN0_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(无错误或至少被节点所应答)
[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。下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 6-35 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	偏移量	R/W	描述	复位值
CAN_ERR	CAN0_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						
7	6	5	4	3	2	1	0
TEC							

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)

计数器	偏移量	R/W	描述	复位值
CAN_BTIME	CAN0_BA+0x0C	R/W	位时序寄存器	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位(CAN_CON[6])和Init位(CAN_CON[0])被置位后才可写。



中断标识符寄存器(CAN_IIDR)

寄存器	偏移量	R/W	描述	复位值
CAN_IIDR	CAN0_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							
7	6	5	4	3	2	1	0
IntId							

位	描述	
[31:16]	Reserved	保留
[15:0]	IntId	<p>中断标识符 (表明中断源)</p> <p>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。</p> <p>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。</p> <p>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。</p>

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 6-36 中断源



测试寄存器 (CAN_TEST)

寄存器	偏移量	R/W	描述	复位值
CAN_TEST	CAN0_BA+0x14	R/W	测试寄存器 (寄存器映射备注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		LBack	Silent	Basic	Reserved	

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	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	环回模式使能位 0=环回模式禁止 1=环回模式使能
[3]	Silent	静默模式 0 = 正常操作 1 = 模块工作在静默模式
[2]	Basic	基本模式 0 = 基本模式禁止 1= IF1 寄存器用作TX Buffer, IF2寄存器用作RX Buffer.
[1:0]	Reserved	保留

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(CAN_CON[7])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	偏移量	R/W	描述	复位值
CAN_BRPE	CAN0_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。下表提供了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
CAN0_BA+0x20	IF1 命令请求	CAN0_BA+0x80	IF2 命令请求
CAN0_BA+0x24	IF1 命令掩码	CAN0_BA+0x84	IF2 命令掩码
CAN0_BA+0x28	IF1 掩码1	CAN0_BA+0x88	IF2 掩码1
CAN0_BA+0x2C	IF1 掩码2	CAN0_BA+0x8C	IF2 掩码2
CAN0_BA+0x30	IF1 仲裁1	CAN0_BA+0x90	IF2 仲裁1
CAN0_BA+0x34	IF1 仲裁2	CAN0_BA+0x94	IF2 仲裁2
CAN0_BA+0x38	IF1 报文控制	CAN0_BA+0x98	IF2 报文控制
CAN0_BA+0x3C	IF1 数据 A 1	CAN0_BA+0x9C	IF2 数据 A 1
CAN0_BA+0x40	IF1 数据 A 2	CAN0_BA+0xA0	IF2 数据 A 2
CAN0_BA+0x44	IF1 数据B 1	CAN0_BA+0xA4	IF2 数据B 1
CAN0_BA+0x48	IF1 数据B 2	CAN0_BA+0xA8	IF2 数据B 2

表 6-37 IF1 和 IF2 报文接口寄存器



IFn 命令请求寄存器 (CAN_IFn_CREQ)

寄存器	偏移量	R/W	描述	复位值
CAN_IF1_CREQ	CAN0_BA+0x20	R/W	IFn (寄存器映像备注 2) 命令请求寄存器	0x0000_0001
CAN_IF2_CREQ	CAN0_BA+0x80	R/W	IFn (寄存器映像备注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	Reserved						
7	6	5	4	3	2	1	0
Reserved		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，如 0x20的说明。 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(CAN_IFn_CREQ[15])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	偏移量	R/W	描述	复位值
CAN_IF1_CMASK	CANO_BA+0x24	R/W	IF1 命令掩码寄存器	0x0000_0000
CAN_IF2_CMASK	CANO_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 + Xtd + MsgVal到IFn报文缓存寄存器

[4]	Control	<p>访问控制位</p> <p>写操作:</p> <p>0=控制位不变 1=传输控制位到报文对象</p> <p>读操作:</p> <p>0=控制位不变 1=传输控制位到IFn报文缓存寄存器</p>
[3]	ClrIntPnd	<p>清除中断挂起位:</p> <p>写操作:</p> <p>当写报文对象时，该位忽略。</p> <p>读操作:</p> <p>0=IntPnd(CAN_IFn_MCON[13])位保持不变 1=清除报文对象中的IntPnd位</p>
[2]	TxRqst/NewDat	<p>写操作时访问传送请求位</p> <p>0=TxRqst位不变 1=设置TxRqst位</p> <p>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。</p> <p>读操作时访问New Data位</p> <p>0>NewDat位保持不变 1=清除报文对象中的NewDat位</p> <p>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）</p>
[1]	DAT_A	<p>访问数据字节[3: 0]</p> <p>写操作:</p> <p>0=数据[3:0]未变 1=写传输数据字节[3:0]到报文对象</p> <p>读操作:</p> <p>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</p>
[0]	DAT_B	<p>访问数据字节[7: 4]</p> <p>写操作:</p> <p>0=数据[7:4]未变 1=写传输数据字节[7:4]到报文对象</p> <p>读操作:</p> <p>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</p>



IFn 掩码1 寄存器 (CAN_IFn_MASK1)

寄存器	偏移量	R/W	描述	复位值
CAN_IF1_MASK1	CAN0_BA+0x28	R/W	IF1 掩码1 寄存器	0x0000_FFFF
CAN_IF2_MASK1	CAN0_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
Msk15-8							
7	6	5	4	3	2	1	0
Msk7-0							

位	描述	
[31:16]	Reserved	保留
[15:0]	Msk15-0	<p>标识符掩码15-0</p> <p>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</p>



IFn 掩码2 寄存器 (CAN_IFn_MASK2)

寄存器	偏移量	R/W	描述	复位值
CAN_IF1_MASK2	CAN0_BA+0x2C	R/W	IF1 掩码2 寄存器	0x0000_FFFF
CAN_IF2_MASK2	CAN0_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	Msk28-24				
7	6	5	4	3	2	1	0
Msk23-16							

位	描述	
[31:16]	Reserved	保留
[15]	MXtd	掩码扩展标识符 0=扩展标识符 (IDE) 对于接收过滤没有影响 1=扩展标识符 (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 0=报文方向(Dir (CAN_IFn_ARB2[13]))位对于接收过滤没有影响 1=报文方向(Dir (CAN_IFn_ARB2[13]))位用于接收过滤
[13]	Reserved	保留
[12:0]	Msk28-16	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	偏移量	R/W	描述	复位值
CAN_IF1_ARB1	CAN0_BA+0x30	R/W	IF1仲裁1寄存器	0x0000_0000
CAN_IF2_ARB1	CAN0_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
ID15-8							
7	6	5	4	3	2	1	0
ID7-0							

位	描述	
[31:16]	Reserved	保留.
[15:0]	ID15-0	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	偏移量	R/W	描述	复位值
CAN_IF1_ARB2	CAN0_BA+0x34	R/W	IF1仲裁2寄存器	0x0000_0000
CAN_IF2_ARB2	CAN0_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	ID28-24				
7	6	5	4	3	2	1	0
ID23-16							

位	描述	
[31:16]	Reserved	保留
[15]	MsgVal	<p>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</p> <p>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。在修改ID28-0(CAN_IFn_ARB1/2)，控制位Xtd(CAN_IFn_ARB2[14]), Dir (CAN_IFn_ARB2[13]), 或数据长度码DLC23-0(CAN_IFn_MCON[3:0])这些位之前需将此位复位，如果报文对象不再需要时也要将此位复位。</p>
[14]	Xtd	<p>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</p>
[13]	Dir	<p>报文方向 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 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(CAN_IFn_CMASK[2])被置位。（如果RmtEn(CAN_IFn_MCON[9]) =1）</p>
[12:0]	ID28-16	<p>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”）</p>



IFn报文控制寄存器 (CAN_IFn_MCON)

寄存器	偏移量	R/W	描述	复位值
CAN_IF1_MCON	CAN0_BA+0x38	R/W	IF1 报文控制寄存器	0x0000_0000
CAN_IF2_MCON	CAN0_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	TxE	RxE	RmtEn	TxRqst
7	6	5	4	3	2	1	0
EoB	Reserved			DLC			

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]	TxE	发送中断使能位 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]	RxE	接收中断使能位



		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	远程使能位 0 = 在收到一远程帧后, TxRqst(CAN_IFn_MCON[8])保持不变 1 = 在收到一远程帧后, TxRqst(CAN_IFn_MCON[8])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 6: 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	偏移量	R/W	描述	复位值
CAN_IF1_DAT_A1	CAN0_BA+0x3C	R/W	IF1 数据A1 寄存器(寄存器映射备注 3)	0x0000_0000
CAN_IF2_DAT_A1	CAN0_BA+0x9C	R/W	IF2 数据A1 寄存器(寄存器映射备注 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	偏移量	R/W	描述	复位值
CAN_IF1_DAT_A2	CAN0_BA+0x40	R/W	IF1 数据A2 寄存器(寄存器映射备注 3)	0x0000_0000
CAN_IF2_DAT_A2	CAN0_BA+0xA0	R/W	IF2 数据A2 寄存器(寄存器映射备注 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	偏移量	R/W	描述	复位值
CAN_IF1_DAT_B1	CAN0_BA+0x44	R/W	IF1 数据B1 寄存器(寄存器映射备注 3)	0x0000_0000
CAN_IF2_DAT_B1	CAN0_BA+0xA4	R/W	IF2 数据B1 寄存器(寄存器映射备注 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	偏移量	R/W	描述	复位值
CAN_IF1_DAT_B2	CAN0_BA+0x48	R/W	IF1 数据B2 寄存器(寄存器映射备注 3)	0x0000_0000
CAN_IF2_DAT_B2	CAN0_BA+0xA8	R/W	IF2 数据B2 寄存器(寄存器映射备注 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。下表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]	Data(0)	Data(1)	Data(2)	Data(3)	Data(4)	Data(5)	Data(6)	Data(7)	

表 6-38 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]), 以及中断标志）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。在收到一远程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	偏移量	R/W	描述	复位值
CAN_TXREQ1	CAN0_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
TxRqst16-9							
7	6	5	4	3	2	1	0
TxRqst8-1							

位	描述	
[31:16]	Reserved	保留
[15:0]	TxRqst16-1	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	偏移量	R/W	描述	复位值
CAN_TXREQ2	CAN0_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]	TxRqst32-17	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	偏移量	R/W	描述	复位值
CAN_NDAT1	CAN0_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
NewData8-1							

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	偏移量	R/W	描述	复位值
CAN_NDAT2	CAN0_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
NewData32-25							
7	6	5	4	3	2	1	0
NewData24-17							

位	描述	
[31:16]	Reserved	保留
[15:0]	NewData32-17	<p>新数据位 32-17 (所有 报文对象)</p> <p>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</p>



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。这也会影响中断寄存器的IntId的值。

寄存器	偏移量	R/W	描述				复位值
CAN_IPND1	CAN0_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
IntPnd8-1							

位	描述	
[31:16]	Reserved	保留
[15:0]	IntPnd16-1	<p>中断挂起位16-1(所有报文对象)</p> <p>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</p>



中断挂起寄存器 2 (CAN_IPND2)

寄存器	偏移量	R/W	描述	复位值
CAN_IPND2	CAN0_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
IntPnd32-25							
7	6	5	4	3	2	1	0
IntPnd24-17							

位	描述	
[31:16]	Reserved	保留
[15:0]	IntPnd32-17	<p>中断挂起位32-17(所有报文对象)</p> <p>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</p>



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	偏移量	R/W	描述	复位值
CAN_MVLD1	CAN0_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
MsgVal16- 9							
7	6	5	4	3	2	1	0
MsgVal8-1							

位	描述	
[31:16]	Reserved	保留
[15:0]	MsgVal16-1	<p>报文有效位16-1 (所有报文对象) (只读)</p> <p>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</p> <p>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。</p>



报文有效寄存器 2 (CAN_MVLD2)

寄存器	偏移量	R/W	描述	复位值
CAN_MVLD2	CAN0_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
MsgVal32-25							
7	6	5	4	3	2	1	0
MsgVal24-17							

位	描述	
[31:16]	Reserved	保留
[15:0]	MsgVal32-17	<p>报文有效位32-17 (所有报文对象) (只读)</p> <p>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</p> <p>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</p>



唤醒使能寄存器 (CAN_WU_EN)

寄存器	偏移量	R/W	描述	复位值
CAN_WU_EN	CAN0_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	<p>唤醒使能位 0 = 禁止唤醒功能 1 = 使能唤醒功能</p> <p>注意: 当CAN_Rx管脚上有下降沿信号时, 用户可唤醒系统</p>



唤醒状态寄存器 (CAN_WU_STATUS)

寄存器	偏移量	R/W	描述	复位值
CAN_WU_STATUS	CAN0_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	<p>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</p>



6.15 模拟数字转换(ADC)

6.15.1 概述

NuMicro™ NUC131 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、BPWM触发器和外部STADC管脚启动转换。

6.15.2 特性

- 模拟输入电压范围 : 0~ V_{REF}
- 12位分辨率和10位精确度保证
- 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，转换顺序从最小号信道到最大号信道。
 - 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 和 BPWM 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2 路输入源：外部模拟电压，内部带隙电压

6.15.3 框图

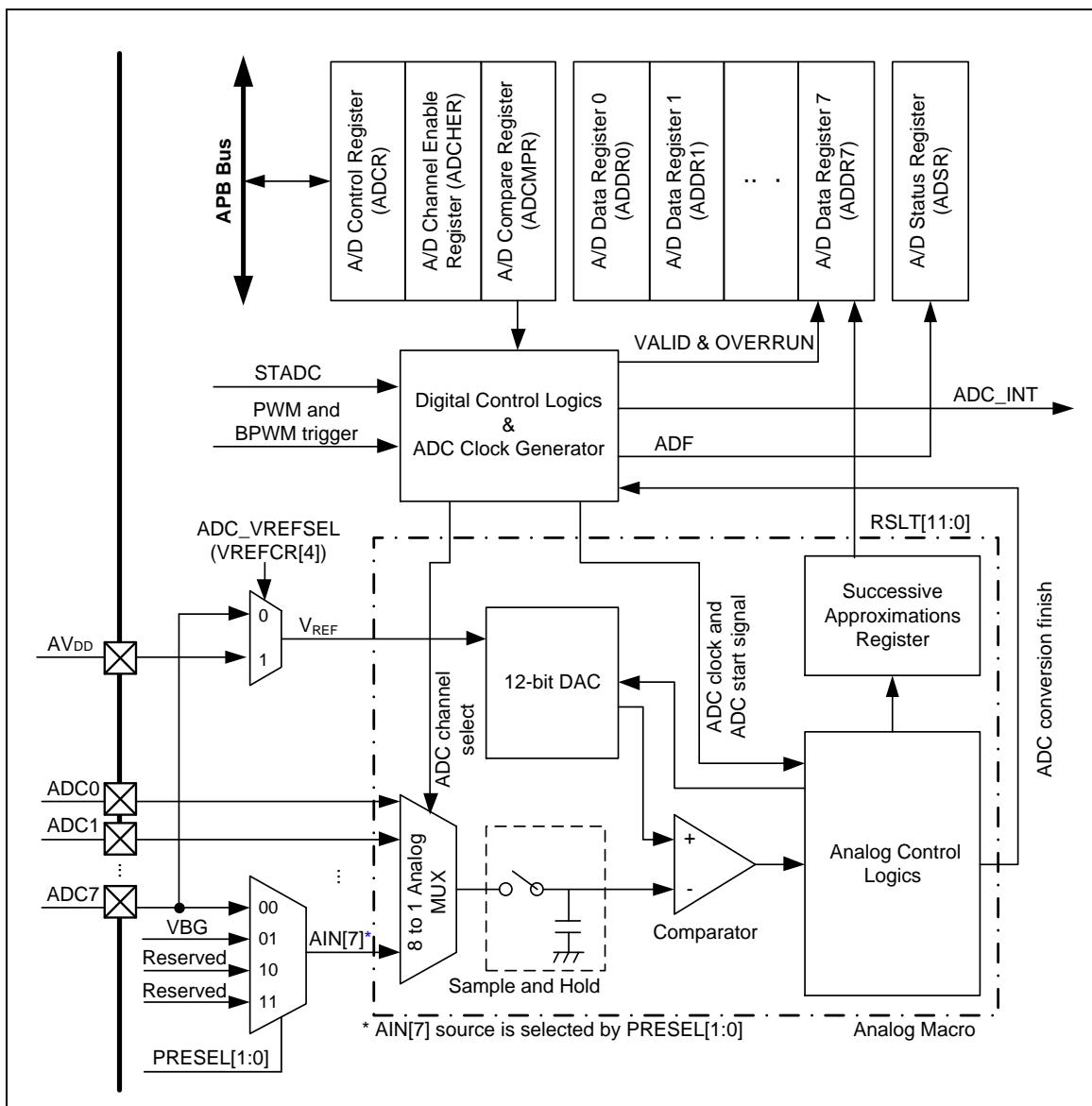


图 6-149 ADC 控制器框图

6.15.4 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，将管脚的数字输入通道关闭。

6.15.5 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。

6.15.5.1 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：

$$\text{ADC时钟频率} = (\text{ADC 时钟源频率}) / (\text{ADC_N (CLKDIV[23:16])} + 1);$$

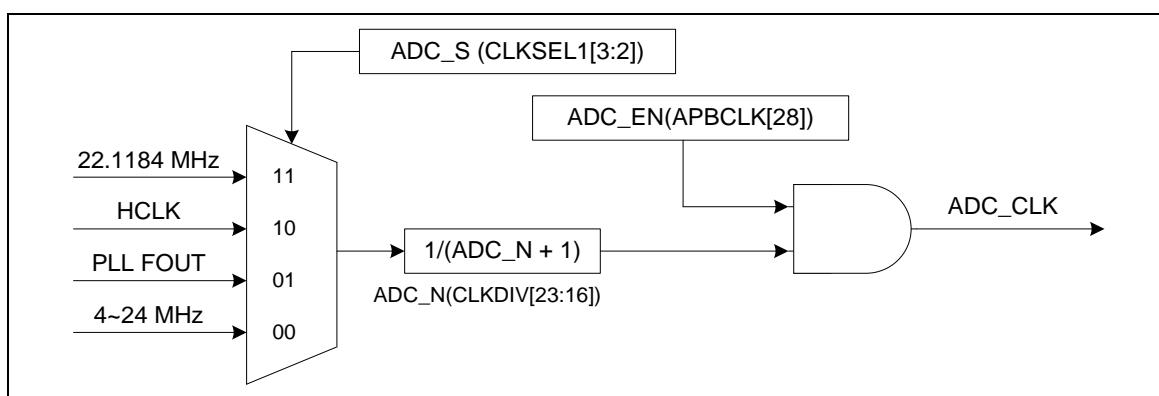


图 6-150 ADC 时钟控制

6.15.5.2 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：

当ADST (ADCR[11]) 位被软件置位时，A/D转换开始。

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

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。

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

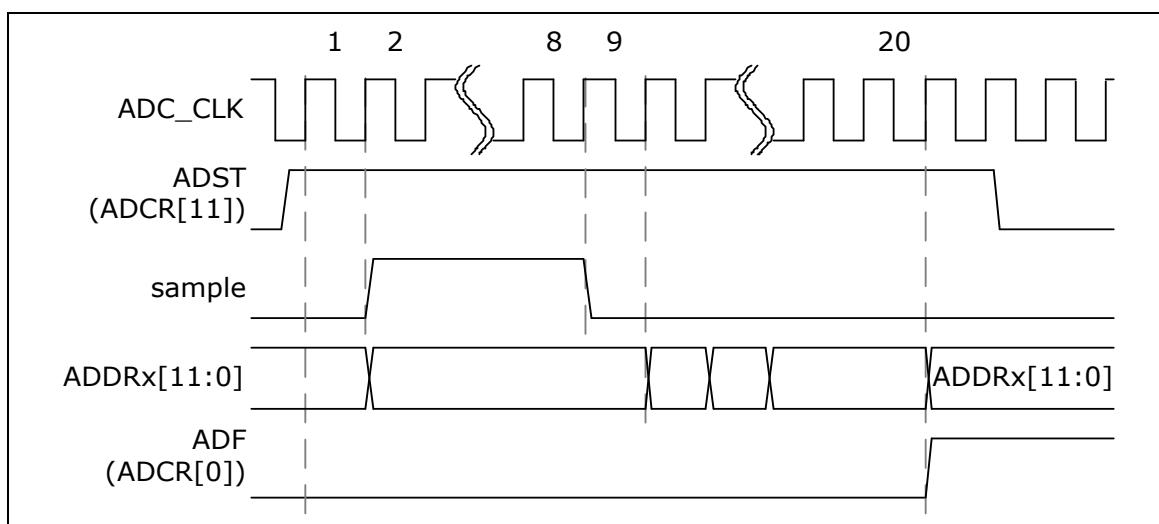


图 6-151 单一转换模式时序图

6.15.5.3 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 ADDR_x 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：

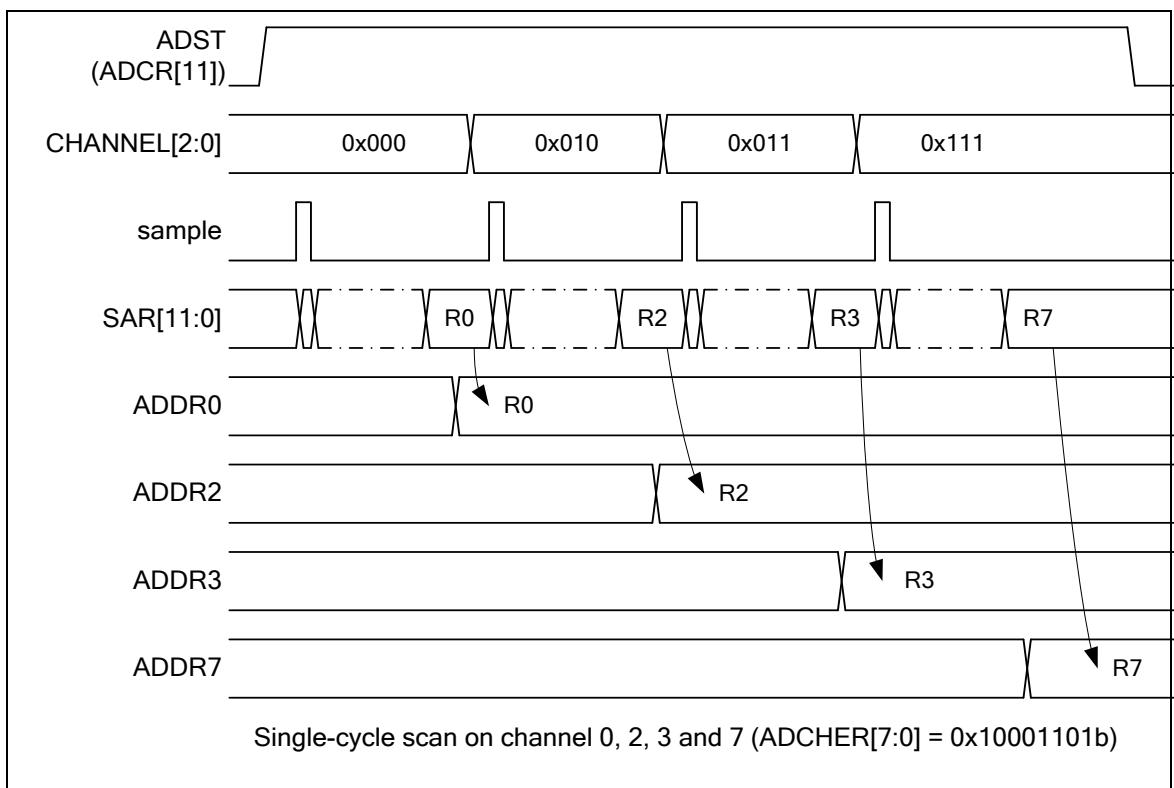


图 6-152 使能通道上的单周期扫描模式时序图

6.15.5.4 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 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：

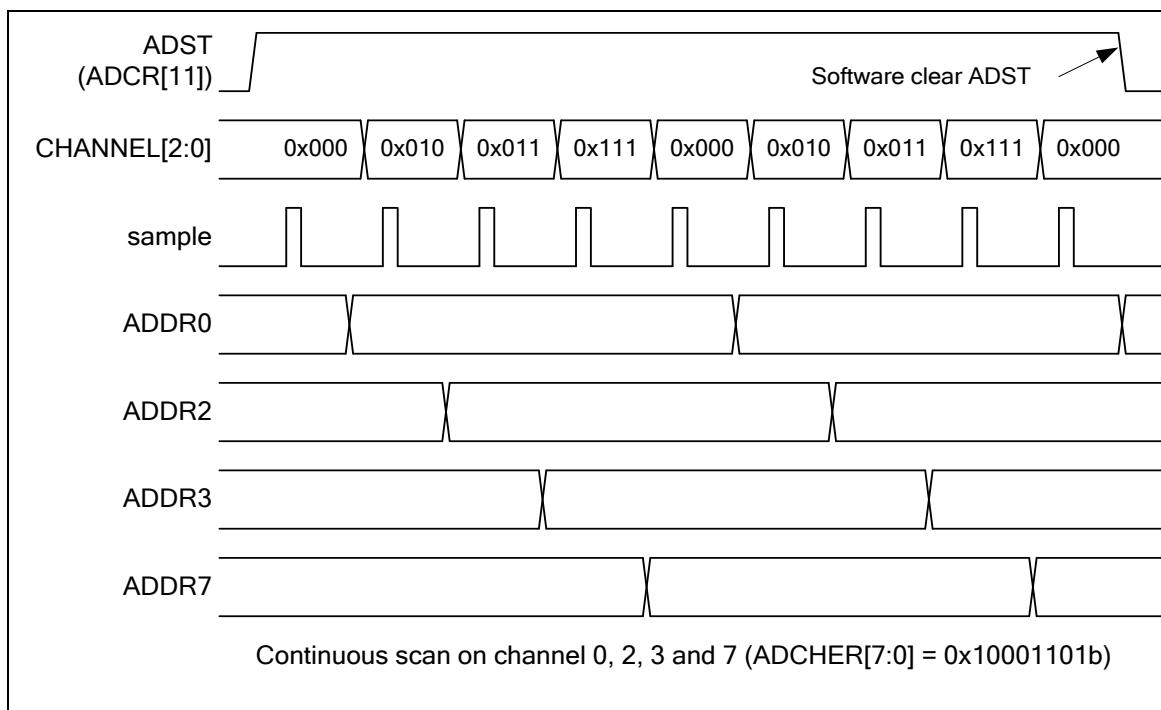


图 6-153 使能信道上的连续扫描模式时序图

6.15.5.5 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 来使用 ADC 外部触发功能时，设定 TRGS (ADCR[5:4]) 为 00 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/下边沿触发还是高/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 PCLK。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。若选择边沿触发模式，高或低的状态至少需要保持 4 个 PCLK，低于该值的脉冲将被忽略。

6.15.5.6 PWM 和 BPWM 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 和 BPWM 就可以作为 ADC 的触发源。当使能 PWM 触发 ADC 功能时，如果触发事件发生，PWM 将产生触发信号给 ADC，BPWM 有同样的功能。PWM 和 BPWM 触发事件，请参考相关章节。

6.15.5.7 比较监控转换结果

ADC 控制器提供两组比较寄存器 ADCMPRO 和 ADCMPRI，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 6-154。软件可通过设定 CMPCH(ADCMPO/1[5:3]) 来选择监控哪路通道，

而CMPCOND位 (ADCMPR0/1[2])则用于检查转换值小于或大于（等于）CMPD(ADCMPR0/1 [27:16])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)匹配时，CMF0/1 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。使用这个功能可以在扫描模式下，无需软件介入就可以监控外部模拟输入管脚电压变化。具体逻辑框图如下：

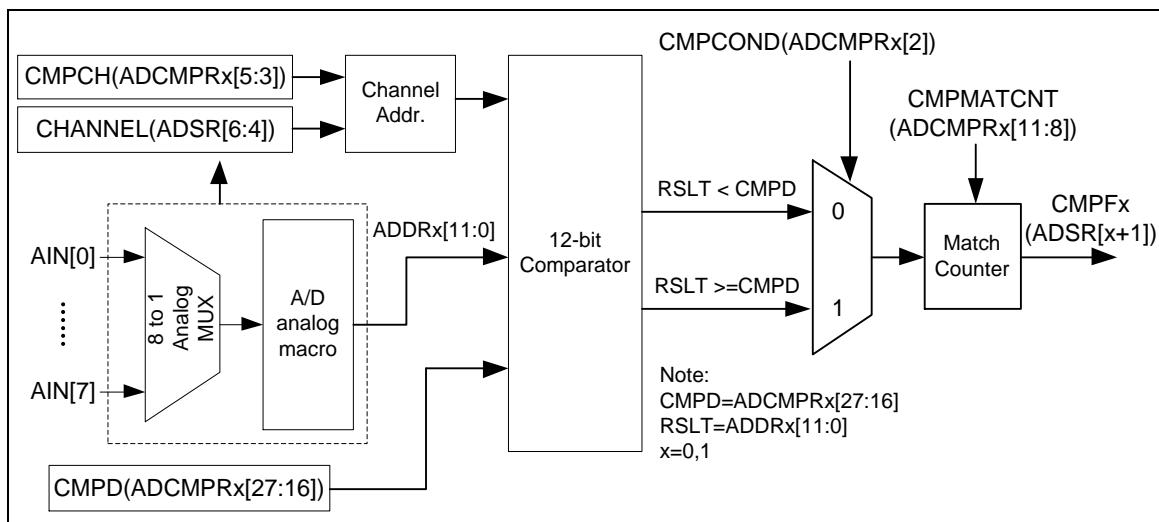


图 6-154 A/D 转换结果监控逻辑图

6.15.5.8 中断源

ADC中断有三个中断源。当某个ADC操作模式结束转换时，A/D转换的结束标志ADF位将被置1。CMF0 (ADSR[1]) 和CMF1 (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，相应的比较标志会被置1。当ADF(ADSR[0])、CMF0和CMF1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该中断请求。

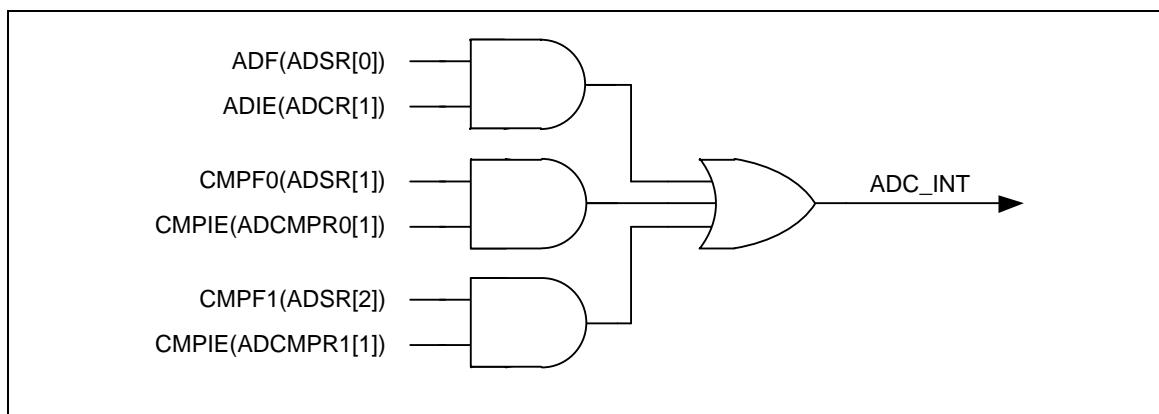


图 6-155 A/D 控制器中断



6.15.6 寄存器映射

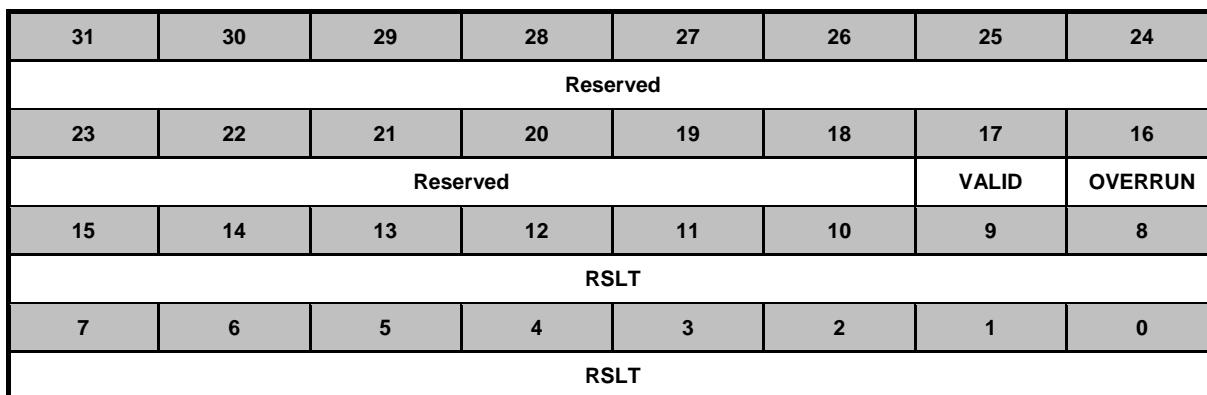
R: 只读, **W:** 只写, **R/W:** 读/写

寄存器	偏移地址	R/W	描述	复位值
ADC 基地址:				
ADC_BA = 0x400E_0000				
ADDR0	ADC_BA+0x00	R	ADC数据寄存器0	0x0000_0000
ADDR1	ADC_BA+0x04	R	ADC 数据寄存器 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	ADC 数据寄存器 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	ADC 数据寄存器3	0x0000_0000
ADDR4	ADC_BA+0x10	R	ADC 数据寄存器4	0x0000_0000
ADDR5	ADC_BA+0x14	R	ADC 数据寄存器 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	ADC 数据寄存器 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	ADC 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

6.15.7 寄存器描述

ADC 数据寄存器(ADDR0 ~ ADDR7)

寄存器	偏移量	R/W	描述	复位值
ADDR0	ADC_BA+0x00	R	ADC 数据寄存器 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	ADC 数据寄存器1	0x0000_0000
ADDR2	ADC_BA+0x08	R	ADC数据寄存器 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	ADC 数据寄存器3	0x0000_0000
ADDR4	ADC_BA+0x10	R	ADC 数据寄存器 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	ADC 数据寄存器 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	ADC 数据寄存器 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	ADC 数据寄存器 7	0x0000_0000



位	描述	
[31:18]	Reserved	保留
[17]	VALID	<p>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</p>
[16]	OVERRUN	<p>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寄存器之前，已经在RSLT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</p>

[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
--------	------	--

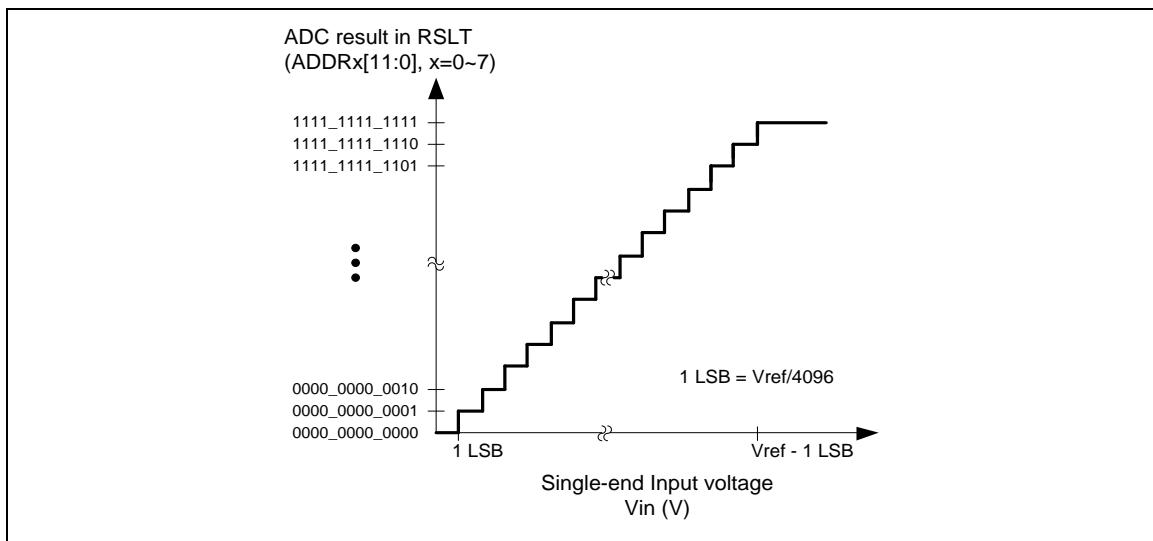


图 6-156 ADC 单端输入电压和转换结果图

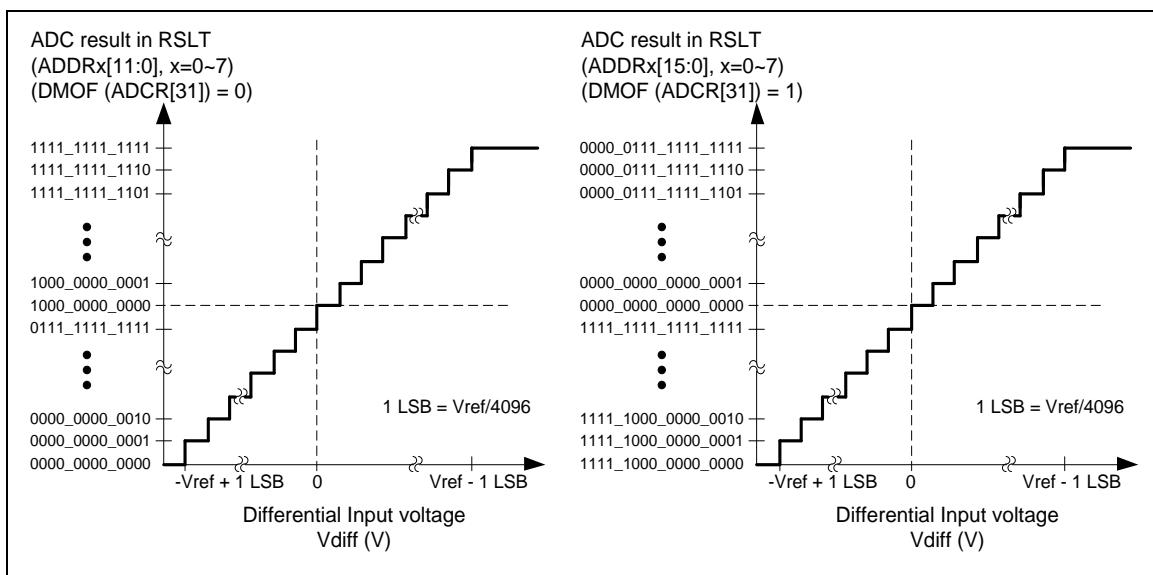


图 6-157 ADC 差分输入电压和转换结果图



ADC 控制寄存器 (ADCR)

寄存器	偏移量	R/W	描述				复位后的值
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	Reserved	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中心对齐触发和外部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	差分输入模式控制 0 = 单端模拟输入模式 1 = 差分模拟输入模式 <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td colspan="2" style="text-align: center;">差分输入对通道</td> <td colspan="2" style="text-align: center;">ADC 模拟输入</td> </tr> <tr> <td colspan="2"></td> <td style="text-align: center;">V_{plus}</td> <td style="text-align: center;">V_{minus}</td> </tr> <tr> <td colspan="2">0</td> <td style="text-align: center;">ADC0</td> <td style="text-align: center;">ADC1</td> </tr> <tr> <td colspan="2">1</td> <td style="text-align: center;">ADC2</td> <td style="text-align: center;">ADC3</td> </tr> <tr> <td colspan="2">2</td> <td style="text-align: center;">ADC4</td> <td style="text-align: center;">ADC5</td> </tr> <tr> <td colspan="2">3</td> <td style="text-align: center;">ADC6</td> <td style="text-align: center;">ADC7</td> </tr> </table>				差分输入对通道		ADC 模拟输入				V_{plus}	V_{minus}	0		ADC0	ADC1	1		ADC2	ADC3	2		ADC4	ADC5	3		ADC6	ADC7
差分输入对通道		ADC 模拟输入																											
		V_{plus}	V_{minus}																										
0		ADC0	ADC1																										
1		ADC2	ADC3																										
2		ADC4	ADC5																										
3		ADC6	ADC7																										
		差分输入电压(V_{diff}) = $V_{plus} - V_{minus}$. 其中 V_{plus} 是模拟正向输入， V_{minus} 是模拟负向输入。 在差分输入模式下，只需要在ADCHER使能两个相应信道的偶数信道即可。转换结果将放置于相应的使能通道的寄存器里																											



[9]	Reserved	保留
[8]	TRGEN	<p>硬件触发使能位</p> <p>使能或禁止通过硬件方式（外部STADC管脚或PWM 中心对齐触发）触发A/D转换</p> <p>0 = 禁止 1 = 使能</p> <p>ADC硬件触发功能 只能在单周期扫描模式下支持</p> <p>在硬件触发模式下，ADST 位 (ADCR[11])可被所选择的硬件触发源置1</p>
[7:6]	TRGCOND	<p>外部触发条件</p> <p>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</p> <p>00 = 低电平 01 = 高电平 10 = 下降沿 11 = 上升沿</p>
[5:4]	TRGS	<p>硬件触发源</p> <p>00 = A/D转换由外部STADC管脚触发开始 11 = A/D转换由PWM中心对齐触发开始 其它 = 保留</p> <p>改变TRGS前，软件必须先禁用TRGEN (ADCR[8])和ADST (ADCR[11])</p>
[3:2]	ADMD	<p>A/D转换器工作模式</p> <p>00 = 单一转换 01 = 保留 10 = 单周期扫描 11 = 连续扫描</p> <p>当改变工作模式时，软件须先禁止ADST位(ADCR[11])。</p>
[1]	ADIE	<p>A/D中断使能位</p> <p>0 = 禁止A/D中断 1 = 使能A/D中断</p> <p>如果ADIE 位 (ADCR[1])被置1，则A/D转换结束后将产生中断请求</p>
[0]	ADEN	<p>A/D转换器使能位</p> <p>0 = 禁止 1 = 使能</p> <p>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</p>

ADC 通道使能寄存器 (ADCHER)

寄存器	偏移量	R/W	描述				复位后的值
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	
7	6	5	4	3	2	1	0
CHEN							

位	描述	
[31:10]	Reserved	保留
[9:8]	PRESEL	<p>模拟输入通道7选择 00 = 外部模拟输入 01 = 内部带隙电压. 10 = 保留 11 = 保留</p>
[7:0]	CHEN	<p>模拟输入通道使能位 设置CHEN[7:0]用于使能相应的模拟输入通道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</p>



ADC比较寄存器0/1 (ADCMR0/1)

寄存器	偏移量	R/W	描述				复位后的值
ADCMR0	ADC_BA+0x28	R/W	ADC 比较寄存器 0				0x0000_0000
ADCMR1	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							
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位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 (ADCMR0/1[2])相匹配时，内部计数器将加1，在比较过程中，当比较数值和比较条件不匹配时，内部计数器将清0，当内部计数器达到设定值(CMPMATCNT (ADCMR0/1[1:8]) +1)时，将置CMPFO1 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 (ADCMR0/1[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(ADCMR0/1[27:16]), 内部匹配计数器将加1 注意： 当内部计数器达到(CMPMATICNT (ADCMR0/1[11:8])+1), CMPF0/1 (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(ADCMR0/1[2])和CMPMATICNT (ADCMR0/1[11:8])中的设定，则CMPF0/1位(ADSR[1]/[2])将会被置1，同时，如果CMPIE(ADCMR0/1[1])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 (ADCMR0/1[27:16])中数据进行比较。



ADC 状态寄存器 (ADSR)

寄存器	偏移量	R/W	描述				复位后的值
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OVERRUN 位 (ADDR0~7[16])的镜像 该位只读
[15:8]	VALID	数据有效标志 该位为VALID 位 (ADDR0~7[17])的镜像 该位只读
[7]	Reserved	保留
[6:4]	CHANNEL	当前转换通道 当BUSY(ADSR[3])=1时， 该域反应当前转换通道号。当BUSY=0时， 它表示下一个要转换的通道号 该位只读
[3]	BUSY	忙/空闲 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ADCMR1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ADCMR1 的设定值不匹配 1 = ADDR中的转换结果和ADCMR1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ADCMR0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ADCMR0 的设定值不匹配 1 = ADDR中的转换结果和ADCMR0的设定值匹配
[0]	ADF	A/D转换结束标志 该状态标志表明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

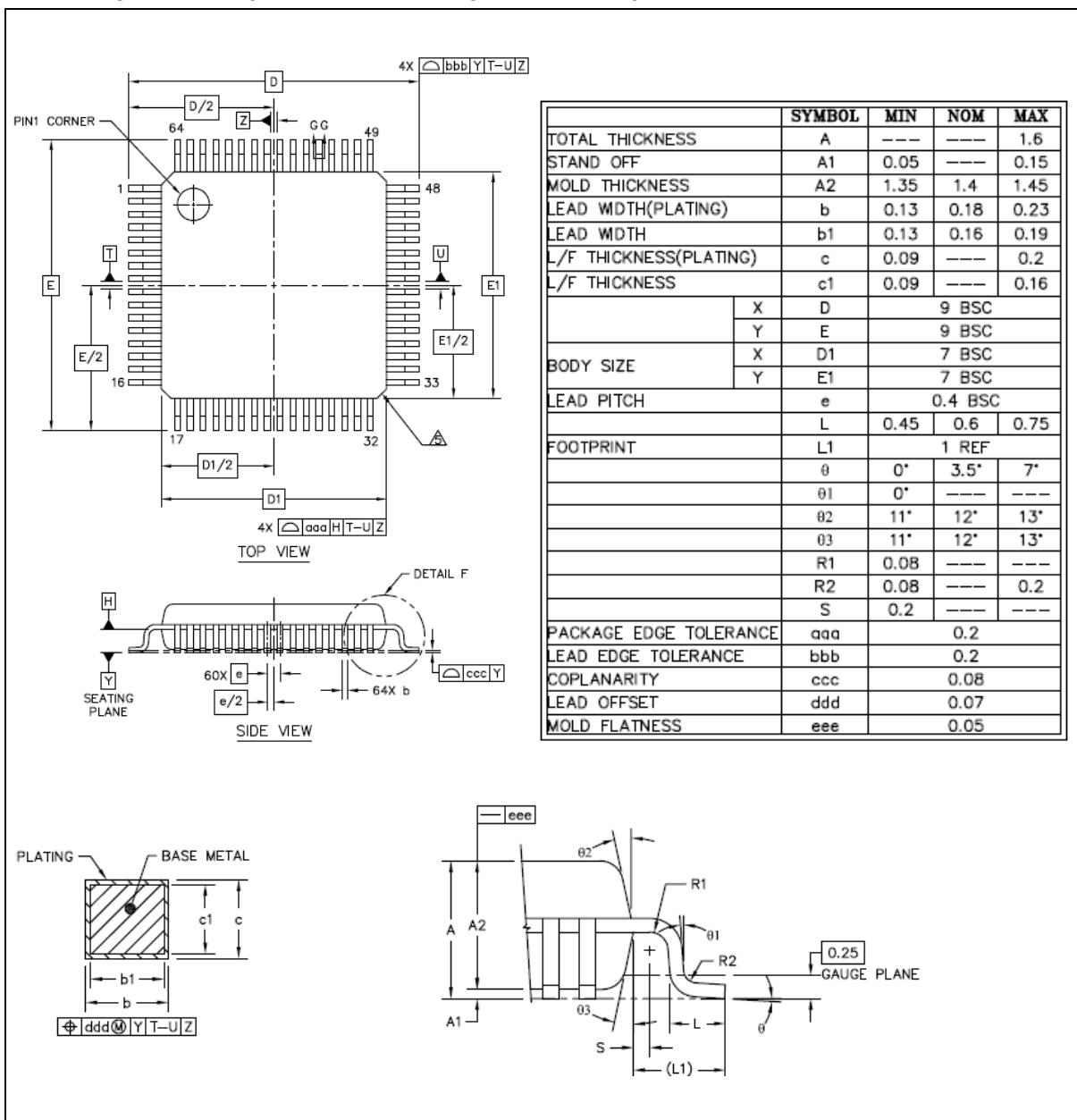


7 电器特性

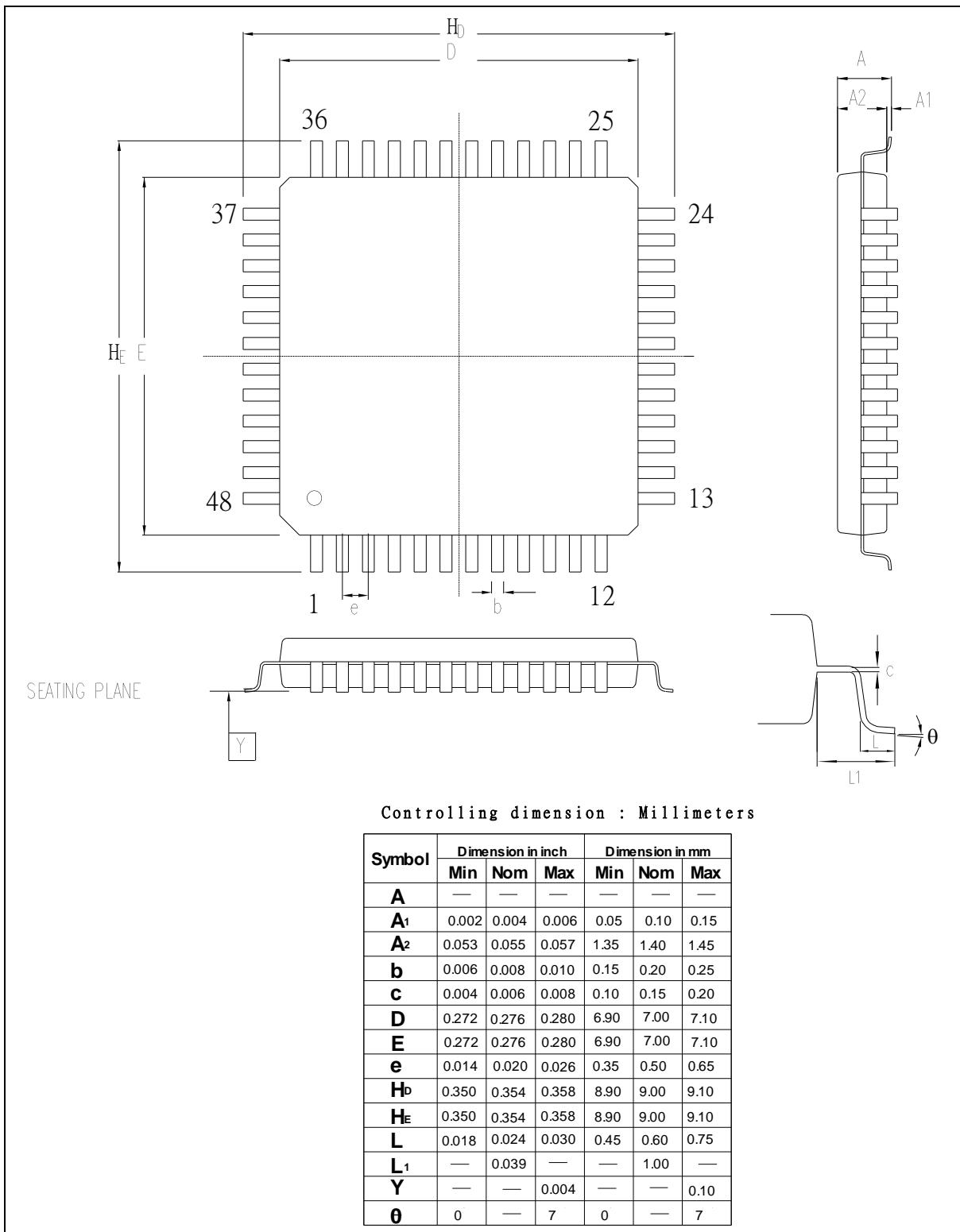
NUC131系列电气特性请参考NuMicro™ NUC131系列规格书

8 封装尺寸

8.1 64-pin LQFP (7x7x1.4 mm footprint 2.0 mm)



8.2 48-pin LQFP (7x7x1.4 mm footprint 2.0 mm)





9 修订历史

版本	日期	描述
1.00	12. 22, 2014	初版

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.

Please note that all data and specifications are subject to change without notice.
All the trademarks of products and companies mentioned in this datasheet belong to their respective owners.