



ARM Cortex®-M0

32位微控制器

# NuMicro™ M051 BN/DN/DE 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](http://www.nuvoton.com)

**目录**

<b>1 概述</b>	11
<b>2 特性</b>	12
<b>3 缩写</b>	16
<b>4 方块图</b>	18
<b>5 选型表和管脚图</b>	19
5.1 NuMicro™ M051 系列 M05xxBN 选型指南	19
5.2 NuMicro™ M051 系列 M05xxDN 选型指南	19
5.3 NuMicro™ M051 系列 M05xxDE 选型指南	20
5.4 管脚图	21
5.4.1 QFN 33-pin	21
5.4.2 LQFP 48-pin	22
5.5 管脚描述	23
<b>6 功能描述</b>	26
6.1 ARM® Cortex®-M0 内核	26
6.2 系统管理器	28
6.2.1 概述	28
6.2.2 系统复位	28
6.2.3 系统电源架构	29
6.2.4 系统存储器映射	30
6.2.5 系统存储器映射表	32
6.2.6 系统管理器控制寄存器映射	33
6.2.7 系统定时器(SysTick)	64
6.2.8 系统定时器控制寄存器映射	64
6.2.9 嵌套向量中断控制器 (NVIC)	68
6.2.10 系统控制块(SCB)	112
6.3 模拟比较器 (ACMP)	120
6.3.1 概述	120
6.3.2 特性	120
6.3.3 框图	121
6.3.4 基本配置	122
6.3.5 功能描述	123
6.3.6 寄存器映射	124
6.3.7 寄存器描述	125
6.4 模拟数字转换(ADC)	131
6.4.1 概述	131
6.4.2 特性	131
6.4.3 ADC 框图	132
6.4.4 基本配置	133
6.4.5 ADC 功能描述	133



6.4.6	ADC 寄存器映射 .....	141
6.4.7	ADC 寄存器描述 .....	142
6.5	时钟控制器 .....	153
6.5.1	概述 .....	153
6.5.2	系统时钟 & SysTick 时钟 .....	156
6.5.3	掉电模式（深度睡眠模式）时钟 .....	157
6.5.4	分频器输出 .....	157
6.5.5	时钟控制寄存器映射 .....	159
6.5.6	寄存器描述 .....	160
6.6	外部总线接口 (EBI) .....	179
6.6.1	概述 .....	179
6.6.2	特性 .....	179
6.6.3	EBI 框图 .....	180
6.6.4	基本配置 .....	181
6.6.5	功能描述 .....	181
6.6.6	寄存器映射 .....	186
6.6.7	寄存器描述 .....	187
6.7	Flash内存控制器(FMC) .....	191
6.7.1	概述 .....	191
6.7.2	特性 .....	191
6.7.3	FMC 框图 .....	192
6.7.4	FMC组织结构 .....	193
6.7.5	Data Flash .....	195
6.7.6	用户配置区 .....	196
6.7.7	启动选择 .....	198
6.7.8	在应用中编程 (IAP) .....	199
6.7.9	在系统编程(ISP) .....	200
6.7.10	ISP寄存器控制过程 .....	200
6.7.11	通过向量重新映射多引导 .....	202
6.7.12	寄存器映射 .....	205
6.7.13	寄存器描述 .....	206
6.8	通用 I/O (GPIO) .....	215
6.8.1	概述 .....	215
6.8.2	特性 .....	215
6.8.3	基本配置 .....	215
6.8.4	功能描述 .....	216
6.8.5	GPIO 中断和唤醒功能 .....	217
6.8.6	寄存器映射 .....	218
6.8.7	寄存器描述 .....	221
6.9	硬件除法器(HDIV) (只有M05xxDN/DE) .....	234
6.9.1	概述 .....	234
6.9.2	特性 .....	234
6.9.3	基本配置 .....	234



6.9.4	功能描述 .....	235
6.9.5	寄存器映射 .....	236
6.9.6	寄存器描述 .....	237
6.10	I <sup>2</sup> C 总线控制器 (主机/从机) .....	242
6.10.1	概述 .....	242
6.10.2	特性 .....	242
6.10.3	基本配置 .....	243
6.10.4	功能描述 .....	243
6.10.5	I <sup>2</sup> C 协议 .....	244
6.10.6	I <sup>2</sup> C 协议寄存器 .....	246
6.10.7	操作模式 .....	250
6.10.8	寄存器映射 .....	258
6.10.9	寄存器描述 .....	259
6.11	PWM发生器和捕捉定时器(PWM) .....	269
6.11.1	概述 .....	269
6.11.2	特性 .....	269
6.11.3	PWM 框图 .....	271
6.11.4	基本配置 .....	275
6.11.5	PWM 功能描述 .....	275
6.11.6	寄存器映射 .....	288
6.11.7	寄存器描述 .....	291
6.12	串行外设接口(SPI)控制器 .....	319
6.12.1	概述 .....	319
6.12.2	特性 .....	319
6.12.3	SPI 框图 .....	320
6.12.4	基本配置 .....	320
6.12.5	SPI 功能描述 .....	321
6.12.6	SPI 时序波形图 .....	330
6.12.7	SPI编程范例 .....	333
6.12.8	寄存器映射 .....	337
6.12.9	寄存器描述 .....	338
6.13	定时器控制器(TMR) .....	355
6.13.1	概述 .....	355
6.13.2	特性 .....	355
6.13.3	定时器方框图 .....	356
6.13.4	基本配置 .....	357
6.13.5	功能描述 .....	357
6.13.6	寄存器映射 .....	359
6.13.7	寄存器描述 .....	362
6.14	UART接口控制器(UART) .....	372
6.14.1	概述 .....	372
6.14.2	特性 .....	372
6.14.3	UART 框图 .....	373

6.14.4 基本配置 .....	375
6.14.5 功能描述 .....	375
6.14.6 寄存器映射 .....	389
6.14.7 寄存器描述 .....	390
<b>6.15 看门狗定时器 (WDT).....</b>	<b>412</b>
6.15.1 概述 .....	412
6.15.2 特性 .....	412
6.15.3 WDT 框图 .....	413
6.15.4 基本配置 .....	414
6.15.5 功能描述 .....	414
6.15.6 寄存器映射 .....	416
6.15.7 寄存器描述 .....	417
<b>6.16 窗看门狗 (WWDT) (M05xxDN/DE ) .....</b>	<b>420</b>
6.16.1 预览 .....	420
6.16.2 特性 .....	420
6.16.3 框图 .....	420
6.16.4 基本配置 .....	421
6.16.5 功能描述 .....	421
6.16.6 寄存器映射 .....	424
6.16.7 寄存器描述 .....	425
<b>7 电器特性.....</b>	<b>430</b>
<b>8 封装尺寸.....</b>	<b>430</b>
8.1 LQFP-48 (7x7x1.4mm <sup>2</sup> Footprint 2.0mm) .....	430
8.2 QFN-33 (5X5 mm <sup>2</sup> , Thickness 0.8mm, Pitch 0.5 mm) .....	431
<b>9 版本历史.....</b>	<b>432</b>

**LIST OF FIGURES**

图5-1 NuMicro™ 命名规则 .....	20
图5-2 NuMicro™ M051 系列QFN33 引脚图 .....	21
图5-3 NuMicro™ M051 系列 LQFP-48 管脚图 .....	22
图6-1 功能框图 .....	26
图6-2 NuMicro™ M051系列电源架构图 .....	29
图6-3 模拟比较器0/1框图 .....	121
图6-4 模拟比较器2/3 框图 (M05xxDN/DE Only) .....	122
图6-5 比较器中断源 .....	123
图6-6 比较器迟滞功能 .....	123
图6-7 ADC 控制器框图 .....	132
图6-8 ADC时钟控制 .....	133
图6-9 单次转换模式时序图 .....	135
图6-10 使能通道上单周期扫描模式时序图 .....	137
图6-11 使能通道的连续扫描时序图 .....	138
图6-12 A/D 转换结果监控框图 .....	139
图6-13 A/D 控制器中断 .....	140
图6-14 ADC单端输入转换电压和转换结果映射图 .....	143
图6-15 ADC差分输入转换电压和转换结果映射图 .....	143
图6-16 时钟发生器框图 .....	153
图6-17 时钟源控制器(1/2) .....	154
图6-18 时钟源控制器 (2/2) .....	155
图6-19 系统时钟框图 .....	156
图6-20 SysTick时钟控制框图 .....	156
图6-21 分频器的时钟源 .....	157
图6-22 分频器框图 .....	158
图6-23 EBI 框图 .....	180
图6-24 16位EBI数据宽度与16位器件连接 .....	181
图6-25 8位EBI数据宽度与8位设备连接 .....	182
图6-26 16位数据宽度的EBI时序控制波形 .....	183
图6-27 8位数据宽度EBI时序控制波形 .....	184
图6-28 插入空闲周期的EBI时序控制波形 .....	185
图6-29 Flash存储器控制器框图 .....	192
图6-30 Flash存储器组织结构 .....	194

图6-31 Flash存储器结构.....	195
图6-32 上电时启动选择(BS)从CONFIG0加载.....	198
图6-33 从APROM和LDROM启动时程序的执行范围.....	198
图6-34 IAP功能使能时可执行地址范围.....	200
图6-35 CBS[0] = 1时通过BS位改变启动选项.....	201
图6-36 ISP 软件编程流程.....	202
图6-37 通过向量重新映射多引导.....	204
图6-38 推挽输出.....	216
图6-39 开漏输出.....	216
图6-40 混双端I/O 模式.....	217
图6-41 硬件除法器操作流程.....	235
图6-42 I2C 总线时序.....	243
图6-43 I <sup>2</sup> C 协议.....	244
图6-44 START 和 STOP 条件.....	244
图6-45 I <sup>2</sup> C总线上的位传输.....	245
图6-46 I <sup>2</sup> C总线上的应答信号.....	245
图6-47 主机向从机传输数据.....	246
图6-48 主机读取从机的数据.....	246
图6-49 I <sup>2</sup> C 数据移动方向.....	247
图6-50 I2C 超时计数器框图.....	249
图6-51 根据当前I <sup>2</sup> C状态码控制I <sup>2</sup> C总线.....	250
图6-52 主机发送模式.....	251
图6-53 主机接收模式.....	252
图6-54 从机模式控制流程.....	253
图6-55 广播呼叫模式.....	255
图6-56 EEPROM 随机读操作.....	256
图6-57 EEPROM 随机读协议.....	257
图6-58 PWM发生器 0 时钟源控制.....	271
图6-59 PWM 发生器0结构框图 .....	271
图6-60 PWM 发生器 2 时钟源控制.....	272
图6-61 PWM 发生器 2结构框图 .....	272
图6-62 PWM 发生器 4 时钟源控制.....	273
图6-63 PWM 发生器 4 结构框图 .....	273

图6-64 PWM 发生器 6 时钟源控制 .....	274
图6-65 PWM 发生器 6 结构框图 .....	274
图6-66 PWM定时器内部比较器输出.....	275
图6-67 PWM定时器操作时序 .....	276
图6-68 PWM 周期中断发生时序图 .....	276
图6-69 中心对齐输出波形.....	277
图6-70 PWM 中心对齐中断发生时序图 .....	278
图6-71 PWM 双缓存图解.....	279
图6-72 PWM 控制器输出占空比 .....	279
图6-73 死区发生器操作 .....	280
图6-74 中心对齐模式下PWM 触发 ADC 标志 (PWMMxTF) 时序图 .....	281
图6-75 边沿对齐时PWM 触发ADC标志 (PWMMxTF) 时序图 .....	282
图6-75 中心对齐模式下PWM 触发ADC 开始转换 .....	283
图6-77 捕捉操作时序 .....	284
图6-78 PWM A组 PWM-定时器中断结构图.....	285
图6-79 PWM B组 PWM-定时器中断结构图.....	285
图6-80 SPI 框图.....	320
图6-81 SPI主机模式应用框图 .....	322
图6-82 SPI从机模式应用框图 .....	322
图6-83 一次传输32-Bit.....	323
图6-84 可变总线时钟频率 .....	324
图6-85 一次传输两笔 (Burst Mode) .....	325
图6-86 字节重排列.....	325
图6-87 字节休眠时序波形.....	326
图6-88 FIFO 模式框图 .....	327
图6-89 主机模式下SPI 时序 .....	330
图6-90 主机模式下SPI 时序(Alternate Phase of SPICLK) .....	331
图6-91 从机模式下SPI 时序 .....	332
图6-92 从机模式下SPI 时序(Alternate Phase of SPICLK) .....	333
图6-93 定时器控制器框图 .....	356
图6-94 定时器控制器时钟源 .....	356
图6-95 连续计数模式 .....	358
图6-96 Inter-Timer 触发捕获时序图 .....	359

图6-97 UART 时钟控制框图 .....	373
图6-98 UART 控制器框图 .....	373
图6-99 自动流控制框图 .....	375
图6-100 UART CTS 自动流控功能 .....	381
图6-101 UART RTS 自动流控使能 .....	382
图6-102 UART RTS 软件流控 .....	382
图6-103 IrDA 框图 .....	383
图6-104 IrDA TX/RX 时序框图 .....	384
图6-105 LIN 帧结构 .....	384
图6-106 自动方向控制模式下RS-485 RTS 驱动电平 .....	386
图6-107 软件控制RS-485 RTS 驱动电平 .....	387
图6-108 RS-485 帧结构 .....	388
图6-109 看门狗定时器时钟控制 .....	413
图6-110 看门狗定时器框图 .....	413
图6-111 看门狗超时中断间隔与复位周期时序 .....	415
图6-112 窗看门狗定时器时钟控制 .....	420
图6-113 窗看门狗定时器框图 .....	421
图6-114 窗看门狗定时器复位和重载行为 .....	422

**LIST OF TABLES**

表1-1 M05xxBN, M05xxDN 和 M05xxDE 比较表 .....	11
表3-1 缩写 .....	17
表5-1 NuMicro™ M051 系列引脚描述 .....	25
表6-1 片上模块的地址空间分配 .....	31
表6-2 异常模型 .....	69
表6-3 系统中断映射向量表 .....	70
表6-4 向量表格式 .....	70
表6-5 M05xxBN 和 M05xxDN/DE 比较器功能比较 (ACMP) .....	120
表6-6 掉电模式控制表 .....	162
表6-7 EBI 时序表 .....	183
表6-8 M05xxBN 和 M05xxDN/DE FMC功能比较表 .....	191
表6-9 Flash存储器地址映射 .....	193
表6-10 启动选项 .....	199
表6-11 ISP 命令列表 .....	202
表6-12 I <sup>2</sup> C 状态码描述表 .....	248
表6-13 M05xxBN 和 M05xxDN/DE 功能差异列表(SPI) .....	319
表6-14 M05xxBN中字节顺序和字节休眠条件 .....	326
表6-15 UART波特率方程表 .....	376
表6-16 UART波特率参数设置表 .....	376
表6-17 UART UART波特率寄存器设置表 .....	377
表6-18 M05xxBN 中UART 控制器中断源与标志列表 .....	378
表6-19 M05xxDN/DE中UART 控制器中断源与标志列表 .....	379
表6-20 UART 字和停止位长度线控 .....	380
表6-21 UART 校验位设定线控 .....	380
表6-22 看门狗超时间隔周期选择 .....	414
表6-23 窗看门狗定时器预分频值选择 .....	421
表6-24 WINCMP 设定限制 .....	423

## 1 概述

NuMicro™ M051 系列是以 ARM® Cortex®-M0 为内核的 32 位微控制器，应用于工业控制和需要丰富通信接口的领域。NuMicro™ M051 系列包括 M052xBN/xDN/xDE, M054xBN/xDN/xDE, M058xBN/xDN/xDE 和 M0516xBN/xDN/xDE.

NuMicro™ M051 系列运行频率最高可达 50MHz，工作电压 2.5V ~ 5.5V，工作温度 -40°C ~ 85°C，M05xxDE 更是高达 105°C (-40°C ~ 105°C)，因此 M051 系列可应用于各种各样的工业控制和需要高性能 CPU 的领域。NuMicro™ M051 系列内嵌有 8K/16K/32K/64K 字节的 flash 存储器，4K 字节数据 flash 存储器，4K 字节在系统编程 (ISP) 的 flash 存储器，及 4K 字节 SRAM 存储器。

许多系统级外设功能，如 I/O 端口、EBI (外部总线接口)、Timer、UART、SPI、I2C、PWM、ADC、看门狗定时器、窗看门狗(只有 M05xxDN/DE 支持)、模拟比较器和欠压检测，都已经被集成在 NuMicro™ M051 系列中，以减少系统外围元器件数量，节省电路板空间和系统成本。这些功能使 NuMicro™ M051 系列适用于广泛应用。

此外，NuMicro™ M051 系列带有 ISP (在系统编程) 和 ICP (在电路编程) 功能，以及 IAP (在应用中编程) (只有 M05xxDN/DE 支持) 允许用户无需取下芯片，直接在电路板上对程序存储器进行升级。

Item	M05xxBN	M05xxDN	M05xxDE
工作温度	-40°C ~ 85°C	-40°C ~ 85°C	-40°C ~ 105°C
硬件除法器	-	●	●
IAP模式	-	●	●
窗 WDT	-	●	●
模拟比较器	2	4	4
POR 以后 I/O 模式可配置	-	●	●
I <sup>2</sup> C	1	2 (支持唤醒)	2 (支持唤醒)
SPI	SPI 时钟源只能选择 HCLK 没有 FIFO	SPI 时钟源可以选择 HCLK 和 PLL 4-level FIFO	SPI 时钟源可以选择 HCLK 和 PLL 4-level FIFO
PWM and ADC	PWM 不能触发 ADC	PWM 可以触发 ADC 转换	PWM 可以触发 ADC 转换

表1-1 M05xxBN, M05xxDN 和 M05xxDE 比较表

## 2 特性

- 内核
  - ARM® Cortex®-M0内核运行频率可达50MHz.
  - 一个 24位系统定时器。
  - 支持低功耗睡眠模式.
  - 单指令周期32位硬件乘法器.
  - 嵌套向量中断控制器NVIC支持32个中断输入，每个中断有4个优先级。
  - 支持串行调试（SWD）接口，2 个观察点/4 个断点。
  - 提供硬件除法器(只有M05xxDN/DE 支持)，支持有符号32-bit 被除数和16-bit 除数
- 宽电压范围：2.5V~5.5V
- 存储器
  - 8KB/16KB/32KB/64KB Flash用于存储用户程序(APROM)
  - 4KB Flash用于存储数据(DataFlash)
  - 4KB Flash用于存储ISP引导代码 (LDROM)
  - 4KB字节SRAM用作内部高速暂存存储器(SRAM)
- 时钟控制
  - 可编程的系统时钟源
  - 22.1184MHz内部高速振荡器
  - 4~24 MHz外部高速晶振输入
  - 低功耗10KHz 的低速振荡器用于看门狗及睡眠模式唤醒CPU
  - PLL支持CPU最高运行在50MHz
- I/O 端口
  - 在LQFP-48管脚封装中，最多支持40个通用I/O端口（GPIO）
  - 4种I/O工作模式：
    - ◆ 准双向模式
    - ◆ 推挽输出模式
    - ◆ 开漏输出模式
    - ◆ 高阻抗输入模式
  - 可选择TTL输入或施密特触发输入
  - I/O管脚可被配置为边沿/电平触发模式的中断源
  - 较强的拉电流驱动能力和灌电流承受能力
- 定时器
  - 4组32位定时器，每组定时器均带有24位上数计数器和8位预分频器
  - 每个定时器有独立的时钟源

- 24位定时器当前值可由定时器数据寄存器（TDR）读出
- 提供3种工作模式：单脉冲模式，周期模式，翻转模式。
- 支持事件计数功能
- 提供外部信号捕获/复位计数器功能
- M05xxDN/DE额外支持的功能：
  - ◆ 多了两个定时器时钟源选择：外部触发引脚输入和内部10 kHz
  - ◆ TIMER 唤醒功能
  - ◆ 外部捕获输入源可以选择ACMP 或者 TxEX引脚
  - ◆ 翻转模式输入引脚可以选择TxEX 或者 TMx引脚
  - ◆ Inter-Timer 触发模式
- 看门狗定时器
  - 多路时钟源选择
  - 支持在掉电模式和休眠模式下唤醒CPU的功能
  - 看门狗定时器溢出时可选择产生中断/系统复位
  - 超时复位延迟周期可以选择(只有M05xxDN/DE支持)
- 窗看门狗 (只有M05xxDN/DE支持)
  - 6-bit 下数计数器，有 11-bit 预分频，可用于更大范围的窗选择
- PWM
  - 内建4个16位PWM发生器，提供8路PWM输出或4对互补的PWM输出
  - 每个PWM发生器可以单独选择时钟源，时钟分频器，8位时钟预分频器，和死区发生器
  - PWM中断与PWM周期同步
  - 16位捕捉定时器(与PWM定时器共享)支持捕获输入信号的上升沿/下降沿
  - 支持捕捉中断
  - M05xxDN/DE额外支持的功能：
    - ◆ PWM可以选择内部 10 kHz 作为时钟源
    - ◆ 极性翻转功能
    - ◆ 中心对齐
    - ◆ 使能定时器duty 中断功能
    - ◆ 两种PWM 中断周期类型选择
    - ◆ 两种PWM 中断duty 类型选择
    - ◆ 周期/duty 触发ADC 功能
    - ◆ PWM 定时器同步启动功能
- UART
  - 最多两组UART设备
  - 可编程波特率发生器
  - 接收器和发送器支持缓冲，均带有15bytes的FIFO缓冲
  - 流控功能供选择(CTS 和 RTS)
  - 支持 IrDA(SIR) 功能

- 支持RS485功能
- 支持LIN功能
- SPI
- 最高支持2组SPI设备
  - 支持 SPI主机/从机模式
  - 全双工同步串行数据传输
  - 从模式下，支持3线模式
  - 数据长度可改变（从1到32位）
  - 可设置MSB或LSB优先的传输模式
  - 接收可在串行时钟的上升/下降沿锁存数据
  - 发送可在串行时钟的上升/下降沿发送数据
  - 32位传输模式下支持字节挂起模式
  - M05xxDN/DE额外支持的功能：
    - ◆ 选择PLL 作为时钟源
    - ◆ 4-级 FIFO 缓冲，可以让SPI传输有更好的性能
- I<sup>2</sup>C
- I<sup>2</sup>C 模组
    - ◆ M05xxBN: 最多有1组
    - ◆ M05xxDN/DE: 最多有2组
  - 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）
  - 多主机同时发送数据时进行仲裁，总线上串行数据不会被损坏
  - 串行时钟同步使得不同比特率的设备可以通过一条串行总线传输数据
  - 串行时钟同步可用作握手机制来暂停和恢复串行传输
  - 可编程配置的时钟可适应多样化的传输速率控制.
  - 支持多地址识别 (4组从机地址带屏蔽选项)
- ADC
- 12位逐次逼近式模数转换器ADC
  - 最多8通道单端输入或者4通道差分输入
  - 支持单次转换模式/突发模式/单周期扫描模式/连续扫描模式
  - 差分模式下，支持2的补码/无符号格式的转换结果
  - 每个通道有独立的存放转换结果的寄存器
  - 支持转换结果监测（或比较），用于阈值电压检测
  - 转换开始可由软件或外部引脚触发

- M05xxDN/DE额外支持的功能功能:
  - ◆ A/D转换由PWM中心对齐触发或者边沿对齐触发
  - ◆ PWM 触发延迟功能
  - ◆ 差分输入和Burst模式下，支持换结果为有符号格式
- ACMP
  - 模拟比较器模块
    - ◆ M05xxBN: 最多2组
    - ◆ M05xxDN/DE: 最多4组
  - 负端可以选择内部带隙电压或者外部输入电压
  - 当比较结果发生改变时发生中断
  - 支持掉电模式下唤醒CPU的功能
- EBI (外部总线接口)，用于外部存储器映射设备的访问
  - 可访问的空间: 8位模式下为64KB, 16位模式下为128KB
  - 支持8bit或者16bit 数据宽度
  - 16bit数据宽度下，支持字节写功能
- ISP(在系统编程) 和 ICP(在电路编程)
- IAP(在应用编程) (只有M05xxDN/DE支持)
- 内嵌温度传感器，1°C分辨率
- 欠压检测(BOD)
  - 支持4级检测电压: 4.4V/3.7V/2.7V/2.2V
  - 支持欠压中断和复位选择
- 96-bit唯一ID号
- LVR (低电压复位)
  - 阈值电压: 2.0V
- 工作温度
  - M05xxBN: -40°C~85°C
  - M05xxDN: -40°C~85°C
  - M05xxDE: -40°C~105°C
- 封装:
  - 无铅封装 (RoHS)
  - 48-pin LQFP, 33-pin QFN



### 3 缩写

缩写	描述
ACMP	Analog Comparator Controller
ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
APB	Advanced Peripheral Bus
AHB	Advanced High-Performance Bus
BOD	Brown-out Detection
CAN	Controller Area Network
DAP	Debug Access Port
DES	Data Encryption Standard
EBI	External Bus Interface
EPWM	Enhanced Pulse Width Modulation
FIFO	First In, First Out
FMC	Flash Memory Controller
FPU	Floating-point Unit
GPIO	General-Purpose Input/Output
HCLK	The Clock of Advanced High-Performance Bus
HIRC	22.1184 MHz Internal High Speed RC Oscillator
HXT	4~24 MHz External High Speed Crystal Oscillator
IAP	In Application Programming
ICP	In Circuit Programming
ISP	In System Programming
LDO	Low Dropout Regulator
LIN	Local Interconnect Network
LIRC	10 kHz internal low speed RC oscillator (LIRC)
MPU	Memory Protection Unit
NVIC	Nested Vectored Interrupt Controller
PCLK	The Clock of Advanced Peripheral Bus
PDMA	Peripheral Direct Memory Access
PLL	Phase-Locked Loop
PWM	Pulse Width Modulation
QEI	Quadrature Encoder Interface
SDIO	Secure Digital Input/Output

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

表3-1 缩写

## 4 方块图

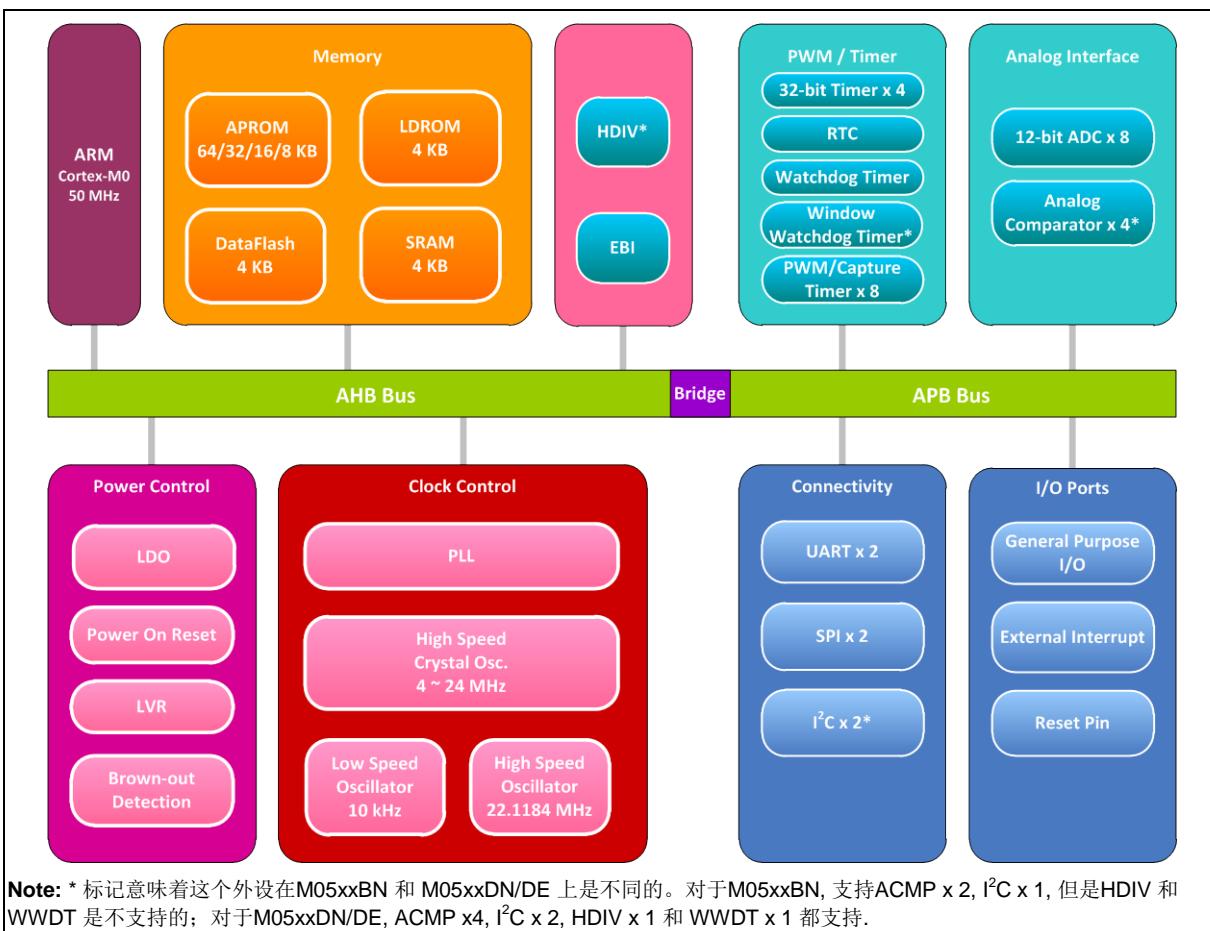


图 4-1 NuMicro™ M051 系列方块图

## 5 选型表和管脚图

### 5.1 NuMicro™ M051 系列 M05xxBN 选型指南

Part Number	APROM	RAM	Data Flash	LDROM	I/O	Timer	Connectivity			COMP	PWM	ADC	EBI	ISP ICP	Package
							UART	SPI	I²C						
M052LBN	8KB	4KB	4KB	4KB	40	4x32-bit	2	2	1	2	8	8X12-bit	v	v	LQFP48
M052ZBN	8KB	4KB	4KB	4KB	24	4x32-bit	2	1	1	2	5	5X12-bit		v	QFN33
M054LBN	16KB	4KB	4KB	4KB	40	4x32-bit	2	2	1	2	8	8X12-bit	v	v	LQFP48
M054ZBN	16KB	4KB	4KB	4KB	24	4x32-bit	2	1	1	2	5	5X12-bit		v	QFN33
M058LBN	32KB	4KB	4KB	4KB	40	4x32-bit	2	2	1	2	8	8X12-bit	v	v	LQFP48
M058ZBN	32KB	4KB	4KB	4KB	24	4x32-bit	2	1	1	2	5	5X12-bit		v	QFN33
M0516LBN	64KB	4KB	4KB	4KB	40	4x32-bit	2	2	1	2	8	8X12-bit	v	v	LQFP48
M0516ZBN	64KB	4KB	4KB	4KB	24	4x32-bit	2	1	1	2	5	5X12-bit		v	QFN33

表 5-1 NuMicro™ M051 系列 M05xxBN 产品选型指南

### 5.2 NuMicro™ M051 系列 M05xxDN 选型指南

Part Number	APROM	RAM	Data Flash	LDROM	I/O	Timer	Connectivity			COMP	PWM	ADC	EBI	ISP ICP IAP	Package
							UART	SPI	I²C						
M052LDN	8KB	4KB	4KB	4KB	40	4x32-bit	2	2	2	4	8	8X12-bit	v	v	LQFP48
M052ZDN	8KB	4KB	4KB	4KB	24	4x32-bit	2	1	2	4	5	5X12-bit		v	QFN33
M054LDN	16KB	4KB	4KB	4KB	40	4x32-bit	2	2	2	4	8	8X12-bit	v	v	LQFP48
M054ZDN	16KB	4KB	4KB	4KB	24	4x32-bit	2	1	2	4	5	5X12-bit		v	QFN33
M058LDN	32KB	4KB	4KB	4KB	40	4x32-bit	2	2	2	4	8	8X12-bit	v	v	LQFP48
M058ZDN	32KB	4KB	4KB	4KB	24	4x32-bit	2	1	2	4	5	5X12-bit		v	QFN33
M0516LDN	64KB	4KB	4KB	4KB	40	4x32-bit	2	2	2	4	8	8X12-bit	v	v	LQFP48
M0516ZDN	64KB	4KB	4KB	4KB	24	4x32-bit	2	1	2	4	5	5X12-bit		v	QFN33

表 5-2 NuMicro™ M051 系列 M05xxDN 选型指南

### 5.3 NuMicro™ M051 系列 M05xxDE 选型指南

Part Number	APROM	RAM	Data Flash	LDROM	I/O	Timer	Connectivity			COMP	PWM	ADC	EBI	ISP ICP IAP	Package
							UART	SPI	I <sup>2</sup> C						
M052LDE	8KB	4KB	4KB	4KB	40	4x32-bit	2	2	2	4	8	8X12-bit	v	v	LQFP48
M052ZDE	8KB	4KB	4KB	4KB	24	4x32-bit	2	1	2	4	5	5X12-bit		v	QFN33
M054LDE	16KB	4KB	4KB	4KB	40	4x32-bit	2	2	2	4	8	8X12-bit	v	v	LQFP48
M054ZDE	16KB	4KB	4KB	4KB	24	4x32-bit	2	1	2	4	5	5X12-bit		v	QFN33
M058LDE	32KB	4KB	4KB	4KB	40	4x32-bit	2	2	2	4	8	8x12-bit	v	v	LQFP48
M058ZDE	32KB	4KB	4KB	4KB	24	4x32-bit	2	1	2	4	5	5x12-bit		v	QFN33
M0516LDE	64KB	4KB	4KB	4KB	40	4x32-bit	2	2	2	4	8	8X12-bit	v	v	LQFP48
M0516ZDE	64KB	4KB	4KB	4KB	24	4x32-bit	2	1	2	4	5	5X12-bit		v	QFN33

表 5-3 NuMicro™ M051 系列 M05xxDE 选型指南

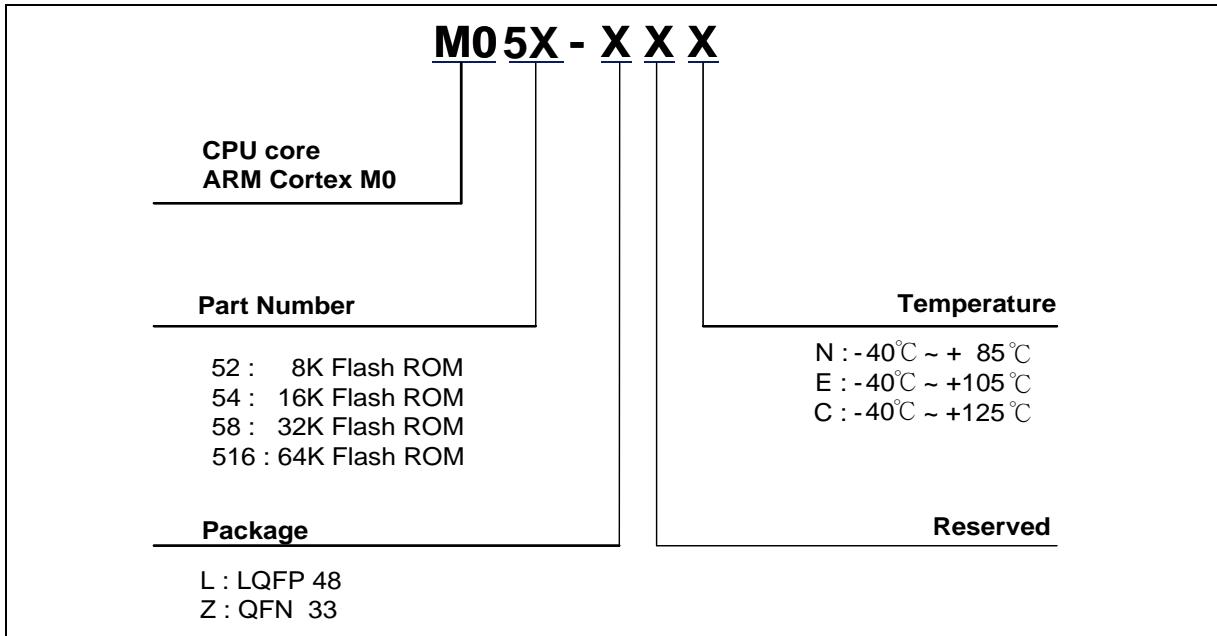


图5-1 NuMicro™ 命名规则

## 5.4 管脚图

### 5.4.1 QFN 33-pin

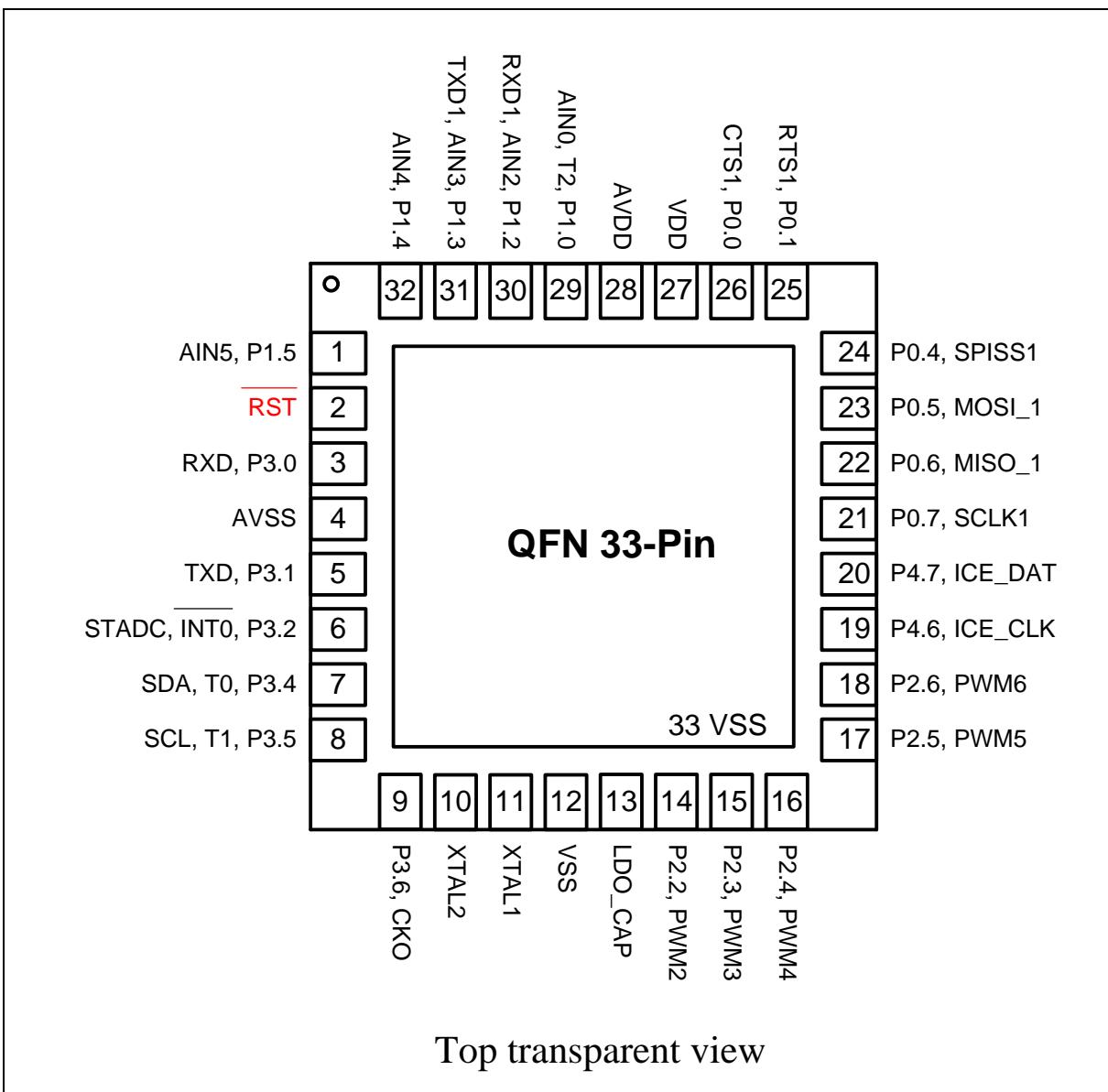


图5-2 NuMicro™ M051 系列QFN33 引脚图

## 5.4.2 LQFP 48-pin

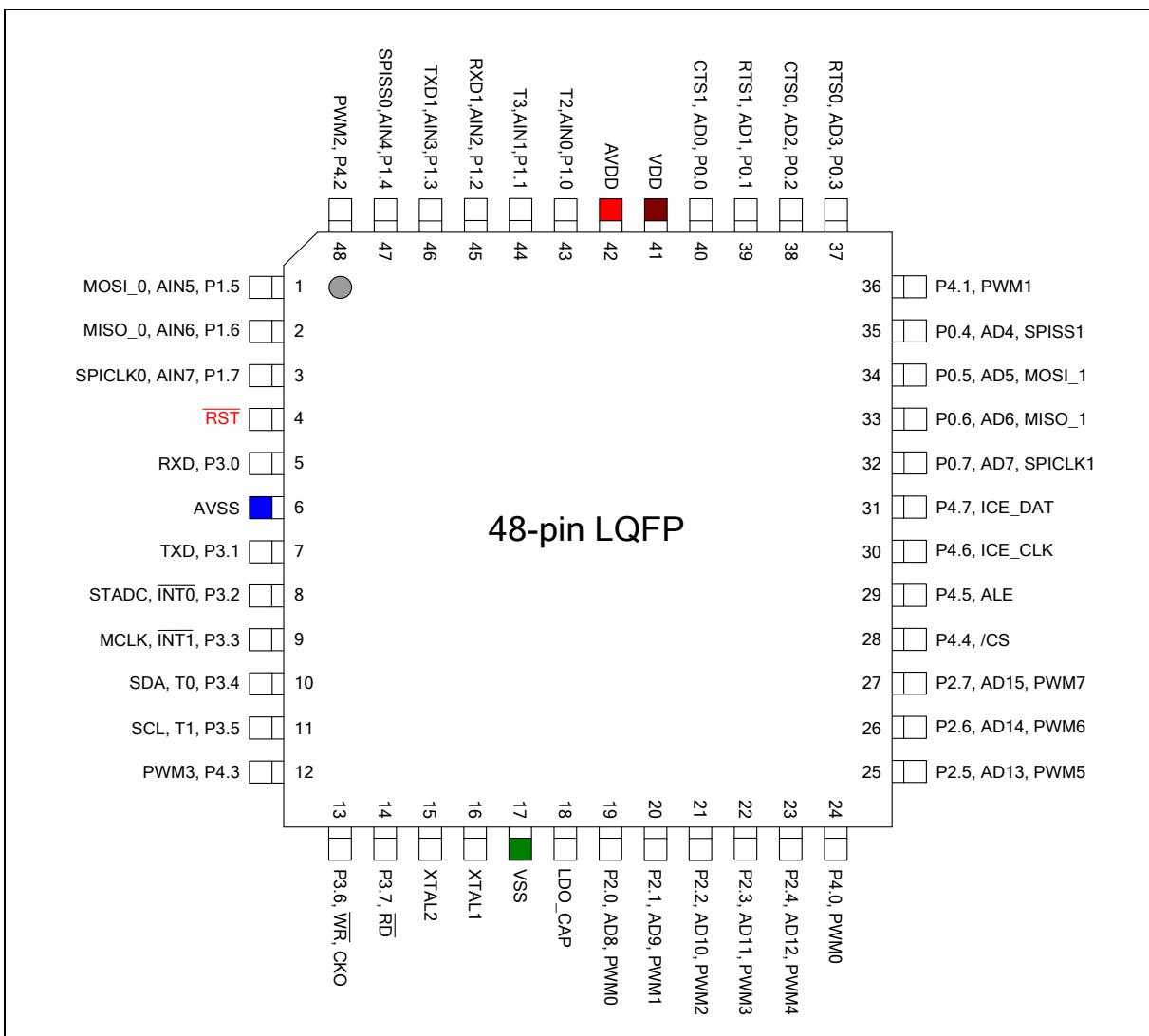


图5-3 NuMicro™ M051 系列 LQFP-48 管脚图

## 5.5 管脚描述

管脚号		符号	复用功能			类型 <sup>[1]</sup>	描述
QFN33	LQFP48		1	2	3		
11	16	XTAL1				I (ST)	外部4~24MHz高速晶振输入引脚
10	15	XTAL2				O	外部4~24MHz高速晶振输出引脚
27	41	VDD				P	电源输入脚：为内部PLL和数字电路以及I/O口和LDO供电。
12	17	VSS				P	数字电源地
33						P	
28	42	AVDD				P	模拟电路电源输入脚
4	6	AVSS				P	模拟电源地
13	18	LDO_CA_P				P	LDO 输出管脚 <b>注：必须外接 1uF 电容。</b>
2	4	nRST				I (ST)	复位脚：nRST管脚为施密特触发输入管脚，用于芯片复位。当该管脚上接入“低”电位，并保持768个内部22 MHz RC 高速晶振时钟周期后，芯片复位。nRST管脚具有内部上拉电阻，对该管脚通过外部电容接地，就可以完成上电复位。
26	40	P0.0	CTS1	AD0	TXD1 <sup>[2]</sup> , ACMP3_P	I/O	P0.0 ~ P0.7管脚被称为Port0，Port0为通用I/O口，可以被配置为输入、输出、准双向和开漏。 Port0为多功能复用引脚，包括CTS1, RTS1, CTS0, RTS0, SPISS1, MOSI_1, MISO_1, SPICLK1, AD0 ~ AD7, TXD1, RXD1, TXD, RXD, ACMP3_P, 和 ACMP3_N。
25	39	P0.1	RTS1	AD1	RXD1 <sup>[2]</sup> , ACMP3_N	I/O	AD0~AD7用于EBI功能，连接外部存储器。 SPISS1, MOSI_1, MISO_1, 和SPICLK1引脚用于SPI功能。
NC	38	P0.2	CTS0	AD2	TXD <sup>[2]</sup>	I/O	CTS0和CTS1引脚是UART0/1清除发送输入引脚
NC	37	P0.3	RTS0	AD3	RXD <sup>[2]</sup>	I/O	RTS0和RTS1是UART0/1请求发送输出引脚
24	35	P0.4	SPISS1	AD4		I/O	RXD和TXD 用于 UART0 功能
23	34	P0.5	MOSI_1	AD5		I/O	RXD1/TXD1用于UART1功能。
22	33	P0.6	MISO_1	AD6		I/O	ACMP3_N 和 ACMP3_P 引脚用于ACMP3 正端/负端输入
21	32	P0.7	SPISCLK1	AD7		I/O	
29	43	P1.0	T2	AIN0	nWRl	I/O	P1.0~P1.7引脚被称为Port1，Port1为通用I/O口，可以被配置为输入如、输出、准双向和开漏模式
NC	44	P1.1	T3	AIN1	nWRH	I/O	Port1为多功能复用引脚，包括T2, T3, RXD1, TXD1,

管脚号		符号	复用功能			类型 <sup>[1]</sup>	描述
QFN33	LQFP48		1	2	3		
30	45	P1.2	RXD1	AIN2		I/O	SPISS0, MOSI_0, MISO_0, SPICLK0, AIN0 ~ AIN7, nWRL, nWRH, ACMP0_N, ACMP0_P, ACMP2_N, 和 ACMP2_P.
31	46	P1.3	TXD1	AIN3		I/O	SPISS0, MOSI_0, MISO_0, 和SPICLK0引脚用于SPI功能。
32	47	P1.4	SPISS0	AIN4	ACMP0_N	I/O	AIN0~AIN7: 用于12位ADC的模拟信号输入脚
1	1	P1.5	MOSI_0	AIN5	ACMP0_P	I/O	RXD1/TXD1: 供UART1使用
NC	2	P1.6	MISO_0	AIN6	ACMP2_N	I/O	nWRL 和 nWRH: EBI模式下, 16-bit数据宽度时, 用于输出低/高字节写使能信号·
NC	3	P1.7	SPICLK0	AIN7	ACMP2_P	I/O	ACMP0_N 和 ACMP0_P: 用于比较器0的正/负端输入. ACMP2_N 和 ACMP2_P: 用于比较器2的正/负端输入. T2: Timer2的外部事件计数器输入管脚和翻转输出管脚 T3: Timer3的外部事件计数器输入管脚和翻转输出管脚
NC	19	P2.0	PWM0 <sup>[2]</sup>	AD8		I/O	P2.0~P2.7引脚被称为Port2, Port2为通用I/O口, 可以被配置为输入、输出、准双向和开漏模式
NC	20	P2.1	PWM1 <sup>[2]</sup>	AD9		I/O	Port2为多功能复用引脚, 包括PWM0 ~ PWM7, AD8 ~ AD15, SCL1, SDA1 和 ACMP1_O。
14	21	P2.2	PWM2 <sup>[2]</sup>	AD10		I/O	PWM0~PWM7, LQFP48封装时用于PWM输出功能
15	22	P2.3	PWM3 <sup>[2]</sup>	AD11		I/O	当外部总线接口（EBI）被使能时, AD8 ~ AD15可复用为EBI的AD[15:8]。
16	23	P2.4	PWM4	AD12	SCL1 <sup>[2]</sup>	I/O	SDA1 和 SCL1 引脚用于I <sup>2</sup> C1 , 都是开漏的.
17	25	P2.5	PWM5	AD13	SDA1 <sup>[2]</sup>	I/O	ACMP1_O: 比较器1的输出引脚
18	26	P2.6	PWM6	AD14	ACMP1_O	I/O	
NC	27	P2.7	PWM7	AD15		I/O	
3	5	P3.0	RXD <sup>[2]</sup>		ACMP1_N	I/O	P3.0~P3.7引脚被称为Port3, Port3为通用I/O口, 可以被配置为输入如、输出、准双向和开漏模式
5	7	P3.1	TXD <sup>[2]</sup>		ACMP1_P	I/O	Port3为多功能复用引脚, 包括RXD, TXD, nINT0, nINT1, T0, T1, nWR, nRD, STADC, MCLK, SDA, SCL, CKO, ACMP1_N, ACMP1_P, T0EX, T1EX, ACMP0_O。
6	8	P3.2	nINT0	STADC	T0EX	I/O	RXD/TXD: 供UART0使用
NC	9	P3.3	nINT1	MCLK	T1EX	I/O	INT0和INT1用于外部中断输入引脚
7	10	P3.4	T0	SDA		I/O	T0: Timer0的外部事件计数器输入管脚和翻转输出引脚
8	11	P3.5	T1	SCL	CKO <sup>[2]</sup>	I/O	T1: Timer1的外部事件计数器输入管脚和翻转输出引脚
9	13	P3.6	nWR	CKO	ACMP0_O	I/O	nWR, nRD 和 MCLK 用于 EBI 功能, MCLK 为 EBI 时钟输出脚
NC	14	P3.7	nRD			I/O	STADC: ADC 外部触发信号输入脚
							SDA/SCL: 供I2C功能使用, 都是开漏的
							CKO: 时钟监控功能 时钟输出引脚
							ACMP1_N 和 ACMP1_P: 比较器1的正/负端输入引脚.

管脚号		符号	复用功能			类型 <sup>[1]</sup>	描述
			1	2	3		
QFN33	LQFP48						T0EX/T1EX: Timer0/1的外部捕获/复位功能输入引脚. ACMP0_O: 比较器0的输出引脚
NC	24	P4.0	PWM0 <sup>[2]</sup>		T2EX	I/O	P4.0~P4.7引脚被称为Port4, Port4为通用I/O口, 可以被配置为输入如、输出、准双向和开漏模式
NC	36	P4.1	PWM1 <sup>[2]</sup>		T3EX	I/O	Port4为多功能复用引脚, 包括PWM0 ~ PWM3, nCS, ALE, ICE_CLK, ICE_DAT, SCL1, SDA1, T2EX 和 T3EX
NC	48	P4.2	PWM2 <sup>[2]</sup>			I/O	PWM0 ~ PWM3 引脚用于PWM 功能
NC	12	P4.3	PWM3 <sup>[2]</sup>			I/O	nCS 引脚为EBI 的片选信号脚。
NC	28	P4.4	nCS	SCL1		I/O	ALE (地址锁存使能脚) : 用于使能地址锁存, 在端口0和端口2上把地址从数据中分离出来。
NC	29	P4.5	ALE	SDA1		I/O	ICE_CLK/ICE_DAT: 用于JTAG仿真。
19	30	P4.6	ICE_CLK			I/O	SDA1 和 SCL1 引脚用于 I <sup>2</sup> C1 功能, 都是开漏的.
20	31	P4.7	ICE_DAT			I/O	T2EX/T3EX: Timer2/3的外部捕获/复位功能输入引脚.

表5-1 NuMicro™ M051 系列引脚描述

**注1:** I/O类型描述。I: 输入, O: 输出, I/O: 准双向, D: 开漏, P: 电源管脚, ST: Schmitt 触发器。

**注2:** PWM0 ~ PWM3, RXD, TXD, RXD1, TXD1, SCL1, SDA1 and CKO 可以被配置为不同的引脚, 然而某一时刻只有一个引脚可以配置为该功能, 例如: 软件不能同时分配 RXD 到 P0.3 和 P3.0.

## 6 功能描述

### 6.1 ARM® Cortex®-M0 内核

Cortex®-M0处理器是32位多级可配置的RISC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。该系列处理器支持两种操作模式Thread模式和Handler模式。当有异常发生时，处理器进入Handler模式。异常返回只能在Handler模式下发生。当复位时，处理器会进入Thread模式，处理器也可在异常返回时进入到Thread模式。下图显示了处理器内核的各个功能模块。

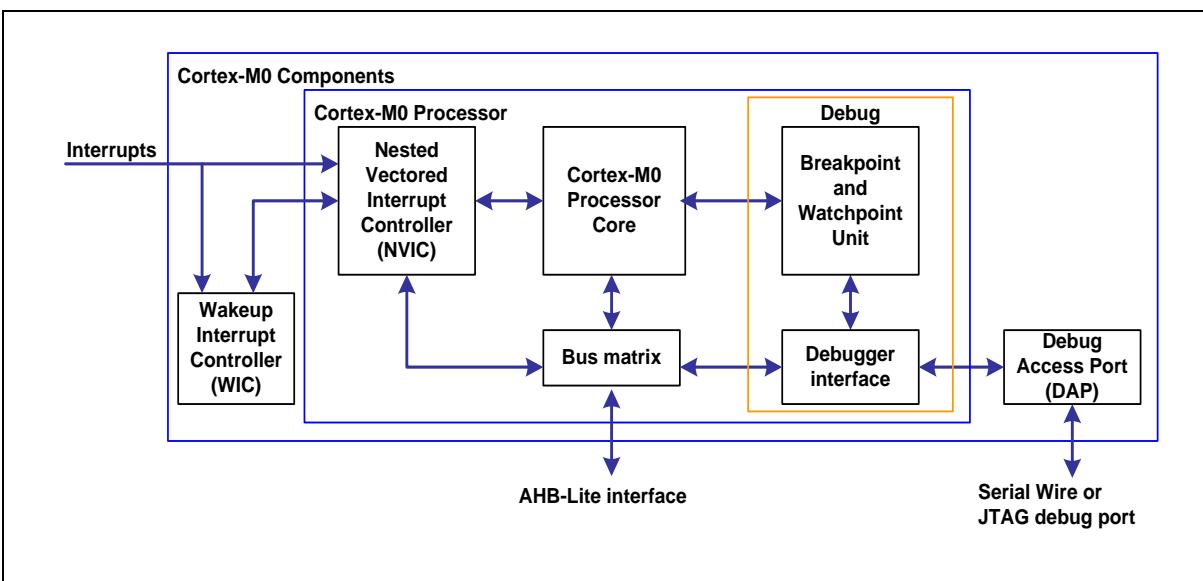


图6-1 功能框图

设备提供：

- 低门数处理器，特性如下：
  - ARMv6-M Thumb®指令集
  - Thumb-2 技术
  - ARMv6-M 兼容24-bit SysTick定时器
  - 32-bit 硬件乘法器
  - 系统接口支持小端（little-endian）数据访问
  - 具有确定性，固定延迟的中断处理能力
  - 可以丢弃和重新开始多次加载/存储和多周期乘法指令以保证快速中断处理
  - 与C应用程序二进制接口兼容的异常模式（C-ABI）
  - ARMv6-M（C-ABI）兼容异常模式允许用户使用纯C函数实现中断处理
  - 使用等待中断（WFI），等待事件（WFE）指令，或者从中断返回时直接进入睡眠的sleep-on-exit特性可以进入低功耗的休眠模式
- NVIC 特性：
  - 32 个外部中断输入，每个中断具有4级优先级
  - 不可屏蔽中断输入（NMI）
  - 支持电平和脉冲触发中断
  - 中断唤醒控制器(WIC)，支持极低功耗休眠模式



- 调试
  - 四个硬件断点
  - 两个观察点
  - 用于非侵入式代码的程序计数采样寄存器（PCSR）
  - 单步和向量捕获能力
- 总线接口:
  - 单一 32 位的 AMBA-3 AHB-Lite 系统接口，为所有的系统外设和存储器提供方便的集成。
  - 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

系统管理包括如下章节

- 系统复位
- 系统电源架构
- 系统存储器映射
- 用于产品ID, 芯片复位及片上模块复位, 多功能管脚控制的系统管理寄存器
- 系统定时器 (SysTick)
- 嵌套向量中断控制器(NVIC)
- 系统控制寄存器

### 6.2.2 系统复位

系统复位可以由如下事件发起, , 这些复位事件标志可以由寄存器**RSTRC**读出.

- 硬件复位
  - 上电复位(POR)
  - 复位脚 (nRST) 上有低电平
  - 看门狗定时溢出复位(WDT)
  - 低电压复位(LVR)
  - 欠压检测复位(BOD)
- 软件复位
  - MCU复位 - SYSRESETREQ(AIRCR[2])
  - CPU复位 - CPU\_RST(IPRSTC1[1])
  - 芯片复位 - CHIP\_RST(IPRSTC1[0])

注: MCU 复位和CPU复位之后, ISPCon.BS 的值不会从CONFIG0重新加载, 保持不变。.

### 6.2.3 系统电源架构

该芯片的电源架构分为2个部分：

- 来自AV<sub>DD</sub>和AV<sub>SS</sub>的模拟电源，为模拟部分提供工作电压。AV<sub>DD</sub>必须等于V<sub>DD</sub>以避免漏电
- 来自V<sub>DD</sub>和V<sub>SS</sub>的数字电源，为内部稳压器和I/O引脚提供电压，内部稳压器向数字操作提供固定的1.8V电压。

内部稳压器(LDO\_CAP)的输出，需要在相应管脚附近接一颗电容。模拟电源(AV<sub>DD</sub>)应该和数字电源(V<sub>DD</sub>)电压相同。下图显示了M051系列的电源分布：

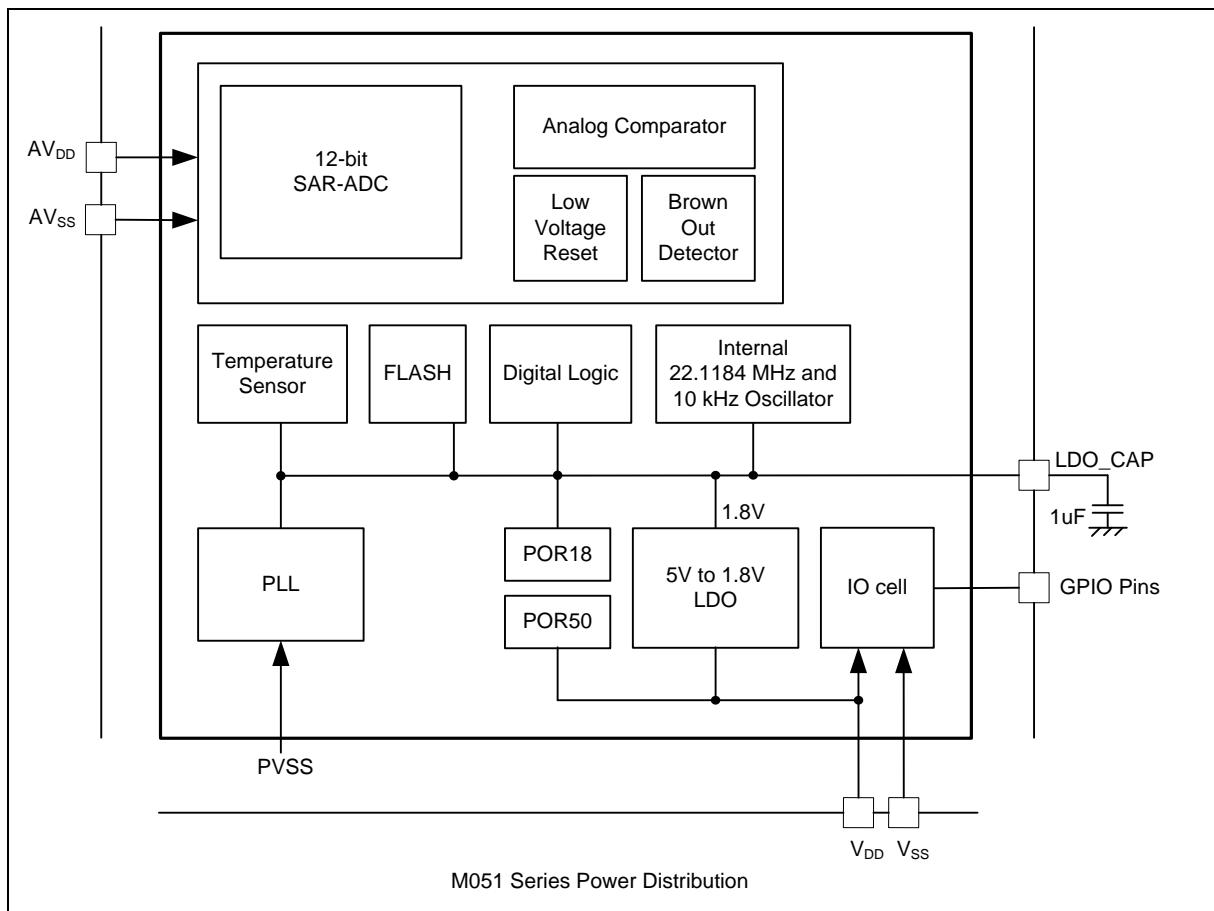


图6-2 NuMicro™ M051系列电源架构图

### 6.2.4 系统存储器映射

NuMicro™ M051系列提供4G字节的寻址空间。每个片上模块存储器地址分配情况如下表所示。详细的寄存器定义和寻址空间以及编程细节将在后续的各个片上外设描述章节里描述。NuMicro™ M051系列仅支持小端数据格式。

地址空间	标志	模块
<b>Flash &amp; SRAM 内存空间</b>		
0x0000_0000 – 0x0000_FFFF	FLASH_BA	FLASH内存空间(64KB)
0x2000_0000 – 0x2000_0FFF	SRAM_BA	SRAM内存空间(4KB)
<b>EBI 空间 (0x6000_0000 ~ 0x6001_FFFF)</b>		
0x6000_0000 – 0x6001_FFFF	EBI_BA	EBI 空间(128 KB)
<b>AHB模块空间(0x5000_0000 – 0x501F_FFFF)</b>		
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 (P0~P4) 控制寄存器
0x5000_C000 – 0x5000_FFFF	FMC_BA	Flash 存储器控制寄存器
0x5001_0000 – 0x5001_3FFF	EBI_CTL_BA	EBI 控制寄存器
0x5001_4000 – 0x5001_7FFF	HDIV_BA	硬件除法器(只有M05xxDN/DE 支持)
<b>APB模块空间(0x4000_0000 ~ 0x400F_FFFF)</b>		
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	I2C_BA	I2C接口控制寄存器
0x4003_0000 – 0x4003_3FFF	SPI0_BA	带主/从功能的SPI0控制寄存器
0x4003_4000 – 0x4003_7FFF	SPI1_BA	带主/从功能的SPI1 控制寄存器
0x4004_0000 – 0x4004_3FFF	PWMA_BA	PWM0/1/2/3 控制寄存器
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0控制寄存器
0x400D_0000 – 0x400D_3FFF	ACMP01_BA	模拟比较器0/1控制寄存器
0x400E_0000 – 0x400E_FFFF	ADC_BA	模数转换器(ADC)控制寄存器
0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3控制寄存器
0x4012_0000 – 0x4012_3FFF	I2C1_BA	I2C1接口控制寄存器(只有M05xxDN/DE 支持)
0x4014_0000 – 0x4014_3FFF	PWMB_BA	PWM4/5/6/7控制寄存器
0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 控制寄存器

0x401D_0000 – 0x401D_3FFF	ACMP23_BA	模拟比较器2/3控制寄存器(只有M05xxDN/DE 支持)
<b>系统控制空间 (0xE000_E000 ~ 0xE000_EFFF)</b>		
0xE000_E010 – 0xE000_E0FF	SCS_BA	系统 定时器控制寄存器
0xE000_E100 – 0xE000_ECFF	SCS_BA	外部中断控制器控制寄存器
0xE000_ED00 – 0xE000_ED8F	SCS_BA	系统控制块寄存器

表6-1 片上模块的地址空间分配

### 6.2.5 系统存储器映射表

M052/54/58/516		
4 GB	Reserved	0xFFFF_FFFF   0xE000_F000
	System Control	0xE000_EFFF   0xE000_E000
	Reserved	0xE000_DFFF   0x6002_0000
	EBI	0x6001_FFFF   0x6000_0000
	Reserved	0x5FFF_FFFF   0x5020_0000
	AHB	0x501F_FFFF   0x5000_0000
	Reserved	0x4FFF_FFFF   0x4020_0000
	APB	0x401F_FFFF   0x4000_0000
1 GB	Reserved	0x3FFF_FFFF   0x2000_1000
0.5 GB	4 KB SRAM (M052/M054/M058/M0516)	0x2000_0FFF   0x2000_0000
0 GB	Reserved	0x1FFF_FFFF   0x0001_0000
	64 KB on-chip Flash (M0516)	0x0000_FFFF
	32 KB on-chip Flash (M058)	0x0000_7FFF
	16 KB on-chip Flash (M054)	0x0000_3FFF
	8 KB on-chip Flash (M052)	0x0000_1FFF   0x0000_0000

**System Control**

System Control Block	0xE000_ED00	SCB_BA
External Interrupt Controller	0xE000_E100	NVIC_BA
System Timer Control	0xE000_E010	SYST_BA
System Control Space	0xE000_E000	SCS_BA

**AHB peripherals**

Hardw are Divider Control	0x5001_4000	HDIV_BA*
EBI Control	0x5001_0000	EBI_CTL_BA
FMC	0x5000_C000	FLASH_BA
GPIO Control	0x5000_4000	GPIO_BA
Interrupt Multiplexer Control	0x5000_0300	INT_BA
Clock Control	0x5000_0200	CLK_BA
System Global Control	0x5000_0000	GCR_BA

**APB peripherals**

ACMPB Control	0x401D_0000	ACMP23_BA*
UART1 Control	0x4015_0000	UART1_BA
PWM4/5/6/7 Control	0x4014_0000	PWMB_BA
I2C1 Control	0x4012_0000	I2C1_BA*
Timer2/Timer3 Control	0x4011_0000	TMR23_BA
ADC Control	0x400E_0000	ADC_BA
ACMPA Control	0x400D_0000	ACMP01_BA
UART0 Control	0x4005_0000	UART0_BA
PWM0/1/2/3 Control	0x4004_0000	PWMA_BA
SPI1 Control	0x4003_4000	SPI1_BA
SPI0 Control	0x4003_0000	SPI0_BA
I2C Control	0x4002_0000	I2C0_BA
Timer0/Timer1 Control	0x4001_0000	TMR01_BA
WDT Control	0x4000_4000	WDT_BA
WWDT Control	0x4000_4100	WWDT_BA*

注: \*表示仅M05xxDN/DE支持.

### 6.2.6 系统管理器控制寄存器映射

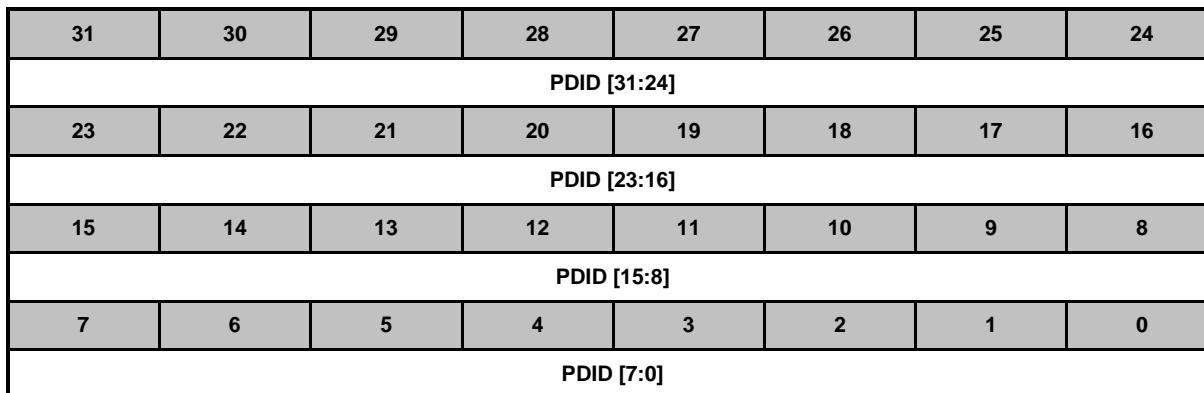
寄存器	偏移量	R/W	描述	复位后的值
<b>GCR Base Address:</b>				
<b>GCR_BA = 0x5000_0000</b>				
PDID	GCR_BA+0x00	R	设备ID寄存器	0x1000_5200
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX
IPRSTC1	GCR_BA+0x08	R/W	外设复位控制寄存器1	0x0000_0000
IPRSTC2	GCR_BA+0x0C	R/W	外设复位控制寄存器2	0x0000_0000
BODCR	GCR_BA+0x18	R/W	欠压检测控制寄存器	0x0000_008X
TEMPCR	GCR_BA+0x1C	R/W	温度传感器控制	0x0000_0000
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_00xx
P0_MFP	GCR_BA+0x30	R/W	P0 复用功能和输入类型控制寄存器	0x0000_0000
P1_MFP	GCR_BA+0x34	R/W	P1 复用功能和输入类型控制寄存器	0x0000_0000
P2_MFP	GCR_BA+0x38	R/W	P2 复用功能和输入类型控制寄存器	0x0000_0000
P3_MFP	GCR_BA+0x3C	R/W	P3 复用功能和输入类型控制寄存器	0x0000_0000
P4_MFP	GCR_BA+0x40	R/W	P4 输入类型控制寄存器	0x0000_00C0
REGWRPROT	GCR_BA+0x100	R/W	寄存器写保护控制寄存器	0x0000_0000



### 设备ID寄存器(PDID)

寄存器	偏移量	R/W	描述	复位后的值
PDID	GCR_BA+0x00	R	设备ID寄存器	0x1000_5200 <sup>[1]</sup>

[1]每个型号的设备复位后都有一个唯一的默认ID.



Bits	描述	
[31:0]	PDID	产品设备识别码. 该寄存器反映器件的识别码。S/W可以读该寄存器识别所使用的器件。 例如, M052LBN PDID 的识别码是0x1000_5200。

NuMicro™ M051 系列	Part Device Identification Number
M052LBN	0x10005200
M054LBN	0x10005400
M058LBN	0x10005800
M0516LBN	0x10005A00
M052ZBN	0x10005203
M054ZBN	0x10005403
M058ZBN	0x10005803
M0516ZBN	0x10005A03
M052LDN	0x20005200
M054LDN	0x20005400
M058LDN	0x20005800
M0516LDN	0x20005A00
M052ZDN	0x20005203

M054ZDN	0x20005403
M058ZDN	0x20005803
M0516ZDN	0x20005A03
M052LDE	0x30005200
M054LDE	0x30005400
M058LDE	0x30005800
M0516LDE	0x30005A00
M052ZDE	0x30005203
M054ZDE	0x30005403
M058ZDE	0x30005803
M0516ZDE	0x30005A03

**系统复位源寄存器(RSTSRC)**

该寄存器提供具体的信息给软件用于识别当前引起芯片复位的复位源。

寄存器	偏移量	R/W	描述	复位后的值
RSTSRC	GCR_BA+04	R/W	系统复位源寄存器	0x0000_00XX

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
RSTS_CPU	RSTS_PMU	RSTS MCU	RSTS_BOD	RSTS_LVR	RSTS_WDT	RSTS_RESET	RSTS_POR

Bits	描述	
[31:8]	保留	保留
[7]	RSTS_CPU	<b>CPU复位标志</b> 如果软件将 <b>CPU_RST</b> (IPRSTCR1[1]) 置为“1”导致Cortex-M0 CPU 内核和FLASH控制器(FMC) 复位， <b>RSTS_CPU</b> 标志将由硬件置位。 1= 软件置CPU_RST为1时，导致Cortex-M0 CPU 内核与FMC复位。 0= CPU无复位 注：向该位写1清除为零。
[6]	保留	保留
[5]	RSTS MCU	<b>MCU复位标志</b> <b>RSTS MCU</b> 由来自MCU Cortex_M0 核的“复位信号”置位，以表示当前的复位源。 1= MCU Cortex_M0 在软件向SYSRESTREQ(AIRCR[2]写1时，发出复位信号以复位系统 0= MCU无复位 注：向该位写1清除为零。
[4]	RSTS_BOD	<b>电压检测复位标志</b> <b>RSTS_BOD</b> 标志位由欠压检测模块的“复位信号”置1，用于表示当前复位源。 1: 欠压检测模块发出复位信号使系统复位。 0: BOD无复位 注：向该位写1清除为零。

[3]	<b>RSTS_LVR</b>	<p><b>低压检测复位标志</b></p> <p><b>RSTS_LVR</b>标志位由低压复位模块的“复位信号”置1，用于表示当前复位源。</p> <p>1: 低压 LVR 模块发出复位信号使系统复位.</p> <p>0: LVR无复位</p> <p>注：向该位写1清除为零.</p>
[2]	<b>RSTS_WDT</b>	<p><b>看门狗复位标志</b></p> <p><b>RSTS_WDT</b> 标志位由看门狗模块的“复位信号”置1，用于说明当前复位源。</p> <p>1: 看门狗模块发出复位信号使系统复位.</p> <p>0: 没有看门狗复位信号</p> <p>注：向该位写1清除为零</p>
[1]	<b>RSTS_RESET</b>	<p><b>复位引脚复位标志</b></p> <p><b>RSTS_RESET</b>标志位由nRST脚的“复位信号”置1，用于说明当前复位源。</p> <p>1: nRST脚上发出复位信号使系统复位.</p> <p>0: nRST脚没有复位信号</p> <p>注：向该位写1清除为零.</p>
[0]	<b>RSTS_POR</b>	<p><b>上电复位标志</b></p> <p><b>RSTS_POR</b>标志位由POR模块的“复位信号”或者写CHIP_RST (IPRSTC1[0])置1，用于说明当前的复位源。</p> <p>1: 上电复位POR发出复位信号或者CHIP_RST置1使系统复位.</p> <p>0: 没有POR复位信号或者CHIP_RST没有被写1</p> <p>注：向该位写1清除为零.</p>

外设复位控制寄存器1 (IPRSTC1)

寄存器	偏移量	R/W	描述	复位后的值
IPRSTC1	GCR_BA+08	R/W	外设复位控制寄存器1	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			HDIV_RST	EBI_RST	保留	CPU_RST	CHIP_RST

Bits	描述	
[31:5]	保留	保留
[4]	HDIV_RST	<p><b>HDIV 控制器复位 (写保护) (只有M05xxDN/DE 支持)</b>            将该位置为1将产生一个复位信号送到硬件除法器。用户需要再将该位置0才能退出复位状态            0 = 硬件除法器正常工作            1 = 复位硬件除法器。  <b>注:</b> 该位是写保护的, 程序需要依次向地址0x5000_0100写入下列数据 “59h”, “16h”, and “88h” 来关闭保护功能. 参考寄存器REGWRPROT, 地址 GCR_BA+0x100</p>
[3]	EBI_RST	<p><b>EBI 控制器复位 (写保护)</b>            设置该位为“1”将产生复位信号到EBI。用户需要再将该位置“0”才能退出复位状态            0= EBI 控制器正常工作            1= EBI 控制器复位  <b>注:</b> 该位是写保护的, 修改该位时, 需要依次向0x5000_0100写入“59h”, “16h”, “88h”解除寄存器保护。参考寄存器 REGWRPROT, 地址GCR_BA + 0x100.</p>
[2]	保留	保留
[1]	CPU_RST	<p><b>CPU内核复位 (写保护)</b>            该位置1, CPU内核和Flash存储控制器复位。两个时钟周期后, 该位自动清零。            0 = 正常            1 = 复位CPU  <b>注:</b> 该位是写保护的, 修改该位时, 需要依次向0x5000_0100写入“59h”, “16h”, “88h”解除寄存器保护。参考寄存器 REGWRPROT, 地址GCR_BA + 0x100</p>

[0]	<b>CHIP_RST</b>	<p><b>芯片复位（写保护）</b></p> <p>该位置1，整个芯片复位，包括CPU内核和所有外设均复位。, 两个时钟周期后，该位自动清零。 CHIP_RST与上电复位一样，所有的芯片模块都复位，芯片缺省设置从CONFIG0/CONFIG1重新加载 0：正常 1：复位芯片 <b>注：</b>该位是写保护的，修改该位时，需要依次向0x5000_0100写入“59h”，“16h”，“88h”解除寄存器保护。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”这些位将会产生异步复位信号给相应的IP。用户需要清零相应位来使IP离开复位状态。

寄存器	偏移量	R/W	描述	复位后的值
IPRST2	GCR_BA+0C	R/W	外设复位控制寄存器2	0x0000_0000

31	30	29	28	27	26	25	24
保留			ADC_RST	保留			
23	22	21	20	19	18	17	16
ACMP23_RST	ACMP01_RST	PWM47_RST	PWM03_RST	保留		UART1_RST	UART0_RST
15	14	13	12	11	10	9	8
保留		SPI1_RST	SPI0_RST	保留		I2C1_RST	I2C0_RST
7	6	5	4	3	2	1	0
保留		TMR3_RST	TMR2_RST	TMR1_RST	TMR0_RST	GPIO_RST	保留

Bits	描述	
[31:29]	保留	保留
[28]	ADC_RST	ADC控制器复位 “0”: ADC 模块正常工作 “1”: ADC 模块复位
[27:24]	保留	保留
[23]	ACMP23_RST	模拟比较器B控制器复位(只有M05xxDN/DE 支持) 0 = 模拟比较器B控制器正常工作 1 = 模拟比较器B控制器复位
[22]	ACMP01_RST	模拟比较器A控制器复位 0 = 模拟比较器A控制器正常工作 1 = 模拟比较器A控制器复位
[21]	PWM47_RST	PWM4~7 控制器复位 0= PWM4~7 控制器正常工作 1= PWM4~7 控制器复位
[20]	PWM03_RST	PWM0~3 控制器复位 0= PWM0~3 控制器正常工作 1= PWM0~3 控制器复位
[19:18]	保留	保留
[17]	UART1_RST	UART1控制器复位 0= UART1 正常工作 1= UART1 控制器复位

[16]	<b>UART0_RST</b>	<b>UART0控制器复位</b> 0= UART0 正常工作 1= UART0 控制器复位
[15:14]	保留	保留
[13]	<b>SPI1_RST</b>	<b>SPI1控制器复位</b> 0= SPI1 正常工作 1= SPI1 控制器复位
[12]	<b>SPI0_RST</b>	<b>SPI0控制器复位</b> 0= SPI0 正常工作 1= SPI0 控制器复位
[11:10]	保留	保留
[9]	<b>I2C1_RST</b>	<b>I2C1控制器复位 (仅M05xxDN/DE支持)</b> 0= I2C正常工作 1= I2C 控制器复位
[8]	<b>I2C0_RST</b>	<b>I2C0控制器复位</b> 0= I2C0正常工作 1= I2C0控制器复位
[7:6]	保留	保留
[5]	<b>TMR3_RST</b>	<b>Timer3控制器复位</b> 0= Timer3 正常工作 1= Timer3 控制器复位
[4]	<b>TMR2_RST</b>	<b>Timer2控制器复位</b> 0= Timer2 正常工作 1= Timer2 控制器复位
[3]	<b>TMR1_RST</b>	<b>Timer1控制器复位</b> 0= Timer1 正常工作 1= Timer1 控制器复位
[2]	<b>TMR0_RST</b>	<b>Timer0控制器复位</b> 0= Timer0 正常工作 1= Timer0 控制器复位
[1]	<b>GPIO_RST</b>	<b>GPIO (P0~P4) 控制器复位</b> 0= GPIO 正常工作 1= GPIO 控制器复位
[0]	保留	保留

欠压检测控制寄存器(BODCR)

BODCR 控制寄存器的部分值由flash配置区域（CONFIG0/CONFIG1）初始化并且是写保护的，编程这些被保护的位需要依次向地址0x5000\_0100写入“59h”，“16h”，“88h”，禁用寄存器保护。参考寄存器REGWRPROT，其地址为GCR\_BA+0x100

寄存器	偏移量	R/W	描述				复位后的值
BODCR	GCR_BA+18	R/W	欠压检测控制寄存器				0x0000_008X

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
LVR_EN	BOD_OUT	BOD_LPM	BOD_INTF	BOD_RSTEN	BOD_VL		BOD_EN

Bits	描述	
[31:8]	保留	保留
[7]	LVR_EN	<p><b>低压复位使能（写保护位）</b> 输入电源电压低于LVR电路设置时，LVR复位整个芯片。默认配置下LVR复位是使能的，LVR复位电压请参考后面LVR特性 1= 使能低电压复位功能，使能该位100US后，LVR功能生效(默认) 0= 禁用低电压复位功能 <b>注：</b>该位是写保护的，修改该位时，需要依次向0x5000_0100写入“59h”，“16h”，“88h”解除寄存器保护。参考寄存器 REGWRPROT，地址GCR_BA + 0x100</p>
[6]	BOD_OUT	<p><b>欠压检测输出的状态位</b> 1 = 欠压检测输出状态为 1，表示检测到的电压低于 BOD_VL 设置。若 BOD_EN 是“0”，BOD 功能被关闭，该位始终保持为 “0” 0 = 欠压检测输出状态为0，表示检测到的电压高于BOD_VL设置</p>
[5]	BOD_LPM	<p><b>欠压检测低功耗模式（写保护位）</b> 1 = 使能 BOD 低功耗模式 0 = BOD 工作于正常模式(默认) <b>注1：</b> BOD 在正常模式下消耗电流约为100uA, 低功耗模式下减少到当前的1/10, 但是BOD响应速度变慢。 <b>注2：</b> 该位是写保护的，修改该位时，需要依次向0x5000_0100写入“59h”，“16h”，“88h”解除寄存器保护。参考寄存器 REGWRPROT，地址GCR_BA + 0x100</p>
[4]	BOD_INTF	<p><b>欠压检测中断标志</b> 1= 欠压检测到V<sub>DD</sub>下降到低于BOD_VL 的设定电压或V<sub>DD</sub>上升高于BOD_VL 的设定电压，该位被置为1，如果欠压中断被使能，则发生欠压中断。</p>

		0= 没有检测到任何电压由V <sub>DD</sub> 下降或上升至BOD_VL 设定值。 注：该位写1清除为0															
[3]	BOD_RSTEN	<p><b>欠压复位使能（写保护位）</b></p> <p>1 = 使能欠压复位功能，当欠压检测功能使能后，检测到电压低于门槛电压，芯片发生复位 0 = 使能欠压中断功能</p> <p>当BOD_EN使能(BOD_EN = 1)，且中断被使能时(BOD_RSTEN=0)，当BOD_OUT = 1时，BOD中断将发生。该中断会持续到将BOD_EN设置为"0". 通过禁用CPU中的NVIC以禁用BOD中断或者通过将BOD_EN=0可禁止CPU响应中断，如果需要BOD功能时，可重新使能BOD_EN功能</p> <p><b>注1：</b>当BOD功能使能(BOD_EN =1)并且BOD复位功能使能(BOD_RSTEN=1)时，当BOD检测到电压低于门槛电压时(BOD_OUT=1)，将发出信号复位芯片。</p> <p><b>注2：</b>缺省值由FLASH中用户配置区寄存器CONFIG0 bit[2:0]决定。</p> <p><b>注3：</b>该位是写保护的，修改该位时，需要依次向0x5000_0100写入"59h", "16h", "88h"解除寄存器保护。参考寄存器 REGWRPROT，地址GCR_BA + 0x100</p>															
[2:1]	BOD_VL	<p><b>欠压检测门槛电压 选择（写保护位）</b></p> <p>默认值 由FLASH中用户配置寄存器CONFIG0 bit[22:21]决定</p> <table border="1"> <thead> <tr> <th>BOV_VL[1]</th> <th>BOV_VL[0]</th> <th>欠压值</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>4.4V</td> </tr> <tr> <td>1</td> <td>0</td> <td>3.7V</td> </tr> <tr> <td>0</td> <td>1</td> <td>2.7V</td> </tr> <tr> <td>0</td> <td>0</td> <td>2.2V</td> </tr> </tbody> </table> <p><b>注：</b>该位是写保护的，修改该位时，需要依次向0x5000_0100写入"59h", "16h", "88h"解除寄存器保护。参考寄存器 REGWRPROT，地址GCR_BA + 0x100</p>	BOV_VL[1]	BOV_VL[0]	欠压值	1	1	4.4V	1	0	3.7V	0	1	2.7V	0	0	2.2V
BOV_VL[1]	BOV_VL[0]	欠压值															
1	1	4.4V															
1	0	3.7V															
0	1	2.7V															
0	0	2.2V															
[0]	BOD_EN	<p><b>欠压检测使能（写保护位）</b></p> <p>默认值 由FLASH中用户配置寄存器CONFIG0 bit[23]决定</p> <p>1 = 使能欠压检测功能 0 = 禁用欠压检测功能</p> <p><b>注：</b>该位是写保护的，修改该位时，需要依次向0x5000_0100写入"59h", "16h", "88h"解除寄存器保护。参考寄存器 REGWRPROT，地址GCR_BA + 0x100</p>															

温度传感器控制寄存器 (TEMPCR)

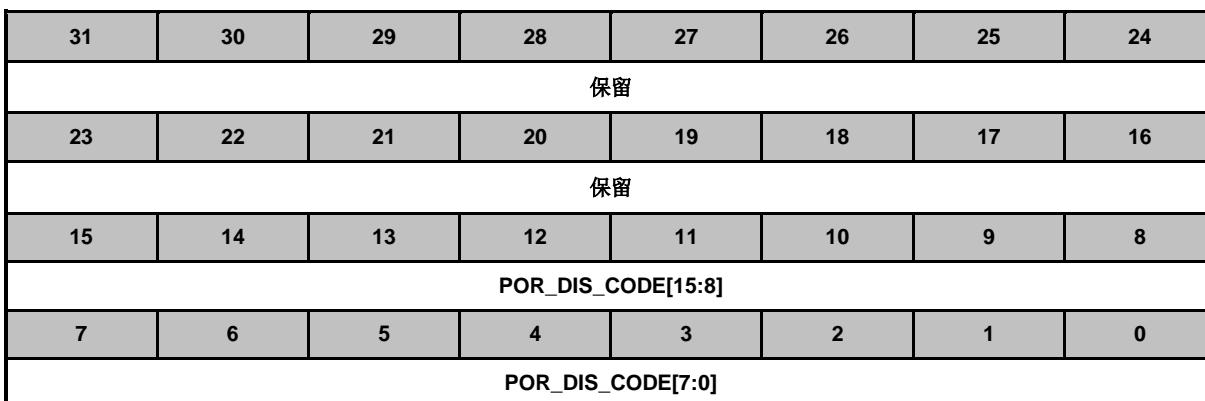
寄存器	偏移量	R/W	描述	复位后的值
TEMPCR	GCR_BA+0x1C	R/W	温度传感器控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
预留							
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							VTEMP_EN

Bits	描述	
[31:1]	保留	保留
[0]	VTEMP_EN	<p><b>使能温度传感器</b>            这个bit 用来使能/禁止温度传感器功能.            1 = 使能温度传感器功能            0 = 禁止温度传感器功能(缺省)</p> <p><b>注:</b> 这个比特设为1之后, ADC通道7可以选择温度传感器输入, 温度值可以在ADC的转换结果中读到. ADC转换功能的细节请参考ADC章节</p>

上电复位控制寄存器(PORCR)

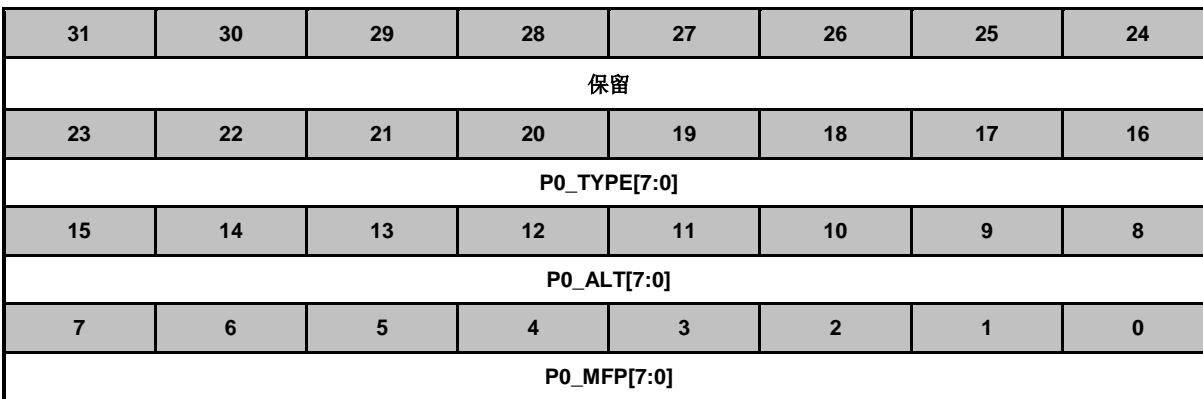
寄存器	偏移量	R/W	描述	复位后的值
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_00xx



Bits	描述	
[31:16]	保留	保留
[15:0]	POR_DIS_CODE	<p>上电复位使能控制（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再次发出复位信号。如果将POR_DIS_CODE 设置为0x5AA5，POR复位功能将被禁用，直到电源电压很低，导致POR_DIS_CODE 设置为其他值，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>nRST引脚复位，看门狗，LVR复位，BOD复位，ICE复位命令和软件复位。</p> <p><b>注：</b>该寄存器是写保护的寄存器，写该位需要先向地址0x5000_0100依次写入“59h”，“16h”，“88h”解除寄存器保护。参考寄存器REGWRPROT的设置，其地址为GCR_BA + 0x100.</p>

**Port0多功能控制寄存器 (P0\_MFP)**

寄存器	偏移量	R/W	描述	复位后的值
P0_MFP	GCR_BA+30	R/W	P0多功能复用与输入类型控制寄存器	0x0000_0000



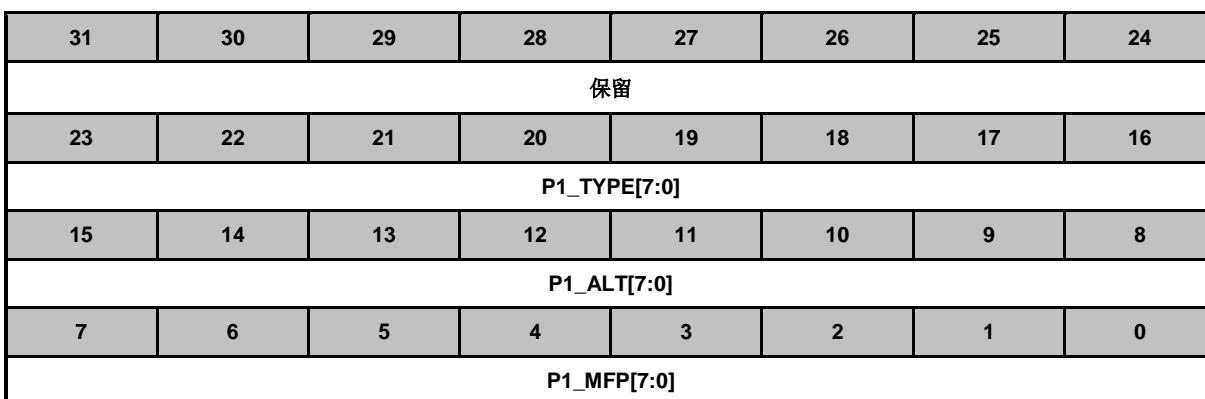
Bits	描述																
[31:26]	保留	保留															
[25]	P0_ALT1[1]	<b>P0.1 功能辅助选择1 (只有M05xxDN/DE支持)</b> P0.1 引脚的功能由 P0_MFP[1], P0_ALT[1], 和 P0_ALT1[1]决定。 细节描述请参考 P0_ALT[1].															
[24]	P0_ALT1[0]	<b>P0.0功能辅助选择1 (只有M05xxDN/DE 支持)</b> P0.0 引脚的功能由P0_MFP[0], P0_ALT[0], 和 P0_ALT1[0].决定 细节描述请参考P0_ALT[0].															
[23:16]	P0_TYPEn	<b>使能P0[7:0]史密特触发输入功能</b> 1= 使能P0[7:0] I/O史密特触发输入功能 0= 禁用P0[7:0] I/O史密特触发功能输入															
[15]	P0_ALT[7]	<b>P0.7 复用功能选择</b> P0.7 的功能取决于P0_MFP[7] 和 P0_ALT[7]. <table border="1"> <tr> <th>P0_ALT[7]</th> <th>P0_MFP[7]</th> <th>P0.7 的功能</th> </tr> <tr> <td></td> <td>0</td> <td>P0.7</td> </tr> <tr> <td>0</td> <td>1</td> <td>AD7(EBI)</td> </tr> <tr> <td>1</td> <td>0</td> <td>SPICLK1(SPI1)</td> </tr> <tr> <td></td> <td>1</td> <td>保留</td> </tr> </table>	P0_ALT[7]	P0_MFP[7]	P0.7 的功能		0	P0.7	0	1	AD7(EBI)	1	0	SPICLK1(SPI1)		1	保留
P0_ALT[7]	P0_MFP[7]	P0.7 的功能															
	0	P0.7															
0	1	AD7(EBI)															
1	0	SPICLK1(SPI1)															
	1	保留															

[14]	<b>P0_ALT[6]</b>	<b>P0.6复用功能选择</b> P0.6的功能取决于 P0_MFP[6] 和 P0_ALT[6].														
		<table border="1"> <thead> <tr> <th><b>P0_ALT[6]</b></th><th><b>P0_MFP[6]</b></th><th><b>P0.6 的功能</b></th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P0.6</td></tr> <tr> <td>0</td><td>1</td><td>AD6(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>MISO_1(SPI1)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	<b>P0_ALT[6]</b>	<b>P0_MFP[6]</b>	<b>P0.6 的功能</b>	0	0	P0.6	0	1	AD6(EBI)	1	0	MISO_1(SPI1)	1	1
<b>P0_ALT[6]</b>	<b>P0_MFP[6]</b>	<b>P0.6 的功能</b>														
0	0	P0.6														
0	1	AD6(EBI)														
1	0	MISO_1(SPI1)														
1	1	保留														
[13]	<b>P0.5复用功能选择</b> P0.5的功能取决于 P0_MFP[5] 和 P0_ALT[5].															
	<table border="1"> <thead> <tr> <th><b>P0_ALT[5]</b></th><th><b>P0_MFP[5]</b></th><th><b>P0.5 的功能</b></th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P0.5</td></tr> <tr> <td>0</td><td>1</td><td>AD5(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>MOSI_1(SPI1)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	<b>P0_ALT[5]</b>	<b>P0_MFP[5]</b>	<b>P0.5 的功能</b>	0	0	P0.5	0	1	AD5(EBI)	1	0	MOSI_1(SPI1)	1	1	保留
<b>P0_ALT[5]</b>	<b>P0_MFP[5]</b>	<b>P0.5 的功能</b>														
0	0	P0.5														
0	1	AD5(EBI)														
1	0	MOSI_1(SPI1)														
1	1	保留														
[12]	<b>P0.4复用功能选择</b> P0.4 的功能取决于 P0_MFP[4] 和 P0_ALT[4].															
	<table border="1"> <thead> <tr> <th><b>P0_ALT[4]</b></th><th><b>P0_MFP[4]</b></th><th><b>P0.4 的功能</b></th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P0.4</td></tr> <tr> <td>0</td><td>1</td><td>AD4(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>SPISS1(SPI1)</td></tr> <tr> <td></td><td>1</td><td>保留</td></tr> </tbody> </table>	<b>P0_ALT[4]</b>	<b>P0_MFP[4]</b>	<b>P0.4 的功能</b>	0	0	P0.4	0	1	AD4(EBI)	1	0	SPISS1(SPI1)		1	保留
<b>P0_ALT[4]</b>	<b>P0_MFP[4]</b>	<b>P0.4 的功能</b>														
0	0	P0.4														
0	1	AD4(EBI)														
1	0	SPISS1(SPI1)														
	1	保留														
[11]	<b>P0_ALT[3]</b>	<b>P0.3复用功能选择</b> P0.3的功能取决于 P0_MFP[3] 和 P0_ALT[3].														
		<table border="1"> <thead> <tr> <th><b>P0_ALT[3]</b></th><th><b>P0_MFP[3]</b></th><th><b>P0.3的功能</b></th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P0.3</td></tr> <tr> <td>0</td><td>1</td><td>AD3(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>RTS0(UART0)</td></tr> <tr> <td></td><td>1</td><td>RXD</td></tr> </tbody> </table>	<b>P0_ALT[3]</b>	<b>P0_MFP[3]</b>	<b>P0.3的功能</b>	0	0	P0.3	0	1	AD3(EBI)	1	0	RTS0(UART0)		1
<b>P0_ALT[3]</b>	<b>P0_MFP[3]</b>	<b>P0.3的功能</b>														
0	0	P0.3														
0	1	AD3(EBI)														
1	0	RTS0(UART0)														
	1	RXD														

[10]	<b>P0_ALT[2]</b>	<p><b>P0.2复用功能选择</b> P0.2的功能取决于P0_MFP[2] 和 P0_ALT[2].</p> <table border="1"> <thead> <tr> <th><b>P0_ALT[2]</b></th><th><b>P0_MFP[2]</b></th><th><b>P0.2的功能</b></th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P0.2</td></tr> <tr> <td>0</td><td>1</td><td>AD2(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>CTS0(UART0)</td></tr> <tr> <td>1</td><td>1</td><td>TXD</td></tr> </tbody> </table>	<b>P0_ALT[2]</b>	<b>P0_MFP[2]</b>	<b>P0.2的功能</b>	0	0	P0.2	0	1	AD2(EBI)	1	0	CTS0(UART0)	1	1	TXD									
<b>P0_ALT[2]</b>	<b>P0_MFP[2]</b>	<b>P0.2的功能</b>																								
0	0	P0.2																								
0	1	AD2(EBI)																								
1	0	CTS0(UART0)																								
1	1	TXD																								
[9]	<b>P0_ALT[1]</b>	<p><b>P0.1复用功能选择</b> P0.1 的功能取决于P0_MFP[1] 、 P0_ALT[1]和P0_ALT1[1].</p> <table border="1"> <thead> <tr> <th><b>P0_ALT1[1]</b></th><th><b>P0_ALT[1]</b></th><th><b>P0_MFP[1]</b></th><th><b>P0.1的功能</b></th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>P0.1</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>AD1(EBI)</td></tr> <tr> <td>0</td><td>1</td><td>0</td><td>RTS1(UART1)</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>RXD1</td></tr> <tr> <td>1</td><td>0</td><td>0</td><td>ACMP3_N ( 只有 M05xxDN/DE 支持)</td></tr> </tbody> </table>	<b>P0_ALT1[1]</b>	<b>P0_ALT[1]</b>	<b>P0_MFP[1]</b>	<b>P0.1的功能</b>	0	0	0	P0.1	0	0	1	AD1(EBI)	0	1	0	RTS1(UART1)	0	1	1	RXD1	1	0	0	ACMP3_N ( 只有 M05xxDN/DE 支持)
<b>P0_ALT1[1]</b>	<b>P0_ALT[1]</b>	<b>P0_MFP[1]</b>	<b>P0.1的功能</b>																							
0	0	0	P0.1																							
0	0	1	AD1(EBI)																							
0	1	0	RTS1(UART1)																							
0	1	1	RXD1																							
1	0	0	ACMP3_N ( 只有 M05xxDN/DE 支持)																							
[8]	<b>P0_ALT[0]</b>	<p><b>P0.0复用功能选择</b> P0.0的功能取决于 P0_MFP[0] 和、 P0_ALT[0]和P0_ALT1[0].</p> <table border="1"> <thead> <tr> <th><b>P0_ALT1[0]</b></th><th><b>P0_ALT[0]</b></th><th><b>P0_MFP[0]</b></th><th><b>P0.0的功能</b></th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td><td>P0.0</td></tr> <tr> <td>0</td><td>0</td><td>1</td><td>AD0(EBI)</td></tr> <tr> <td>0</td><td>1</td><td>0</td><td>CTS1(UART1)</td></tr> <tr> <td>0</td><td>1</td><td>1</td><td>TXD1</td></tr> <tr> <td>1</td><td>0</td><td>0</td><td>ACMP3_P ( 只有 M05xxDN/DE 支持)</td></tr> </tbody> </table>	<b>P0_ALT1[0]</b>	<b>P0_ALT[0]</b>	<b>P0_MFP[0]</b>	<b>P0.0的功能</b>	0	0	0	P0.0	0	0	1	AD0(EBI)	0	1	0	CTS1(UART1)	0	1	1	TXD1	1	0	0	ACMP3_P ( 只有 M05xxDN/DE 支持)
<b>P0_ALT1[0]</b>	<b>P0_ALT[0]</b>	<b>P0_MFP[0]</b>	<b>P0.0的功能</b>																							
0	0	0	P0.0																							
0	0	1	AD0(EBI)																							
0	1	0	CTS1(UART1)																							
0	1	1	TXD1																							
1	0	0	ACMP3_P ( 只有 M05xxDN/DE 支持)																							
[7:0]	<b>P0_MFP[7:0]</b>	<p><b>P0复用功能选择</b> P0 引脚的功能取决于 P0_MFP 和 P0_ALT. 参考P0_ALT的详细描述</p>																								

**Port1多功能控制寄存器 (P1\_MFP)**

寄存器	偏移量	R/W	描述	复位后的值
P1_MFP	GCR_BA+34	R/W	P1复用功能与输入类型控制寄存器	0x0000_0000



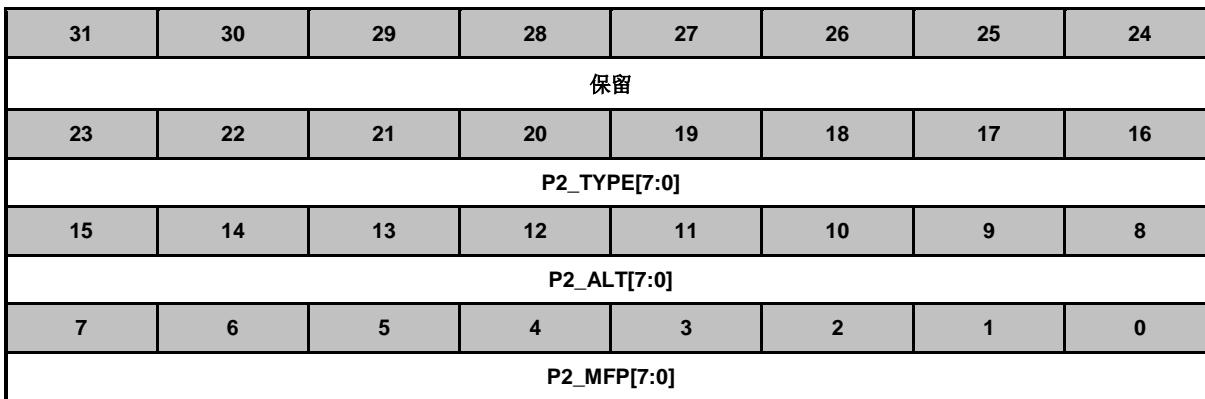
Bits	描述																
[31:24]	保留	保留															
[23:16]	P1_TYPEn	<p>使能P1[7:0] 输入史密特触发功能 1= 使能P1[7:0] I/O输入史密特触发功能 0= 禁用P1[7:0] I/O输入史密特触发功能</p>															
[15]	P1_ALT[7]	<p><b>P1.7复用功能选择</b> P1.7的功能取决于P1_MFP[7] 和 P1_ALT[7].</p> <table border="1"> <thead> <tr> <th>P1_ALT[7]</th> <th>P1_MFP[7]</th> <th>P1.7 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>P1.</td> </tr> <tr> <td>0</td> <td>1</td> <td>AIN7(ADC)</td> </tr> <tr> <td>1</td> <td>0</td> <td>SPICLK0(SPI0)</td> </tr> <tr> <td>1</td> <td>1</td> <td>ACMP2_P ( 只有 M05xxDN/DE支持)</td> </tr> </tbody> </table>	P1_ALT[7]	P1_MFP[7]	P1.7 的功能	0		P1.	0	1	AIN7(ADC)	1	0	SPICLK0(SPI0)	1	1	ACMP2_P ( 只有 M05xxDN/DE支持)
P1_ALT[7]	P1_MFP[7]	P1.7 的功能															
0		P1.															
0	1	AIN7(ADC)															
1	0	SPICLK0(SPI0)															
1	1	ACMP2_P ( 只有 M05xxDN/DE支持)															

[14]	P1_ALT[6]	<p><b>P1.6 复用功能选择</b></p> <p>P1.6 的功能取决于 P1_MFP[6] 和 P1_ALT[6].</p> <table border="1"> <thead> <tr> <th>P1_ALT[6]</th><th>P1_MFP[6]</th><th>P1.6 的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td></td><td>P1.6</td></tr> <tr> <td>0</td><td>1</td><td>AIN6(ADC)</td></tr> <tr> <td>1</td><td>0</td><td>MISO_0(SPI0)</td></tr> <tr> <td>1</td><td>1</td><td>ACMP2_N (只有 M05xxDN/DE 支持)</td></tr> </tbody> </table>	P1_ALT[6]	P1_MFP[6]	P1.6 的功能	0		P1.6	0	1	AIN6(ADC)	1	0	MISO_0(SPI0)	1	1	ACMP2_N (只有 M05xxDN/DE 支持)
P1_ALT[6]	P1_MFP[6]	P1.6 的功能															
0		P1.6															
0	1	AIN6(ADC)															
1	0	MISO_0(SPI0)															
1	1	ACMP2_N (只有 M05xxDN/DE 支持)															
[13]	P1_ALT[5]	<p><b>P1.5 复用功能选择</b></p> <p>P1.5 的功能取决于 P1_MFP[5] 和 P1_ALT[5].</p> <table border="1"> <thead> <tr> <th>P1_ALT[5]</th><th>P1_MFP[5]</th><th>P1.5 的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P1.5</td></tr> <tr> <td>0</td><td>1</td><td>AIN5(ADC)</td></tr> <tr> <td>1</td><td>0</td><td>MOSI_0(SPI0)</td></tr> <tr> <td>1</td><td>1</td><td>ACMP0_P</td></tr> </tbody> </table>	P1_ALT[5]	P1_MFP[5]	P1.5 的功能	0	0	P1.5	0	1	AIN5(ADC)	1	0	MOSI_0(SPI0)	1	1	ACMP0_P
P1_ALT[5]	P1_MFP[5]	P1.5 的功能															
0	0	P1.5															
0	1	AIN5(ADC)															
1	0	MOSI_0(SPI0)															
1	1	ACMP0_P															
[12]	P1_ALT[4]	<p><b>P1.4 复用功能选择</b></p> <p>P1.4 的功能取决于 P1_MFP[4] 和 P1_ALT[4].</p> <table border="1"> <thead> <tr> <th>P1_ALT[4]</th><th>P1_MFP[4]</th><th>P1.4 的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P1.4</td></tr> <tr> <td></td><td>1</td><td>AIN4(ADC)</td></tr> <tr> <td>1</td><td>0</td><td>SPISS0(SPI0)</td></tr> <tr> <td>1</td><td>1</td><td>ACMP0_N</td></tr> </tbody> </table>	P1_ALT[4]	P1_MFP[4]	P1.4 的功能	0	0	P1.4		1	AIN4(ADC)	1	0	SPISS0(SPI0)	1	1	ACMP0_N
P1_ALT[4]	P1_MFP[4]	P1.4 的功能															
0	0	P1.4															
	1	AIN4(ADC)															
1	0	SPISS0(SPI0)															
1	1	ACMP0_N															
[11]	P1_ALT[3]	<p><b>P1.3 复用功能选择</b></p> <p>P1.3 的功能取决于 P1_MFP[3] 和 P1_ALT[3].</p> <table border="1"> <thead> <tr> <th>P1_ALT[3]</th><th>P1_MFP[3]</th><th>P1.3 的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P1.3</td></tr> <tr> <td>0</td><td>1</td><td>AIN3(ADC)</td></tr> <tr> <td>1</td><td>0</td><td>TXD1(UART1)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P1_ALT[3]	P1_MFP[3]	P1.3 的功能	0	0	P1.3	0	1	AIN3(ADC)	1	0	TXD1(UART1)	1	1	保留
P1_ALT[3]	P1_MFP[3]	P1.3 的功能															
0	0	P1.3															
0	1	AIN3(ADC)															
1	0	TXD1(UART1)															
1	1	保留															

[10]	P1_ALT[2]	<p><b>P1.2 复用功能选择</b></p> <p>P1.2的功能取决于P1_MFP[2] 和 P1_ALT[2].</p> <table border="1"> <thead> <tr> <th>P1_ALT[2]</th><th>P1_MFP[2]</th><th>P1.2的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P1.2</td></tr> <tr> <td>0</td><td>1</td><td>AIN2(ADC)</td></tr> <tr> <td>1</td><td>0</td><td>RXD1(UART1)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P1_ALT[2]	P1_MFP[2]	P1.2的功能	0	0	P1.2	0	1	AIN2(ADC)	1	0	RXD1(UART1)	1	1	保留
P1_ALT[2]	P1_MFP[2]	P1.2的功能															
0	0	P1.2															
0	1	AIN2(ADC)															
1	0	RXD1(UART1)															
1	1	保留															
[9]	P1_ALT[1]	<p><b>P1.1 复用功能选择</b></p> <p>P1.1的功能取决于P1_MFP[1] and P1_ALT[1].</p> <table border="1"> <thead> <tr> <th>P1_ALT[1]</th><th>P1_MFP[1]</th><th>P1.1的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P1.1</td></tr> <tr> <td></td><td>1</td><td>AIN1(ADC)</td></tr> <tr> <td>1</td><td>0</td><td>T3(Timer3)</td></tr> <tr> <td>1</td><td>1</td><td>nWRH</td></tr> </tbody> </table>	P1_ALT[1]	P1_MFP[1]	P1.1的功能	0	0	P1.1		1	AIN1(ADC)	1	0	T3(Timer3)	1	1	nWRH
P1_ALT[1]	P1_MFP[1]	P1.1的功能															
0	0	P1.1															
	1	AIN1(ADC)															
1	0	T3(Timer3)															
1	1	nWRH															
[8]	P1_ALT[0]	<p><b>P1.0 复用功能选择</b></p> <p>P1.0的功能取决于P1_MFP[0] and P1_ALT[0].</p> <table border="1"> <thead> <tr> <th>P1_ALT[0]</th><th>P1_MFP[0]</th><th>P1.0的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P1.0</td></tr> <tr> <td>0</td><td>1</td><td>AIN0(ADC)</td></tr> <tr> <td>1</td><td>0</td><td>T2 (Timer2)</td></tr> <tr> <td>1</td><td>1</td><td>nWRL</td></tr> </tbody> </table>	P1_ALT[0]	P1_MFP[0]	P1.0的功能	0	0	P1.0	0	1	AIN0(ADC)	1	0	T2 (Timer2)	1	1	nWRL
P1_ALT[0]	P1_MFP[0]	P1.0的功能															
0	0	P1.0															
0	1	AIN0(ADC)															
1	0	T2 (Timer2)															
1	1	nWRL															
[7:0]	P1_MFP[7:0]	<p><b>P1 复用功能选择</b></p> <p>P1的功能取决于P1_MFP 和 P1_ALT.</p> <p>参考P1_ALT的详细描述</p>															

**Port2多功能控制寄存器(P2\_MFP)**

寄存器	偏移量	R/W	描述	复位后的值
P2_MFP	GCR_BA+0x38	R/W	P2复用功能与输入类型控制寄存器	0x0000_0000



Bits	描述																
[31:24]	保留	保留															
[23:16]	P2_TYPEn	<b>P2[7:0] 输入史密特触发功能使能</b> 1= 使能P2[7:0] I/O输入史密特触发功能 0= 禁用P2[7:0] I/O输入史密特触发功能															
[15]	P2_ALT[7]	<b>P2.7 复用功能选择</b> P2.7的功能取决于P2_MFP[7] and P2_ALT[7]. <table border="1"> <thead> <tr> <th>P2_ALT[7]</th> <th>P2_MFP[7]</th> <th>P2.7 的功能</th> </tr> </thead> <tbody> <tr> <td></td> <td>0</td> <td>P2.7</td> </tr> <tr> <td>0</td> <td>1</td> <td>AD15(EBI)</td> </tr> <tr> <td>1</td> <td>0</td> <td>PWM7(PWMB 通道3)</td> </tr> <tr> <td>1</td> <td>1</td> <td>保留</td> </tr> </tbody> </table>	P2_ALT[7]	P2_MFP[7]	P2.7 的功能		0	P2.7	0	1	AD15(EBI)	1	0	PWM7(PWMB 通道3)	1	1	保留
P2_ALT[7]	P2_MFP[7]	P2.7 的功能															
	0	P2.7															
0	1	AD15(EBI)															
1	0	PWM7(PWMB 通道3)															
1	1	保留															

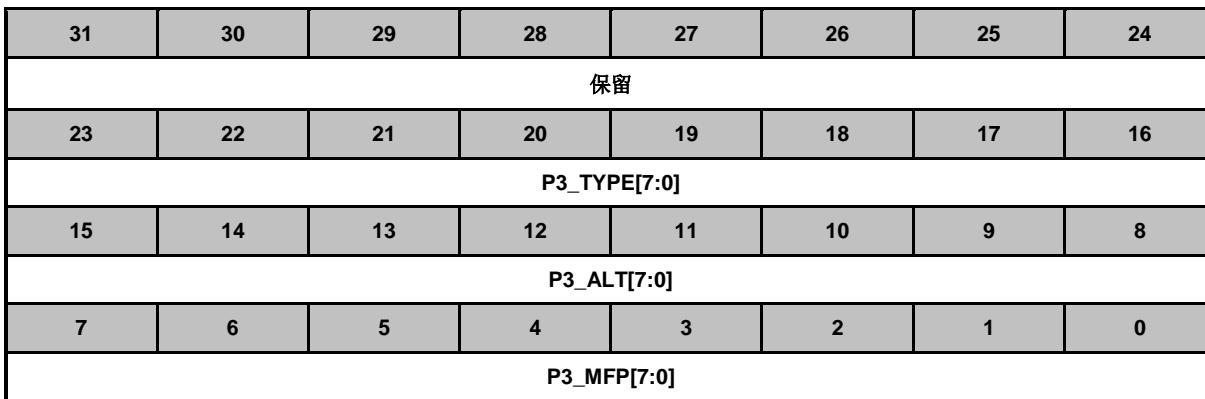
[14]	P2_ALT[6]	<b>P2.6 复用功能选择</b> P2.6的功能取决于P2_MFP[6] and P2_ALT[6].													
		<table border="1"> <thead> <tr> <th>P2_ALT[6]</th><th>P2_MFP[6]</th><th>P2.6 的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P2.6</td></tr> <tr> <td>0</td><td>1</td><td>AD14(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>PWM6(PWMB 通道 2)</td></tr> <tr> <td>1</td><td>1</td><td>ACMP1_O</td></tr> </tbody> </table>	P2_ALT[6]	P2_MFP[6]	P2.6 的功能	0	0	P2.6	0	1	AD14(EBI)	1	0	PWM6(PWMB 通道 2)	1
P2_ALT[6]	P2_MFP[6]	P2.6 的功能													
0	0	P2.6													
0	1	AD14(EBI)													
1	0	PWM6(PWMB 通道 2)													
1	1	ACMP1_O													
[13]	P2_ALT[5]	<b>P2.5 复用功能选择</b> P2.5的功能取决于P2_MFP[5] and P2_ALT[5].													
		<table border="1"> <thead> <tr> <th>P2_ALT[5]</th><th>P2_MFP[5]</th><th>P2.5 的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P2.5</td></tr> <tr> <td>0</td><td>1</td><td>AD13(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>PWM5(PWMB 通道 1)</td></tr> <tr> <td>1</td><td>1</td><td>SDA1(I2C1) (只有 M05xxDN/DE 支持)</td></tr> </tbody> </table>	P2_ALT[5]	P2_MFP[5]	P2.5 的功能	0	0	P2.5	0	1	AD13(EBI)	1	0	PWM5(PWMB 通道 1)	1
P2_ALT[5]	P2_MFP[5]	P2.5 的功能													
0	0	P2.5													
0	1	AD13(EBI)													
1	0	PWM5(PWMB 通道 1)													
1	1	SDA1(I2C1) (只有 M05xxDN/DE 支持)													
[12]	P2_ALT[4]	<b>P2.4 复用功能选择</b> P2.4的功能取决于P2_MFP[4] and P2_ALT[4].													
		<table border="1"> <thead> <tr> <th>P2_ALT[4]</th><th>P2_MFP[4]</th><th>P2.4的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P2.4</td></tr> <tr> <td>0</td><td>1</td><td>AD12(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>PWM4(PWMB 通道 0)</td></tr> <tr> <td>1</td><td>1</td><td>SCL1(I2C1) 只有 M05xxDN/DE 支持)</td></tr> </tbody> </table>	P2_ALT[4]	P2_MFP[4]	P2.4的功能	0	0	P2.4	0	1	AD12(EBI)	1	0	PWM4(PWMB 通道 0)	1
P2_ALT[4]	P2_MFP[4]	P2.4的功能													
0	0	P2.4													
0	1	AD12(EBI)													
1	0	PWM4(PWMB 通道 0)													
1	1	SCL1(I2C1) 只有 M05xxDN/DE 支持)													

[11]	P2_ALT[3]	<p><b>P2.3 复用功能选择</b> P2.3的功能取决于P2_MFP[3] and P2_ALT[3].</p> <table border="1"> <thead> <tr> <th>P2_ALT[3]</th><th>P2_MFP[3]</th><th>P2.3的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P2.3</td></tr> <tr> <td>0</td><td>1</td><td>AD11(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>PWM3(PWMA 通道3)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P2_ALT[3]	P2_MFP[3]	P2.3的功能	0	0	P2.3	0	1	AD11(EBI)	1	0	PWM3(PWMA 通道3)	1	1	保留
P2_ALT[3]	P2_MFP[3]	P2.3的功能															
0	0	P2.3															
0	1	AD11(EBI)															
1	0	PWM3(PWMA 通道3)															
1	1	保留															
[10]	P2_ALT[2]	<p><b>P2.2 复用功能选择</b> P2.2的功能取决于P2_MFP[2] 和 P2_ALT[2].</p> <table border="1"> <thead> <tr> <th>P2_ALT[2]</th><th>P2_MFP[2]</th><th>P2.2的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P2.2</td></tr> <tr> <td></td><td>1</td><td>AD10(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>PWM2(PWMA 通道2)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P2_ALT[2]	P2_MFP[2]	P2.2的功能	0	0	P2.2		1	AD10(EBI)	1	0	PWM2(PWMA 通道2)	1	1	保留
P2_ALT[2]	P2_MFP[2]	P2.2的功能															
0	0	P2.2															
	1	AD10(EBI)															
1	0	PWM2(PWMA 通道2)															
1	1	保留															
[9]	P2_ALT[1]	<p><b>P2.1 复用功能选择</b> P2.1的功能取决于P2_MFP[1] and P2_ALT[1].</p> <table border="1"> <thead> <tr> <th>P2_ALT[1]</th><th>P2_MFP[1]</th><th>P2.1的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P2.1</td></tr> <tr> <td>0</td><td>1</td><td>AD9(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>PWM1(PWMA 通道1)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P2_ALT[1]	P2_MFP[1]	P2.1的功能	0	0	P2.1	0	1	AD9(EBI)	1	0	PWM1(PWMA 通道1)	1	1	保留
P2_ALT[1]	P2_MFP[1]	P2.1的功能															
0	0	P2.1															
0	1	AD9(EBI)															
1	0	PWM1(PWMA 通道1)															
1	1	保留															
[8]	P2_ALT[0]	<p><b>P2.0 复用功能选择</b> P2.0的功能取决于P2_MFP[0] and P2_ALT[0].</p> <table border="1"> <thead> <tr> <th>P2_ALT[0]</th><th>P2_MFP[0]</th><th>P2.0的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td></td><td>P2.0</td></tr> <tr> <td>0</td><td>1</td><td>AD8(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>PWM0(PWMA通道0)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P2_ALT[0]	P2_MFP[0]	P2.0的功能	0		P2.0	0	1	AD8(EBI)	1	0	PWM0(PWMA通道0)	1	1	保留
P2_ALT[0]	P2_MFP[0]	P2.0的功能															
0		P2.0															
0	1	AD8(EBI)															
1	0	PWM0(PWMA通道0)															
1	1	保留															

[7:0]	P2_MFP[7:0]	<b>P2 复用功能选择</b> P2的功能取决于P2_MFP 和 P2_ALT. 参考P2_ALT的详细描述.
-------	-------------	--

**Port3多功能控制寄存器(P3\_MFP)**

寄存器	偏移量	R/W	描述	复位后的值
P3_MFP	GCR_BA+3C	R/W	P3复用功能与输入类型控制寄存器	0x0000_0000



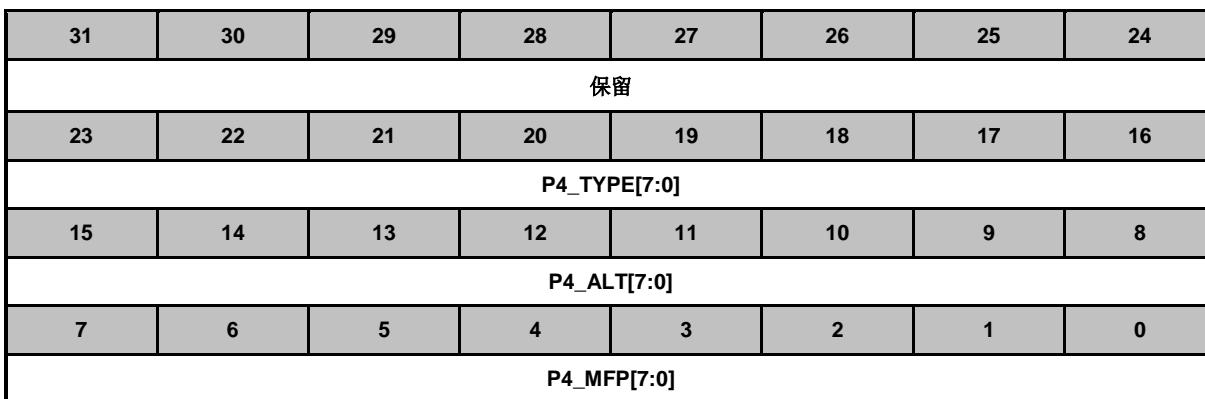
Bits	描述															
[31:24]	保留															
[23:16]	<p>P3_TYPEn 使能P3[7:0] 输入史密特触发功能 1= 使能P3[7:0] I/O输入史密特触发功能 0= 禁用P3[7:0] I/O输入史密特触发功能</p>															
[15]	<p><b>P3.7 复用功能选择</b> P3.7的功能取决于P3_MFP[7] 和 P3_ALT[7].</p> <table border="1"> <tr><th>P3_ALT[7]</th><th>P3_MFP[7]</th><th>P3.7 的功能</th></tr> <tr><td>0</td><td>0</td><td>P3.7</td></tr> <tr><td>0</td><td>1</td><td>RD(EBI)</td></tr> <tr><td>1</td><td>x</td><td>保留</td></tr> </table>	P3_ALT[7]	P3_MFP[7]	P3.7 的功能	0	0	P3.7	0	1	RD(EBI)	1	x	保留			
P3_ALT[7]	P3_MFP[7]	P3.7 的功能														
0	0	P3.7														
0	1	RD(EBI)														
1	x	保留														
[14]	<p><b>P3.6 复用功能选择</b> P3.6的功能取决于P3_MFP[6] 和 P3_ALT[6].</p> <table border="1"> <tr><th>P3_ALT[6]</th><th>P3_MFP[6]</th><th>P3.6 的功能</th></tr> <tr><td>0</td><td>0</td><td>P3.6</td></tr> <tr><td>0</td><td>1</td><td>WR(EBI)</td></tr> <tr><td>1</td><td>0</td><td>CKO(时钟除频输出)</td></tr> <tr><td>1</td><td>1</td><td>ACMP0_o</td></tr> </table>	P3_ALT[6]	P3_MFP[6]	P3.6 的功能	0	0	P3.6	0	1	WR(EBI)	1	0	CKO(时钟除频输出)	1	1	ACMP0_o
P3_ALT[6]	P3_MFP[6]	P3.6 的功能														
0	0	P3.6														
0	1	WR(EBI)														
1	0	CKO(时钟除频输出)														
1	1	ACMP0_o														

[13]	P3_ALT[5]	<p><b>P3.5 复用功能选择</b></p> <p>P3.5的功能取决于P3_MFP[5] 和 P3_ALT[5].</p> <table border="1"> <thead> <tr> <th>P3_ALT[5]</th><th>P3_MFP[5]</th><th>P3.5 的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P3.5</td></tr> <tr> <td>0</td><td>1</td><td>T1(Timer1)</td></tr> <tr> <td>1</td><td>0</td><td>SCL0(I<sup>2</sup>C0)</td></tr> <tr> <td>1</td><td>1</td><td>CKO(时钟除频输出)(只有M05xxDN/DE 支持)</td></tr> </tbody> </table>	P3_ALT[5]	P3_MFP[5]	P3.5 的功能	0	0	P3.5	0	1	T1(Timer1)	1	0	SCL0(I <sup>2</sup> C0)	1	1	CKO(时钟除频输出)(只有M05xxDN/DE 支持)
P3_ALT[5]	P3_MFP[5]	P3.5 的功能															
0	0	P3.5															
0	1	T1(Timer1)															
1	0	SCL0(I <sup>2</sup> C0)															
1	1	CKO(时钟除频输出)(只有M05xxDN/DE 支持)															
[12]	P3_ALT[4]	<p><b>P3.4 复用功能选择</b></p> <p>P3.4的功能取决于P3_MFP[4] 和 P3_ALT[4].</p> <table border="1"> <thead> <tr> <th>P3_ALT[4]</th><th>P3_MFP[4]</th><th>P3.4的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P3.4</td></tr> <tr> <td>0</td><td>1</td><td>T0(Timer0)</td></tr> <tr> <td>1</td><td></td><td>SDA0(I<sup>2</sup>C0)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P3_ALT[4]	P3_MFP[4]	P3.4的功能	0	0	P3.4	0	1	T0(Timer0)	1		SDA0(I <sup>2</sup> C0)	1	1	保留
P3_ALT[4]	P3_MFP[4]	P3.4的功能															
0	0	P3.4															
0	1	T0(Timer0)															
1		SDA0(I <sup>2</sup> C0)															
1	1	保留															
[11]	P3_ALT[3]	<p><b>P3.3 复用功能选择</b></p> <p>P3.3的功能取决于P3_MFP[3] 和 P3_ALT[3].</p> <table border="1"> <thead> <tr> <th>P3_ALT[3]</th><th>P3_MFP[3]</th><th>P3.3的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P3.3</td></tr> <tr> <td>0</td><td>1</td><td>/INT</td></tr> <tr> <td>1</td><td>0</td><td>MCLK(EBI)</td></tr> <tr> <td>1</td><td>1</td><td>T1EX</td></tr> </tbody> </table>	P3_ALT[3]	P3_MFP[3]	P3.3的功能	0	0	P3.3	0	1	/INT	1	0	MCLK(EBI)	1	1	T1EX
P3_ALT[3]	P3_MFP[3]	P3.3的功能															
0	0	P3.3															
0	1	/INT															
1	0	MCLK(EBI)															
1	1	T1EX															
[10]	P3_ALT[2]	<p><b>P3.2 复用功能选择</b></p> <p>P3.2的功能取决于P3_MFP[2] and P3_ALT[2].</p> <table border="1"> <thead> <tr> <th>P3_ALT[2]</th><th>P3_MFP[2]</th><th>P3.2的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P3.2</td></tr> <tr> <td>0</td><td>1</td><td>/INT0</td></tr> <tr> <td>1</td><td>0</td><td>T0EX</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P3_ALT[2]	P3_MFP[2]	P3.2的功能	0	0	P3.2	0	1	/INT0	1	0	T0EX	1	1	保留
P3_ALT[2]	P3_MFP[2]	P3.2的功能															
0	0	P3.2															
0	1	/INT0															
1	0	T0EX															
1	1	保留															

[9]	P3_ALT[1]	<p><b>P3.1 复用功能选择</b></p> <p>P3.1的功能取决于P3_MFP[1] and P3_ALT[1].</p> <table border="1"> <thead> <tr> <th>P3_ALT[1]</th><th>P3_MFP[1]</th><th>P3.1的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P3.1</td></tr> <tr> <td>0</td><td>1</td><td>TXD(UART0)</td></tr> <tr> <td>1</td><td>0</td><td>ACMP1_P</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P3_ALT[1]	P3_MFP[1]	P3.1的功能	0	0	P3.1	0	1	TXD(UART0)	1	0	ACMP1_P	1	1	保留
P3_ALT[1]	P3_MFP[1]	P3.1的功能															
0	0	P3.1															
0	1	TXD(UART0)															
1	0	ACMP1_P															
1	1	保留															
[8]	P3_ALT[0]	<p><b>P3.0 复用功能选择</b></p> <p>P3.0的功能取决于P3_MFP[0] and P3_ALT[0].</p> <table border="1"> <thead> <tr> <th>P3_ALT[0]</th><th>P3_MFP[0]</th><th>P3.0的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P3.0</td></tr> <tr> <td>0</td><td>1</td><td>RXD(UART0)</td></tr> <tr> <td>1</td><td>0</td><td>ACMP1_N</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P3_ALT[0]	P3_MFP[0]	P3.0的功能	0	0	P3.0	0	1	RXD(UART0)	1	0	ACMP1_N	1	1	保留
P3_ALT[0]	P3_MFP[0]	P3.0的功能															
0	0	P3.0															
0	1	RXD(UART0)															
1	0	ACMP1_N															
1	1	保留															
[7:0]	P3_MFP[7:0]	<p><b>P3 复用功能选择</b></p> <p>P3的功能取决于P3_MFP 和 P3_ALT.</p> <p>参考 P3_ALT的详细描述.</p>															

**Port4多功能控制寄存器(P4\_MFP)**

寄存器	偏移量	R/W	描述	复位后的值
P4_MFP	GCR_BA+40	R/W	P4复用功能与输入类型控制寄存器	0x0000_00C0



Bits	描述												
[31:24]	保留												
[23:16]	<p>P4_TYPEn</p> <p>使能P4[7:0] 输入史密特触发功能 1= 使能P4[7:0] I/O输入史密特触发功能 0= 禁用P4[7:0] I/O输入史密特触发功能</p>												
[15]	<p><b>P4.7 复用功能选择</b> P4.7的功能取决于P4_MFP[7] and P4_ALT[7].</p> <table border="1"> <thead> <tr> <th>P4_ALT[7]</th> <th>P4_MFP[7]</th> <th>P4.7 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P4.7</td> </tr> <tr> <td>0</td> <td>1</td> <td>ICE_DAT(ICE)</td> </tr> <tr> <td>1</td> <td>x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[7]	P4_MFP[7]	P4.7 的功能	0	0	P4.7	0	1	ICE_DAT(ICE)	1	x	保留
P4_ALT[7]	P4_MFP[7]	P4.7 的功能											
0	0	P4.7											
0	1	ICE_DAT(ICE)											
1	x	保留											
[14]	<p><b>P4.6 复用功能选择</b> P4.6的功能取决于P4_MFP[6] and P4_ALT[6].</p> <table border="1"> <thead> <tr> <th>P4_ALT[6]</th> <th>P4_MFP[6]</th> <th>P4.6 的功能</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>P4.6</td> </tr> <tr> <td>0</td> <td>1</td> <td>ICE_CLK(ICE)</td> </tr> <tr> <td>1</td> <td>x</td> <td>保留</td> </tr> </tbody> </table>	P4_ALT[6]	P4_MFP[6]	P4.6 的功能	0	0	P4.6	0	1	ICE_CLK(ICE)	1	x	保留
P4_ALT[6]	P4_MFP[6]	P4.6 的功能											
0	0	P4.6											
0	1	ICE_CLK(ICE)											
1	x	保留											

[13]	P4_ALT[5]	<p><b>P4.5 复用功能选择</b> P4.5的功能取决于P4_MFP[5] and P4_ALT[5].</p> <table border="1"> <thead> <tr> <th>P4_ALT[5]</th><th>P4_MFP[5]</th><th>P4.5 的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P4.5</td></tr> <tr> <td>0</td><td>1</td><td>ALE(EBI)</td></tr> <tr> <td>1</td><td>0</td><td>SDA1(I<sup>2</sup>C1) (只有M05xxDN/DE 支持)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P4_ALT[5]	P4_MFP[5]	P4.5 的功能	0	0	P4.5	0	1	ALE(EBI)	1	0	SDA1(I <sup>2</sup> C1) (只有M05xxDN/DE 支持)	1	1	保留
P4_ALT[5]	P4_MFP[5]	P4.5 的功能															
0	0	P4.5															
0	1	ALE(EBI)															
1	0	SDA1(I <sup>2</sup> C1) (只有M05xxDN/DE 支持)															
1	1	保留															
[12]	P4_ALT[4]	<p><b>P4.4 复用功能选择</b> P4.4的功能取决于P4_MFP[4] and P4_ALT[4].</p> <table border="1"> <thead> <tr> <th>P4_ALT[4]</th><th>P4_MFP[4]</th><th>P4.4的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P4.4</td></tr> <tr> <td>0</td><td>1</td><td>nCS (EBI)</td></tr> <tr> <td>1</td><td>0</td><td>SCL1(I<sup>2</sup>C1) (只有M05xxDN/DE 支持)</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P4_ALT[4]	P4_MFP[4]	P4.4的功能	0	0	P4.4	0	1	nCS (EBI)	1	0	SCL1(I <sup>2</sup> C1) (只有M05xxDN/DE 支持)	1	1	保留
P4_ALT[4]	P4_MFP[4]	P4.4的功能															
0	0	P4.4															
0	1	nCS (EBI)															
1	0	SCL1(I <sup>2</sup> C1) (只有M05xxDN/DE 支持)															
1	1	保留															
[11]	P4_ALT[3]	<p><b>P4.3 复用功能选择</b> P4.3的功能取决于P4_MFP[3] and P4_ALT[3].</p> <table border="1"> <thead> <tr> <th>P4_ALT[3]</th><th>P4_MFP[3]</th><th>P4.3的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P4.3</td></tr> <tr> <td>0</td><td>1</td><td>PWM3(PWMA 通道3)</td></tr> <tr> <td>1</td><td>x</td><td>保留</td></tr> </tbody> </table>	P4_ALT[3]	P4_MFP[3]	P4.3的功能	0	0	P4.3	0	1	PWM3(PWMA 通道3)	1	x	保留			
P4_ALT[3]	P4_MFP[3]	P4.3的功能															
0	0	P4.3															
0	1	PWM3(PWMA 通道3)															
1	x	保留															
[10]	P4_ALT[2]	<p><b>P4.2 复用功能选择</b> P4.2的功能取决于P4_MFP[2] and P4_ALT[2].</p> <table border="1"> <thead> <tr> <th>P4_ALT[2]</th><th>P4_MFP[2]</th><th>P4.2的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P4.2</td></tr> <tr> <td>0</td><td>1</td><td>PWM2(PWMA 通道2)</td></tr> <tr> <td>1</td><td>x</td><td>保留</td></tr> </tbody> </table>	P4_ALT[2]	P4_MFP[2]	P4.2的功能	0	0	P4.2	0	1	PWM2(PWMA 通道2)	1	x	保留			
P4_ALT[2]	P4_MFP[2]	P4.2的功能															
0	0	P4.2															
0	1	PWM2(PWMA 通道2)															
1	x	保留															

[9]	P4_ALT[1]	<p><b>P4.1 复用功能选择</b></p> <p>P4.1的功能取决于P4_MFP[1] and P4_ALT[1].</p> <table border="1"> <thead> <tr> <th>P4_ALT[1]</th><th>P4_MFP[1]</th><th>P4.1的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P4.1</td></tr> <tr> <td>0</td><td>1</td><td>PWM1(PWMA 通道1)</td></tr> <tr> <td>1</td><td>0</td><td>T3EX</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P4_ALT[1]	P4_MFP[1]	P4.1的功能	0	0	P4.1	0	1	PWM1(PWMA 通道1)	1	0	T3EX	1	1	保留
P4_ALT[1]	P4_MFP[1]	P4.1的功能															
0	0	P4.1															
0	1	PWM1(PWMA 通道1)															
1	0	T3EX															
1	1	保留															
[8]	P4_ALT[0]	<p><b>P4.0 复用功能选择</b></p> <p>P4.0的功能取决于P4_MFP[0] and P4_ALT[0].</p> <table border="1"> <thead> <tr> <th>P4_ALT[0]</th><th>P4_MFP[0]</th><th>P4.0的功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>P4.0</td></tr> <tr> <td>0</td><td>1</td><td>PWM0(PWMA 通道0)</td></tr> <tr> <td>1</td><td>0</td><td>T2EX</td></tr> <tr> <td>1</td><td>1</td><td>保留</td></tr> </tbody> </table>	P4_ALT[0]	P4_MFP[0]	P4.0的功能	0	0	P4.0	0	1	PWM0(PWMA 通道0)	1	0	T2EX	1	1	保留
P4_ALT[0]	P4_MFP[0]	P4.0的功能															
0	0	P4.0															
0	1	PWM0(PWMA 通道0)															
1	0	T2EX															
1	1	保留															
[7:0]	P4_MFP[7:0]	<p><b>P4 复用功能选择</b></p> <p>P4的功能取决于P4_MFP 和 P4_ALT.</p> <p>参考P4_ALT 的详细描述.</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，这些系统控制寄存器在上电复位后是被保护的，直到用户关闭寄存器保护。用户如果要编程这些被保护的寄存器，需要写一个解锁序列到REGWRPROT寄存器。解锁序列就是连续写“59h”，“16h”，“88h”到REGWRPROT寄存器，地址是0x5000\_010。在写这三个数据过程中，任何不同的数据值，不同的顺序或者任何对其他地址的写，都会中止解锁序列。

保护被关闭之后，用户可以检查保护禁止位（地址0x5000\_0100的比特0），"1"表示保护关闭，"0"表示保护使能。然后用户可以更新被保护的目标寄存器的值，并随时可以通过写任何其它数据到地址"0x5000\_0100"来使能寄存器保护。

写该寄存器来关闭/使能寄存器保护，读该寄存器来获取REGPROTDIS状态。

寄存器	偏移量	R/W	描述	复位后的值
REGWRPROT	GCR_BA+100	R/W	寄存器写保护控制寄存器	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
REGWRPROT[7:1]							REGWRPROT[0] REGPROTDIS

Bits	描述	
[31:16]	保留	保留
[7:0]	REGWRPROT	寄存器写保护码(只写) 一些寄存器有写保护功能，编程这些寄存器之前必须通过一个解锁序列来关闭写保护功能，解锁序列就是依次写“59h”，“16h”，“88h”到这个域，这个序列完成之后，REGPROTDIS位将被置1，写保护寄存器可以正常写入数据。

[0]	REGPROTDIS	<b>寄存器写保护(只读)</b>																																					
		1 = 解除写保护, 写保护寄存器可以正常编程																																					
		0 = 使能寄存器的写保护, 任何写受保护寄存器的操作将被忽略.																																					
		受保护的寄存器有:																																					
		<table border="1"> <thead> <tr> <th>Registers</th><th>Address</th><th>Note</th></tr> </thead> <tbody> <tr> <td>IPRSTC1</td><td>0x5000_0008</td><td></td></tr> <tr> <td>BODCR</td><td>0x5000_0018</td><td></td></tr> <tr> <td>PORCR</td><td>0x5000_001C</td><td></td></tr> <tr> <td>PWRCON</td><td>0x5000_0200</td><td>在清除电源唤醒中断时bit[6]不被保护</td></tr> <tr> <td>APBCLK bit[0]</td><td>0x5000_0208</td><td>bit[0]: 看门狗时钟使能</td></tr> <tr> <td>CLKSEL0</td><td>0x5000_0210</td><td>HCLK和CPU STCLK时钟源选择</td></tr> <tr> <td>CLKSEL1 bit[1:0]</td><td>0x5000_0214</td><td>看门狗时钟源选择</td></tr> <tr> <td>NMI_SEL bit[8]</td><td>0x5000_0380</td><td>NMI 中断使能</td></tr> <tr> <td>ISPCON</td><td>0x5000_C000</td><td>Flash ISP控制寄存器</td></tr> <tr> <td>ISPTRG</td><td>0x5000_C010</td><td>ISP触发控制寄存器</td></tr> <tr> <td>WTCR</td><td>0x4000_4000</td><td>看门狗时钟控制寄存器</td></tr> <tr> <td>FATCON</td><td>0x5000_C018</td><td>Flash访问时间控制寄存器</td></tr> </tbody> </table>	Registers	Address	Note	IPRSTC1	0x5000_0008		BODCR	0x5000_0018		PORCR	0x5000_001C		PWRCON	0x5000_0200	在清除电源唤醒中断时bit[6]不被保护	APBCLK bit[0]	0x5000_0208	bit[0]: 看门狗时钟使能	CLKSEL0	0x5000_0210	HCLK和CPU STCLK时钟源选择	CLKSEL1 bit[1:0]	0x5000_0214	看门狗时钟源选择	NMI_SEL bit[8]	0x5000_0380	NMI 中断使能	ISPCON	0x5000_C000	Flash ISP控制寄存器	ISPTRG	0x5000_C010	ISP触发控制寄存器	WTCR	0x4000_4000	看门狗时钟控制寄存器	FATCON
Registers	Address	Note																																					
IPRSTC1	0x5000_0008																																						
BODCR	0x5000_0018																																						
PORCR	0x5000_001C																																						
PWRCON	0x5000_0200	在清除电源唤醒中断时bit[6]不被保护																																					
APBCLK bit[0]	0x5000_0208	bit[0]: 看门狗时钟使能																																					
CLKSEL0	0x5000_0210	HCLK和CPU STCLK时钟源选择																																					
CLKSEL1 bit[1:0]	0x5000_0214	看门狗时钟源选择																																					
NMI_SEL bit[8]	0x5000_0380	NMI 中断使能																																					
ISPCON	0x5000_C000	Flash ISP控制寄存器																																					
ISPTRG	0x5000_C010	ISP触发控制寄存器																																					
WTCR	0x4000_4000	看门狗时钟控制寄存器																																					
FATCON	0x5000_C018	Flash访问时间控制寄存器																																					
注: 在相应寄存器的描述中, 写保护的bit有标注"(写保护)"																																							

### 6.2.7 系统定时器(SysTick)

Cortex-M0 包含一个集成的系统定时器，SysTick。SysTick 提供一种简单的，24位写清零，下数计数，计数至0后自装载的计数器，有一个灵活的控制机制。计数器可作为实时操作系统的节拍定时器或者作为一个简单的计数器。

使能后，系统定时器从SysTick 当前值寄存器(SYST\_CVR)的值向下计数到0，并在下一个时钟边沿，重新加载SysTick重装载值寄存器(SYST\_RVR)的值到SysTick当前值寄存器(SYST\_CVR)，然后随接下来的时钟递减。当计数器减到0时，标志位COUNTFLAG置位，标志位COUNTFLAG是读清0的。

复位后，SYST\_CVR 的值未知。使能前，软件应该写该寄存器使其清0。这样确保定时器在使能后以 SYST\_RVR中的值计数，而非任意值。

若SYST\_RVR 是0，在重新加载后，定时器将保持当前值0，这种机制可以用来在不使用系统定时器的使能位的情形下禁用系统定时器。

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

### 6.2.8 系统定时器控制寄存器映射

**R:** 只读, **W:** 只写, **R/W:** 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>SYST 基地址:</b>				
<b>SYST_BA = 0xE000_E010</b>				
<b>SYST_CSR</b>	SYST_BA+ 0x00	R/W	SysTick控制与状态寄存器	0x0000_0000
<b>SYST_RVR</b>	SYST_BA + 0x04	R/W	SysTick重装载值寄存器	0xFFFF_FFFF
<b>SYST_CVR</b>	SYST_BA + 0x08	R/W	SysTick 当前值寄存器	0xFFFF_FFFF

SysTick 控制与状态 (SYST\_CSR)

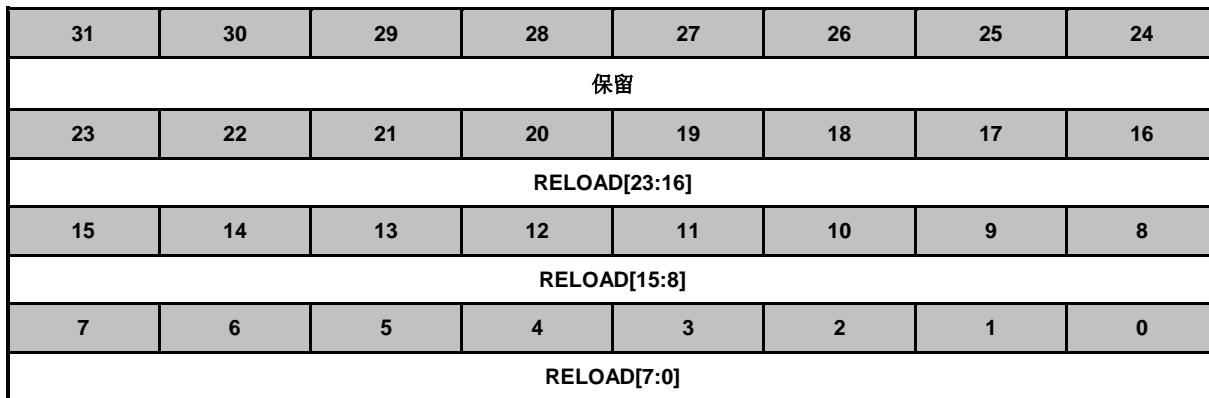
寄存器	偏移量	R/W	描述	复位后的值
SYST_CSR	SYST_BA+0x00	R/W	SysTick控制与状态寄存器	0x0000_0004

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					CLKSRC	TICKINT	ENABLE

Bits	描述	
[31:17]	保留	保留
[16]	<b>COUNTFLAG</b>	系统节拍器计数标志 从上次该寄存器被读，如果定时器计数到0，则返回1。 计数由1到0时，COUNTFLAG 置位。 读该位或向当前值寄存器（SYST_CVR）写时，COUNTFLAG 被清零。
[15:3]	保留	保留
[2]	<b>CLKSRC</b>	系统节拍器时钟源选择 1= 内核时钟用于SysTick. 0= 时钟源可选，参考 <a href="#">STCLK_S</a> .
[1]	<b>TICKINT</b>	系统节拍器中断使能 1：向下计数到0将导致SysTick 中断发生。通过软件寄存器写操作清SysTick 当前值寄存器的值将不会导致SysTick 发生中断。 0：向下计数到0不会引起SysTick发生中断。软件可以使用COUNTFLAG 来确定是否已经发生计数到0
[0]	<b>ENABLE</b>	系统节拍器计数器使能 1：计数器开始计数，并运行于周期模式。 0：禁用计数器

SysTick重装载值寄存器 (SYST\_RVR)

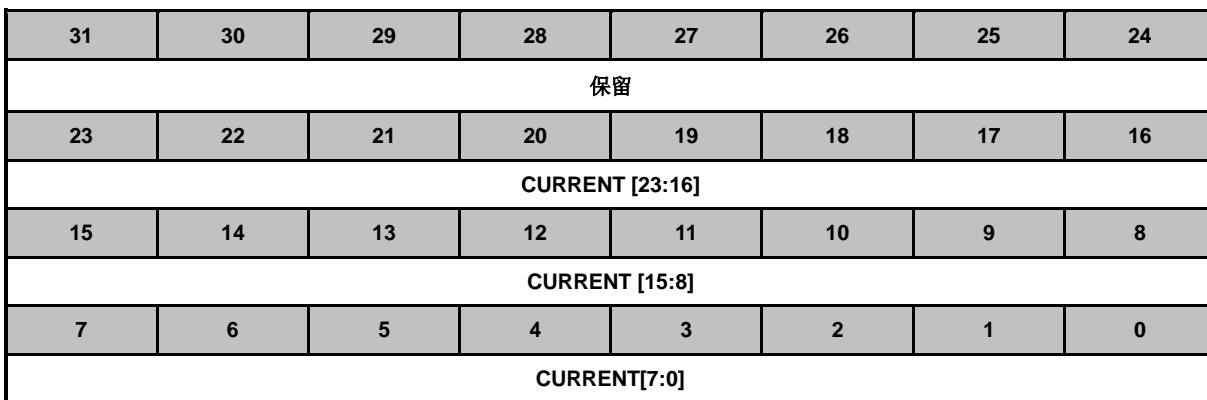
寄存器	偏移量	R/W	描述	复位后的值
SYST_RVR	SYST_BA + 0x04	R/W	SysTick重装载值寄存器	0xFFFF_FFFF



Bits	描述	
[31:24]	保留	保留
[23:0]	RELOAD	系统节拍器重装载值 当计数器达到0时，该值加载到当前值寄存器。

SysTick当前值寄存器 (SYST\_CVR)

寄存器	偏移量	R/W	描述	复位后的值
SYST_CVR	SYST_BA + 0x08	R/W	SysTick当前值寄存器	0xFFFF_FFFF



Bits	描述	
[31:24]	保留	保留
[23:0]	CURRENT	<p><b>系统节拍器当前值</b></p> <p>当前计数值，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，该寄存器为写清零的，软件写入任何值将清寄存器为0。这些位不支持读为零(read as zero)，参见系统重装载值寄存器 (SYST_RVR)。</p>

### 6.2.9 嵌套向量中断控制器 (NVIC)

Cortex-M0提供中断控制器，作为异常模式的组成部分，称之为“嵌套向量中断控制器(NVIC)”。它与处理器内核紧密联系，并具有以下特性：

- 支持嵌套和向量中断
- 自动保存和恢复上下文
- 可动态改变优先级
- 简化的精确的中断延迟

NVIC对所有支持的异常按优先级排序并处理，所有异常在“处理模式”处理。 NVIC结构支持具有四级优先级的32个(IRQ[31:0])离散中断。所有的中断和大多数系统异常可以配置为不同的优先级。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于当前中断，则新中断将代替当前中断被处理。

当任何中断被响应时，中断服务程序（ISR）的起始地址从内存的向量表中取得。不需要由软件确定响应哪个中断，也不要软件跳转到相应ISP的起始地址。当取得起始地址时，NVIC将自动保存处理器状态，包括以下寄存器“PC, PSR, LR, R0~R3, R12”的值到栈中。在ISR结束时，NVIC 将从栈中恢复相关寄存器的值，恢复正常操作，因此处理器将花费更少的并且确定的时间去处理中断请求。

NVIC支持末尾链接“Tail Chaining”，有效处理尾对尾中断“back-to-back interrupts”，即无需重复保存和恢复当前状态从而减少从当前ISR结束切换到等待处理的ISR的延迟时间。NVIC还支持晚到“Late Arrival”，可以提升同时发生的ISR的效率。在当前ISR开始执行（保存处理器状态并获取起始地址阶段）之前如果较高优先级中断请求发生，NVIC将立即选择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”。

#### 6.2.9.1 异常模式和系统中断映射

下表列出了NuMicro™ M051系列支持的异常模型。软件可以对其中一些异常以及所有中断设置4级优先级。最高用户可配置优先级记为“0”，最低优先级记为“3”，所有用户可配置的优先级的默认值为“0”。注意：优先级“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
保留	4 ~ 10	保留
SVCALL	11	可配置

保留	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 异常模型

异常号	向量地址	中断号 (Bit In Interrupt Registers)	中断名	源IP	中断描述	掉电唤醒
1-15	0x00-0x3C	-	-	-	系统异常	
16	0x40	0	BOD_OUT	Brown-Out	欠压检测中断	Yes
17	0x44	1	WDT_INT/WWDT_INT	WDT	看门狗/窗看门狗定时器中断	Yes
18	0x48	2	EINT0	GPIO	P3.2 脚上的外部信号中断	Yes
19	0x4C	3	EINT1	GPIO	P3.3 脚上的外部信号中断	Yes
20	0x50	4	GP01_INT	GPIO	P0[7:0] / P1[7:0] 外部信号中断	Yes
21	0x54	5	GP234_INT	GPIO	P2[7:0]/P3[7:0]/P4[7:0] 外部信号中断，除 P32 和 P33	Yes
22	0x58	6	PWMA_INT	PWM0~3	PWM0, PWM1, PWM2 和 PWM3 中断	No
23	0x5C	7	PWMB_INT	PWM4~7	PWM4, PWM5, PWM6 和 PWM7 中断	No
24	0x60	8	TMR0_INT	TMR0	Timer 0 中断	No
25	0x64	9	TMR1_INT	TMR1	Timer 1 中断	No
26	0x68	10	TMR2_INT	TMR2	Timer 2 中断	No
27	0x6C	11	TMR3_INT	TMR3	Timer 3 中断	No
28	0x70	12	UART0_INT	UART0	UART0 中断	Yes
29	0x74	13	UART1_INT	UART1	UART1 中断	Yes
30	0x78	14	SPI0_INT	SPI0	SPI0 中断	No
31	0x7C	15	SPI1_INT	SPI1	SPI1 中断	No
32-33	0x80-0x84	16-17	预留	-	-	-
34	0x88	18	I2C0_INT	I <sup>2</sup> C0	I <sup>2</sup> C0 中断	No(M05xxBN) Yes(M05xxDN/DE)
35	0x8C	19	I2C1_INT	I <sup>2</sup> C1	I <sup>2</sup> C1 中断(只有M05xxDN/DE 支持)	
36-40	0x90-0xA0	20-24	预留	-	-	-
41	0xA4	25	ACMP01_INT	ACMP0/1	模拟比较器0和模拟比较器1的中断	Yes

42	0xA8	26	ACMP23_INT	ACMP2/3	模拟比较器2和模拟比较器3的中断(只有M05xxDN/DE支持)	Yes
43	0xAC	27	预留			
44	0xB0	28	PWRWU_INT	CLKC	从掉电状态唤醒的时钟控制器中断	Yes
45	0xB4	29	ADC_INT	ADC	ADC 中断	No
46-47	0xB8-0xBC	30-31	-	预留	-	

表6-3 系统中断映射向量表

#### 6.2.9.2 向量表

当任何中断被响应时，处理器会自动从内存的向量表中获取中断服务程序（ISR）的起始地址。对于ARMv6-M，向量表的基地址固定在0x00000000。向量表包括复位后栈指针的初始值，和所有异常处理函数的入口地址。上一节的向量号(异常号)决定了异常处理函数在向量表中的入口顺序。

向量表字偏移量	描述
0	SP_main -主堆栈指针
Vector Number	异常入口指针，用向量号表示

表6-4 向量表格式

#### 6.2.9.3 操作描述

通过写相应中断使能设置寄存器或清使能寄存器位域，可以使能NVIC中断或禁用NVIC中断，这些寄存器通过写1使能和写1清除为零，读取这两个寄存器均返回当前相应中断的使能状态。当某一个中断被禁用时，中断声明将使该中断处于等待处理状态，然而，该中断不会被激活。如果某一个中断在被禁用时处于激活状态，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。清使能位可以阻止相关中断被再次激活。

NVIC中断可以使用互补的寄存器对来挂起/解除挂起以使能/禁用这些中断，这些寄存器分别为Set-Pending寄存器与Clear-Pending寄存器，这些寄存器使用写1使能和写1清除的方式，读取这两种寄存器都返回相应中断的当前挂起状态。Clear-Pending寄存器不会对处于激活状态的中断的执行状态产生任何影响。

NVIC中断通过更新32位寄存器中的各个8位字段（每个寄存器支持4个中断）来分配中断的优先级。

与NVIC相关的通用寄存器都可以通过系统控制空间的内存区域访问，下一节将作出描述。

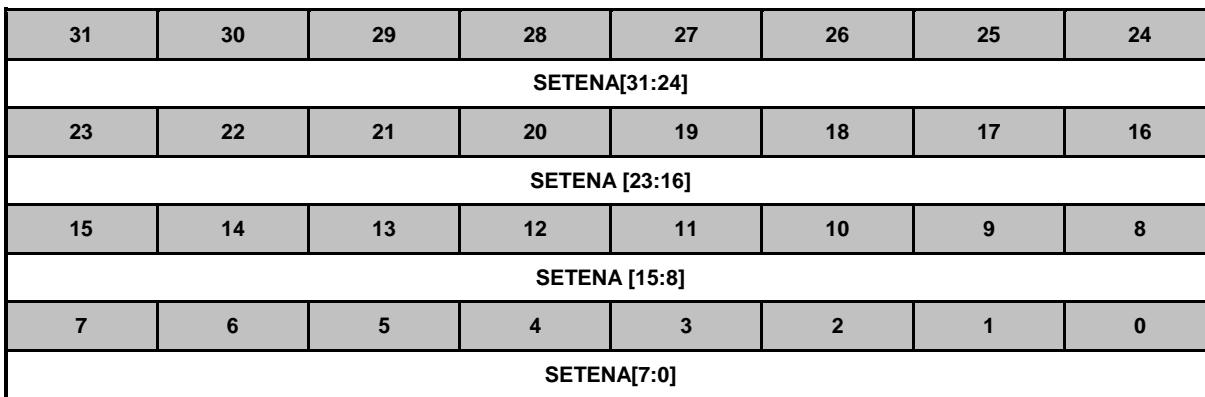
## 6.2.9.4 NVIC 控制寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>NVIC 基址:</b>				
<b>NVIC_BA = 0xE000_E100</b>				
<b>NVIC_ISET</b>	NVIC_BA + 0x000	R/W	IRQ0 ~ IRQ31 设置使能控制寄存器	0x0000_0000
<b>NVIC_ICER</b>	NVIC_BA + 0x080	R/W	IRQ0 ~ IRQ31 清使能控制寄存器	0x0000_0000
<b>NVIC_ISPR</b>	NVIC_BA + 0x100	R/W	IRQ0 ~ IRQ31 设置挂起控制寄存器	0x0000_0000
<b>NVIC_ICPR</b>	NVIC_BA + 0x180	R/W	IRQ0 ~ IRQ31 清挂起控制寄存器	0x0000_0000
<b>NVIC_IPR0</b>	NVIC_BA + 0x300	R/W	IRQ0 ~ IRQ3 优先级 控制寄存器	0x0000_0000
<b>NVIC_IPR1</b>	NVIC_BA + 0x304	R/W	IRQ4 ~ IRQ7优先级控制寄存器	0x0000_0000
<b>NVIC_IPR2</b>	NVIC_BA + 0x308	R/W	IRQ8 ~ IRQ11优先级控制寄存器	0x0000_0000
<b>NVIC_IPR3</b>	NVIC_BA + 0x30C	R/W	IRQ12 ~ IRQ15优先级控制寄存器	0x0000_0000
<b>NVIC_IPR4</b>	NVIC_BA + 0x310	R/W	IRQ16 ~ IRQ19优先级控制寄存器	0x0000_0000
<b>NVIC_IPR5</b>	NVIC_BA + 0x314	R/W	IRQ20 ~ IRQ23优先级控制寄存器	0x0000_0000
<b>NVIC_IPR6</b>	NVIC_BA + 0x318	R/W	IRQ24 ~ IRQ27优先级控制寄存器	0x0000_0000
<b>NVIC_IPR7</b>	NVIC_BA + 0x31C	R/W	IRQ28 ~ IRQ31优先级控制寄存器	0x0000_0000

IRQ0 ~ IRQ31设置使能控制寄存器 (NVIC\_ISER)

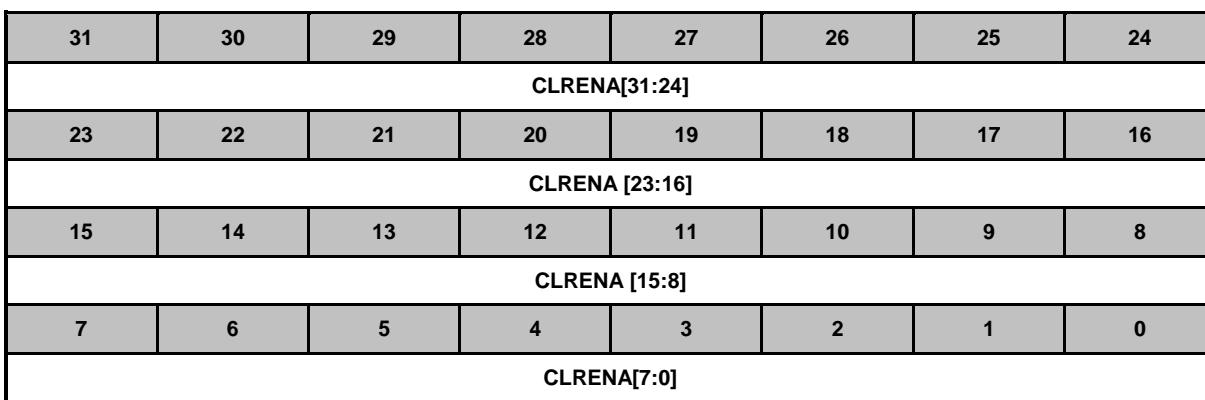
寄存器	偏移量	R/W	描述	复位的值
NVIC_ISER	NVIC_BA + 0x000	R/W	IRQ0 ~ IRQ31设置使能控制寄存器	0x0000_0000



Bits	描述	
[31:0]	SETENA	<p><b>中断使能寄存器</b>          使能1个或多个中断，每位代表从IRQ0 ~ IRQ31的中断号(向量号：16 ~ 47).          写：          1 = 使能相关中断          0 = 无效          读：          1 = 相应中断是使能的          0 = 相应中断是禁止的          寄存器读取返回当前使能状态.</p>

IRQ0 ~ IRQ31清使能控制寄存器 (NVIC\_ICER)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ICER	NVIC_BA + 0x080	R/W	IRQ0 ~ IRQ31清使能控制寄存器	0x0000_0000



Bits	描述	
[31:0]	CLRENA	<p><b>中断禁止寄存器</b></p> <p>禁用1个或多个中断，每位代表从IRQ0 ~ IRQ31的中断号 (向量号： 16 ~ 47).</p> <p>写：</p> <p>1 = 禁用相应中断 0 = 无效</p> <p>读：</p> <p>1 = 相应中断时使能的 0 = 相应中断时禁止的</p> <p>寄存器读取返回当前使能状态.</p>

IRQ0 ~ IRQ31设置挂起控制寄存器 (NVIC\_ISPR)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ISPR	NVIC_BA + 0x100	R/W	IRQ0 ~ IRQ31设置挂起控制寄存器	0x0000_0000

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

Bits	描述
[31:0]	<b>SETPEND</b> 设置中断挂起寄存器 软件写1，挂起相应中断。每位代表从IRQ0 ~ IRQ31 的中断号(向量号： 16 ~ 47). 写0无效 寄存器读取返回当前挂起状态

IRQ0 ~ IRQ31清挂起控制寄存器 (NVIC\_ICPR)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ICPR	NVIC_BA + 0x180	R/W	IRQ0 ~ IRQ31清挂起控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	<b>CLRPEND</b>	<p><b>清中断挂起寄存器</b>            写1清除挂起的相应中断，每位代表从IRQ0 ~ IRQ31的中断号 (向量号： 16 ~ 47).            写0无效.            寄存器读取返回当前挂起状态.</p>

IRQ0 ~ IRQ3中断优先级寄存器 (NVIC\_IPR0)

寄存器	偏移量	R/W	描述				复位后的值
NVIC_IPR0	NVIC_BA + 0x300	R/W	IRQ0 ~ IRQ3中断优先级寄存器				0x0000_0000

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

Bits	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
[23:22]	PRI_2	IRQ2优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_1	IRQ1优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_0	IRQ0优先级 “0”表示最高优先级& “3”表示最低优先级

IRQ4 ~ IRQ7中断优先级寄存器 (NVIC\_IPR1)

寄存器	偏移量	R/W	描述				复位后的值
NVIC_IPR1	NVIC_BA + 0x304	R/W	IRQ4 ~ IRQ7中断优先级寄存器				0x0000_0000

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

Bits	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
[23:22]	PRI_6	IRQ6优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_5	IRQ5优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_4	IRQ4优先级 “0”表示最高优先级& “3”表示最低优先级

IRQ8 ~ IRQ11中断优先级寄存器 (NVIC\_IPR2)

寄存器	偏移量	R/W	描述					复位后的值
NVIC_IPR2	NVIC_BA + 0x308	R/W	IRQ8 ~ IRQ11中断优先级寄存器					0x0000_0000

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

Bits	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
[23:22]	PRI_10	IRQ10优先级 “0”表示最高优先级&“3”表示最低优先级
[15:14]	PRI_9	IRQ9优先级 “0”表示最高优先级&“3”表示最低优先级
[7:6]	PRI_8	IRQ8优先级 “0”表示最高优先级&“3”表示最低优先级

IRQ12 ~ IRQ15中断优先级寄存器 (NVIC\_IPR3)

寄存器	偏移量	R/W	描述					复位后的值
NVIC_IPR3	NVIC_BA + 0x30C	R/W	IRQ12 ~ IRQ15中断优先级寄存器					0x0000_0000

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

Bits	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
[23:22]	PRI_14	IRQ14优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_13	IRQ13优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_12	IRQ12优先级 “0”表示最高优先级& “3”表示最低优先级

IRQ16 ~ IRQ19中断优先级寄存器 (NVIC\_IPR4)

寄存器	偏移量	R/W	描述					复位后的值
NVIC_IPR4	NVIC_BA + 410	R/W	IRQ16 ~ IRQ19中断优先级寄存器					0x0000_0000

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

Bits	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
[23:22]	PRI_18	IRQ18优先级 “0”表示最高优先级&“3”表示最低优先级
[15:14]	PRI_17	IRQ17优先级 “0”表示最高优先级&“3”表示最低优先级
[7:6]	PRI_16	IRQ16优先级 “0”表示最高优先级&“3”表示最低优先级

IRQ20 ~ IRQ23中断优先级寄存器 (NVIC\_IPR5)

寄存器	偏移量	R/W	描述					复位后的值
NVIC_IPR5	NVIC_BA + 0x314	R/W	IRQ20 ~ IRQ23中断优先级寄存器					0x0000_0000

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

Bits	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
[23:22]	PRI_22	IRQ22优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_21	IRQ21优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_20	IRQ20优先级 “0”表示最高优先级& “3”表示最低优先级

IRQ24 ~ IRQ27中断优先级寄存器 (NVIC\_IPR6)

寄存器	偏移量	R/W	描述					复位后的值
NVIC_IPR6	NVIC_BA + 0x318	R/W	IRQ24 ~ IRQ27中断优先级寄存器					0x0000_0000

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

Bits	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
[23:22]	PRI_26	IRQ26优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_25	IRQ25优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_24	IRQ24优先级 “0”表示最高优先级& “3”表示最低优先级

IRQ28 ~ IRQ31中断优先级寄存器 (NVIC\_IPR7)

寄存器	偏移量	R/W	描述					复位后的值
NVIC_IPR7	NVIC_BA + 0x31C	R/W	IRQ28 ~ IRQ31中断优先级寄存器					0x0000_0000

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

Bits	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
[23:22]	PRI_30	IRQ30优先级 “0”表示最高优先级& “3”表示最低优先级
[15:14]	PRI_29	IRQ29优先级 “0”表示最高优先级& “3”表示最低优先级
[7:6]	PRI_28	IRQ28优先级 “0”表示最高优先级& “3”表示最低优先级

### 6.2.9.5 中断源控制寄存器

除了与NVIC相关的中断控制寄存器外，NuMicro™ M051系列还有一些特殊控制寄存器以利于中断功能，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:** 可读写, **W&C:** 写1清除为零

寄存器	偏移量	R/W	描述	复位后的值
<b>INT基地址</b>				
<b>INT_BA = 0x5000_0300</b>				
<b>IRQ0_SRC</b>	INT_BA+0x00	R	IRQ0 (BOD) 中断源识别	0XXXXX_XXXX
<b>IRQ1_SRC</b>	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别	0XXXXX_XXXX
<b>IRQ2_SRC</b>	INT_BA+0x08	R	IRQ2 ((EINT0) 中断源识别	0XXXXX_XXXX
<b>IRQ3_SRC</b>	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别	0XXXXX_XXXX
<b>IRQ4_SRC</b>	INT_BA+0x10	R	IRQ4 (P0/1) 中断源识别	0XXXXX_XXXX
<b>IRQ5_SRC</b>	INT_BA+0x14	R	IRQ5 (P2/3/4) 中断源识别	0XXXXX_XXXX
<b>IRQ6_SRC</b>	INT_BA+0x18	R	IRQ6 (PWMA) 中断源识别	0XXXXX_XXXX
<b>IRQ7_SRC</b>	INT_BA+0x1C	R	IRQ7 (PWMB) 中断源识别	0XXXXX_XXXX
<b>IRQ8_SRC</b>	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别	0XXXXX_XXXX
<b>IRQ9_SRC</b>	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别	0XXXXX_XXXX
<b>IRQ10_SRC</b>	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别	0XXXXX_XXXX
<b>IRQ11_SRC</b>	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别	0XXXXX_XXXX
<b>IRQ12_SRC</b>	INT_BA+0x30	R	IRQ12 (URTO) 中断源识别	0XXXXX_XXXX
<b>IRQ13_SRC</b>	INT_BA+0x34	R	IRQ13 (URT1) 中断源识别	0XXXXX_XXXX
<b>IRQ14_SRC</b>	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别	0XXXXX_XXXX
<b>IRQ15_SRC</b>	INT_BA+0x3C	R	IRQ15 (SPI1) 中断源识别	0XXXXX_XXXX
<b>IRQ16_SRC</b>	INT_BA+0x40	保留	保留	0XXXXX_XXXX
<b>IRQ17_SRC</b>	INT_BA+0x44	保留	保留	0XXXXX_XXXX
<b>IRQ18_SRC</b>	INT_BA+0x48	R	IRQ18 (I <sup>2</sup> C0) 中断源识别	0XXXXX_XXXX
<b>IRQ19_SRC</b>	INT_BA+0x4C	R	IRQ19 (I <sup>2</sup> C1) 中断源识别(只有M05xxDN/DE 支持)	0XXXXX_XXXX
<b>IRQ20_SRC</b>	INT_BA+0x50	保留	保留	0XXXXX_XXXX
<b>IRQ21_SRC</b>	INT_BA+0x54	保留	保留	0XXXXX_XXXX
<b>IRQ22_SRC</b>	INT_BA+0x58	保留	保留	0XXXXX_XXXX

<b>IRQ23_SRC</b>	INT_BA+0x5C	保留	保留	0xXXXX_XXXX
<b>IRQ24_SRC</b>	INT_BA+0x60	保留	保留	0xXXXX_XXXX
<b>IRQ25_SRC</b>	INT_BA+0x64	保留	IRQ25 (ACMP01) 中断源识别	0xXXXX_XXXX
<b>IRQ26_SRC</b>	INT_BA+0x68	保留	IRQ26 (ACMP23) 中断源识别(只有M05xxDN/DE 支持)	0xXXXX_XXXX
<b>IRQ27_SRC</b>	INT_BA+0x6C	保留	保留	0xXXXX_XXXX
<b>IRQ28_SRC</b>	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源识别	0xXXXX_XXXX
<b>IRQ29_SRC</b>	INT_BA+0x74	R	IRQ29 (ADC) 中断源识别	0xXXXX_XXXX
<b>IRQ30_SRC</b>	INT_BA+0x78	保留	保留	0xXXXX_XXXX
<b>IRQ31_SRC</b>	INT_BA+0x7C	保留	保留	0xXXXX_XXXX
<b>NMI_SEL</b>	INT_BA+0x80	R/W	NMI中断源选择控制寄存器	0x0000_0000
<b>MCU_IRQ</b>	INT_BA+0x84	R/W	MCU IRQ号识别寄存器	0x0000_0000

中断源识别寄存器(IRQn\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQn_SRC	INT_BA+0x00 ..... INT_BA+0x7C	R	MCU IRQ0 (BOD) 中断源识别 ： MCU IRQ31 (保留) 中断源识别	0XXXX_XXXX

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				INT_SRC[3]	INT_SRC[2:0]		

地址	INT-Num	Bits	描述
INT_BA+0x00	0	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: BOD_INT
INT_BA+0x04	1	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: WDT_INT
INT_BA+0x08	2	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: EINT0 -P3.2引脚上的外部中断0
INT_BA+0x0C	3	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: EINT1 - P3.3上的外部中断1
INT_BA+0x10	4	[2:0]	Bit2: 1'b0 Bit1: P1_INT Bit0: P0_INT
INT_BA+0x14	5	[2:0]	Bit2: P4_INT Bit1: P3_INT Bit0: P2_INT
INT_BA+0x18	6	[3:0]	Bit3: PWM3_INT Bit2: PWM2_INT Bit1: PWM1_INT Bit0: PWM0_INT

INT_BA+0x1C	7	[3:0]	Bit3: PWM7_INT Bit2: PWM6_INT Bit1: PWM5_INT Bit0: PWM4_INT
INT_BA+0x20	8	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: TMR0_INT
INT_BA+0x24	9	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: TMR1_INT
INT_BA+0x28	10	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: TMR2_INT
INT_BA+0x2C	11	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: TMR3_INT
INT_BA+0x30	12	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: URT0_INT
INT_BA+0x34	13	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: URT1_INT
INT_BA+0x38	14	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: SPI0_INT
INT_BA+0x3C	15	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: SPI1_INT
INT_BA+0x40	16	[2:0]	保留
INT_BA+0x44	17	[2:0]	保留
INT_BA+0x48	18	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: I2C_INT
INT_BA+0x4C	19	[2:0]	保留
INT_BA+0x50	20	[2:0]	保留
INT_BA+0x54	21	[2:0]	保留
INT_BA+0x58	22	[2:0]	保留
INT_BA+0x5C	23	[2:0]	保留
INT_BA+0x60	24	[2:0]	保留
INT_BA+0x64	25	[2:0]	保留

INT_BA+0x68	26	[2:0]	保留
INT_BA+0x6C	27	[2:0]	保留
INT_BA+0x70	28	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: PWRWU_INT
INT_BA+0x74	29	[2:0]	Bit2: 1'b0 Bit1: 1'b0 Bit0: ADC_INT
INT_BA+0x78	30	[2:0]	保留
INT_BA+0x7C	31	[2:0]	保留

IRQ0 (BOD) 中断源识别寄存器(IRQ0\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述	
[2:0]	INT_SRC	<b>IRQ0 中断源识别</b> INT_SRC[2] = 保留. INT_SRC[1] = 保留. INT_SRC[0]: 0 = IRQ0 中断源不是来自 BOD 中断 (BOD_INT). 1 = IRQ0 中断源来自 BOD 中断 (BOD_INT). 注: 当中断标志被清除时, 该位将被自动清除.

IRQ1 (WDT) 中断源识别寄存器(IRQ1\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述	
[2:0]	INT_SRC	<p><b>IRQ1中断源识别</b></p> <p>INT_SRC[2] = 保留..</p> <p>INT_SRC[1] = 保留..</p> <p>INT_SRC[0]:</p> <p>0 = IRQ1 中断源不是来自看门狗中断 (WDT_INT).</p> <p>1 = IRQ1 中断源来自看门狗中断(WDT_INT).</p> <p><b>注:</b> 当中断标志被清除时, 该位将被自动清除</p>

IRQ2 (EINT0) 中断源识别寄存器(IRQ2\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ2中断源识别</b> INT_SRC[2] = 保留.. INT_SRC[1] = 保留.. INT_SRC[0]: 0 = IRQ2中断源不是来自外部中断0 – P3.2 (EINT0). 1 = IRQ2中断源是来自外部中断0 – P3.2 (EINT0). <b>注:</b> 当中断标志被清除时，该位将被自动清除.

IRQ3 (EINT1) 中断源识别寄存器(IRQ3\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述	
[2:0]	INT_SRC	<p><b>IRQ3中断源识别</b></p> <p>INT_SRC[2] = 保留..</p> <p>INT_SRC[1] = 保留..</p> <p>INT_SRC[0]:</p> <p>0 = IRQ3中断源不是来自外部中断1 – P3.3 (EINT1).</p> <p>1 = IRQ3中断源是来自外部中断1 – P3.3 (EINT1).</p> <p><b>注:</b> 当中断标志被清除时，该位将被自动清除.</p>

IRQ4 (P0/1) 中断源识别寄存器(IRQ4\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (P0/1) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ4中断源识别</b> INT_SRC[2] =保留.. INT_SRC[1]: 0 = IRQ4中断源不是来自P1中断 (P1_INT). 1 = IRQ4中断源来自P1中断 (P1_INT). INT_SRC[0]: 0 = IRQ4中断源不是来自P0中断 (P0_INT). 1 = IRQ4中断源来自P0中断(P0_INT). <b>注1:</b> 同一时刻，IRQ4 中断可以来自多个中断源. <b>注2:</b> 当中断标志被清除时，该位将被自动清除.

IRQ5 (P2/3/4) 中断源识别寄存器(IRQ5\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (P2/3/4) 中断源识别	0xXXXX_XXXX



Bits	描述
[2:0]	<b>IRQ5中断源识别</b> <b>INT_SRC[2]:</b> 0 = IRQ5中断源不是来自P4中断(P4_INT). 1 = IRQ5中断源来自P4中断(P4_INT). <b>INT_SRC[1]:</b> 0 = IRQ5中断源不是来自P3中断 (P3_INT). 1 = IRQ5中断源来自P3中断(P3_INT). <b>INT_SRC[0]:</b> 0 = IRQ5中断源不是来自P2中断(P2_INT). 1 = IRQ5中断源来自P2中断(P2_INT). <b>注1:</b> 同一时刻，IRQ5 中断可以来自多个中断源. <b>注2:</b> 当中断标志被清除时，该位将被自动清除

IRQ6 (PWMA) 中断源识别寄存器(IRQ6\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) 中断源识别	0XXXX_XXXX

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				INT_SRC[3:0]			

Bits	描述
[3:0]	<p><b>IRQ6中断源识别</b></p> <p><b>INT_SRC[3]:</b> 0 = IRQ6中断源不是来自PWM3(PWMA 通道 3)中断(PWM3_INT). 1 = IRQ6中断源来自PWM3(PWMA 通道 3)中断(PWM3_INT).</p> <p><b>INT_SRC[2]:</b> 0 = IRQ6中断源不是来自PWM2(PWMA 通道 2)中断(PWM2_INT). 1 = IRQ6中断源来自PWM2(PWMA 通道 2)中断(PWM2_INT).</p> <p><b>INT_SRC[1]:</b> 0 = IRQ6中断源不是来自PWM1(PWMA 通道 1)中断(PWM1_INT). 1 = IRQ6中断源来自PWM1(PWMA 通道 1)中断(PWM1_INT).</p> <p><b>INT_SRC[0]:</b> 0 = IRQ6中断源不是来自PWM0(PWMA 通道 0)中断(PWM0_INT). 1 = IRQ6中断源来自PWM0(PWMA 通道 0)中断(PWM0_INT).</p> <p><b>注1:</b> 同一时刻，IRQ6 中断可以来自多个中断源.</p> <p><b>注2:</b> 当中断标志被清除时，该位将被自动清除</p>

IRQ7 (PWMB) 中断源识别寄存器(IRQ7\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (PWMB) 中断源识别	0xXXXX_XXXX



Bits	描述	
[3:0]	INT_SRC	<p><b>IRQ7中断源识别</b></p> <p>INT_SRC[3]:</p> <p>0 = IRQ7中断源不是来自PWM7(PWMB 通道 3)中断 (PWM7_INT).      1 = IRQ7中断源来自PWM7(PWMB 通道 3)中断 (PWM7_INT).</p> <p>INT_SRC[2]:</p> <p>0 = IRQ7中断源不是来自PWM6(PWMB 通道 2)中断 (PWM6_INT).      1 = IRQ7中断源来自PWM6(PWMB 通道 2)中断 (PWM6_INT).</p> <p>INT_SRC[1]:</p> <p>0 = IRQ7中断源不是来自PWM5(PWMB 通道 1)中断 (PWM5_INT).      1 = IRQ7中断源来自PWM5(PWMB 通道 1)中断 (PWM5_INT).</p> <p>INT_SRC[0]:</p> <p>0 = IRQ7中断源不是来自PWM4(PWMB 通道 0)中断 (PWM4_INT).      1 = IRQ7中断源来自PWM4(PWMB 通道 0)中断 (PWM4_INT).</p> <p><b>注1:</b> 同一时刻，IRQ7 中断可以来自多个中断源.  <b>注2:</b> 当中断标志被清除时，该位将被自动清除</p>

IRQ8 (TMR0) 中断源识别寄存器(IRQ8\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ8中断源识别</b> INT_SRC[2] = 保留.. INT_SRC[1] = 保留.. INT_SRC[0]: 0 = IRQ8中断源不是来自定时器0中断(TMR0_INT). 1 = IRQ8中断源来自定时器0中断(TMR0_INT). <b>注:</b> 当中断标志被清除时，该位将被自动清除.

IRQ9 (TMR1) 中断源识别寄存器(IRQ9\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别	0xXXXX_XXXX



Bits	描述	
[2:0]	INT_SRC	<p><b>IRQ9中断源识别</b></p> <p>INT_SRC[2] = 保留..</p> <p>INT_SRC[1] = 保留..</p> <p>INT_SRC[0]:</p> <p>0 = IRQ9中断源不是来自定时器1中断(TMR1_INT).</p> <p>1 = IRQ9中断源来自定时器1中断(TMR1_INT).</p> <p><b>注:</b> 当中断标志被清除时, 该位将被自动清除</p>

IRQ10 (TMR2) 中断源识别寄存器(IRQ10\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别	0XXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ10中断源识别</b> INT_SRC[2] = 保留.. INT_SRC[1] = 保留.. INT_SRC[0]: 0 = IRQ10中断源不是来自定时器2中断(TMR2_INT). 1 = IRQ10中断源来自定时器2中断(TMR2_INT). <b>注:</b> 当中断标志被清除时，该位将被自动清除.

IRQ11 (TMR3) 中断源识别寄存器(IRQ11\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述	
[2:0]	INT_SRC	<p><b>IRQ11中断源识别</b></p> <p>INT_SRC[2] = 保留..</p> <p>INT_SRC[1] = 保留..</p> <p>INT_SRC[0]:</p> <p>0 = IRQ11中断源不是来自定时器3中断(TMR3_INT).</p> <p>1 = IRQ11中断源来自定时器3中断(TMR3_INT).</p> <p><b>注:</b> 当中断标志被清除时, 该位将被自动清除</p>

IRQ12 (UART0) 中断源识别寄存器(IRQ12\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (UART0) 中断源识别	0XXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ12中断源识别</b> INT_SRC[2] = 保留.. INT_SRC[1] = 保留.. INT_SRC[0]: 0 = IRQ12中断源不是来自UART0中断(UART0_INT). 1 = IRQ12中断源来自UART0中断(UART0_INT). <b>注:</b> 当中断标志被清除时, 该位将被自动清除

IRQ13 (UART1) 中断源识别寄存器(IRQ13\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (UART1) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述	
[2:0]	INT_SRC	<p><b>IRQ13中断源识别</b></p> <p>INT_SRC[2] = 保留..</p> <p>INT_SRC[1] = 保留..</p> <p>INT_SRC[0]:</p> <p>0 = IRQ13中断源不是来自UART1中断(UART1_INT).</p> <p>1 = IRQ13中断源来自UART1中断(UART1_INT).</p> <p><b>注:</b> 当中断标志被清除时，该位将被自动清除.</p>

IRQ14 (SPI0) 中断源识别寄存器(IRQ14\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ14中断源识别</b> INT_SRC[2] = 保留.. INT_SRC[1] = 保留.. INT_SRC[0]: 0 = IRQ14中断源不是来自SPI0中断(SPI0_INT). 1 = IRQ14中断源来自SPI0中断(SPI0_INT). <b>注:</b> 当中断标志被清除时, 该位将被自动清除.

IRQ15 (SPI1) 中断源识别寄存器(IRQ15\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) 中断源识别	0xXXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述	
[2:0]	INT_SRC	<p><b>IRQ15中断源识别</b></p> <p>INT_SRC[2] = 保留..</p> <p>INT_SRC[1] = 保留..</p> <p>INT_SRC[0]:</p> <p>0 = IRQ15中断源不是来自SPI1中断(SPI1_INT).</p> <p>1 = IRQ15中断源来自SPI1中断(SPI1_INT).</p> <p><b>注:</b> 当中断标志被清除时，该位将被自动清除.</p>

IRQ18 (I<sup>2</sup>C0) 中断源识别寄存器(IRQ18\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I <sup>2</sup> C0) 中断源识别	0XXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ18中断源识别</b> INT_SRC[2] =保留.. INT_SRC[1] =保留.. INT_SRC[0]: 0 = IRQ18中断源不是来自I <sup>2</sup> C0中断(I2C0_INT). 1 = IRQ18中断源来自I <sup>2</sup> C0中断(I2C0_INT). <b>注:</b> 当中断标志被清除时，该位将被自动清除.

IRQ19 (I<sup>2</sup>C1) 中断源识别寄存器(IRQ19\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I <sup>2</sup> C1) 中断源识别 (M05xxDN/DE Only)	0xXXXX_XXXX



Bits	描述	
[2:0]	INT_SRC	<p>IRQ19中断源识别(只有M05xxDN/DE支持)</p> <p>INT_SRC[2] = 保留..</p> <p>INT_SRC[1] = 保留..</p> <p>INT_SRC[0]:</p> <p>0 = IRQ19中断源不是来自I<sup>2</sup>C1中断(I2C1_INT).</p> <p>1 = IRQ19中断源来自I<sup>2</sup>C1中断(I2C1_INT).</p> <p>注: 当中断标志被清除时, 该位将被自动清除.</p>

IRQ25 (ACMP01) 中断源识别寄存器(IRQ25\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (ACMP01) 中断源识别	0XXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ25中断源识别</b> INT_SRC[2] =保留.. INT_SRC[1] =保留.. INT_SRC[0]: 0 = IRQ25中断源不是来自ACMP01中断 (ACMP01_INT). 1 = IRQ25中断源来自ACMP01中断(ACMP01_INT). <b>注:</b> 当中断标志被清除时，该位将被自动清除.

IRQ26 (ACMP23) 中断源识别寄存器(IRQ26\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (ACMP23) 中断源识别(M05xxDN/DE Only)	0xXXXX_XXXX



Bits	描述	
[2:0]	INT_SRC	<p>IRQ26中断源识别(只有M05xxDN/DE支持)</p> <p>INT_SRC[2] =保留..</p> <p>INT_SRC[1] =保留..</p> <p>INT_SRC[0]:</p> <p>0 = IRQ26中断源不是来自ACMP23中断 (ACMP23_INT).</p> <p>1 = IRQ26中断源来自ACMP23中断(ACMP23_INT).</p> <p><b>注:</b> 当中断标志被清除时, 该位将被自动清除</p>

IRQ28 (PWRWU) 中断源识别寄存器(IRQ28\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源识别	0XXXX_XXXX

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					INT_SRC[2:0]		

Bits	描述
[2:0]	<b>INT_SRC</b> <b>IRQ28中断源识别</b> INT_SRC[2] =保留.. INT_SRC[1] =保留.. INT_SRC[0]: 0 = IRQ28中断源不是来自睡眠唤醒中断(PWRWU_INT). 1 = IRQ28中断源来自睡眠唤醒中断(PWRWU_INT). <b>注:</b> 当中断标志被清除时，该位将被自动清除

IRQ29 (ADC) 中断源识别寄存器(IRQ29\_SRC)

寄存器	偏移量	R/W	描述	复位后的值
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (ADC) 中断源识别	0xXXXX_XXXX



Bits	描述	
[2:0]	INT_SRC	<p><b>IRQ29中断源识别</b></p> <p>INT_SRC[2] = 保留..</p> <p>INT_SRC[1] = 保留.</p> <p>INT_SRC[0]:</p> <p>0 = IRQ29中断源不是来自ADC中断 (ADC_INT).</p> <p>1 = IRQ29中断源来自ADC中断(ADC_INT).</p> <p><b>注:</b> 当中断标志被清除时, 该位将被自动清除</p>

NMI中断源选择控制寄存器(NMI\_SEL)

寄存器	偏移量	R/W	描述	复位后的值
<b>NMI_SEL</b>	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
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			<b>NMI_SEL[4:0]</b>				

Bits	描述	
[31:9]	保留	保留
[8]	<b>NMI_EN</b>	<b>NMI 中断使能 (写保护)</b> 1 = 使能 NMI 中断 0 = 禁止 NMI 中断 注：这个比特是写保护的，意味着在写这个比特之前需要连续写“59h”，“16h”，“88h”到地址 0x5000_0100 来解锁。参考寄存器 REGWRPROT，地址 GCR_BA+0x100
[7:5]	保留	保留
[4:0]	<b>NMI_SEL</b>	<b>NMI中断源选择</b> Cortex®-M0的NMI 中断源可以从32个外设中断中选择一个 NMI_SEL bit[4:0] 用于选择NMI 中断源

### 6.2.10 系统控制块(SCB)

Cortex®-M0的状态和操作模式由系统控制寄存器控制，包括CPUID，Cortex®-M0中断优先级和Cortex®-M0电源管理都可以通过这些系统控制寄存器控制。

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

#### 6.2.10.1 系统控制块寄存器映射

**R:** 只读, **W:** 只写, **R/W:** 可读写, **W&C:** 写1清除为零

寄存器	偏移量	R/W	描述	复位后的值
<b>SCB基地址</b>				
<b>SCS_BA = 0xE000_ED00</b>				
<b>CPUID</b>	SCS_BA + 0x000	R	CPUID 寄存器	0x0000_0000
<b>ICSR</b>	SCS_BA + 0x004	R/W	中断控制状态寄存器	0x0000_0000
<b>AIRCR</b>	SCS_BA + 0x00C	R/W	中断应用和复位控制寄存器	0xFA05_0000
<b>SCR</b>	SCS_BA + 0x010	R/W	系统控制寄存器	0x0000_0000
<b>SHPR2</b>	SCS_BA + 0x01C	R/W	系统处理函数优先级寄存器2	0x0000_0000
<b>SHPR3</b>	SCS_BA + 0x020	R/W	系统处理函数优先级寄存器3	0x0000_0000

CPUID Base寄存器(CPUID)

寄存器	偏移量	R/W	描述	复位后的值
CPUID	SCS_BA + 0x000	R	CPUID寄存器	0x 410CC200

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

Bits	描述	
[31:24]	<b>IMPLEMENTER</b>	执行码 由ARM分配执行码. ( ARM = 0x41)
[23:20]	保留	保留
[19:16]	<b>PART</b>	处理器架构码 ARMv6-M 读取值为0xC
[15:4]	<b>PARTNO</b>	处理器型号 读取值为 0xC20.
[3:0]	<b>REVISION</b>	版本号 读取值为 0x0



### 中断控制状态寄存器(ICSR)

寄存器	偏移量	R/W	描述				复位后的值
ICSR	SCS_BA + 0x004	R/W	中断控制状态寄存器				0x 00000000

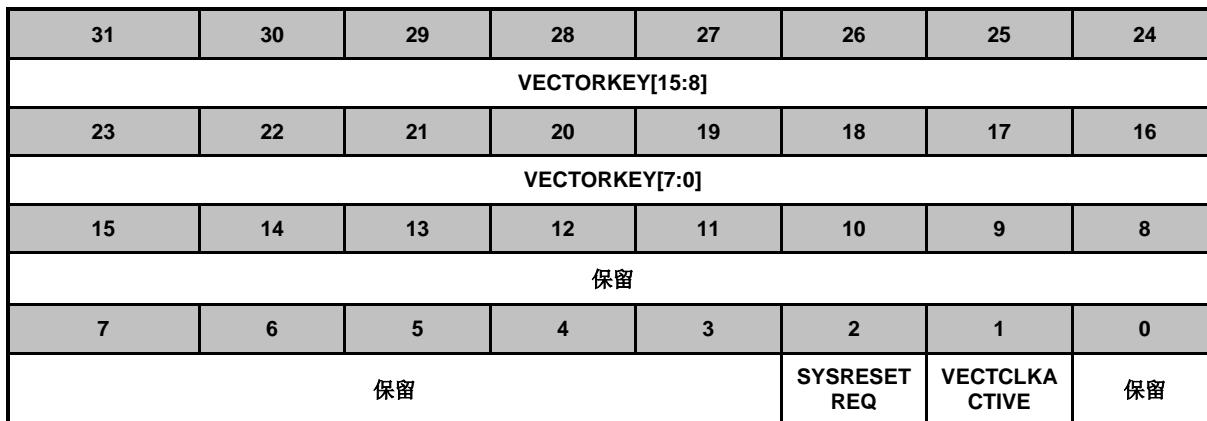
31	30	29	28	27	26	25	24
NMIPENDSET	保留		PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	保留
23	22	21	20	19	18	17	16
ISRPREEMPT	ISRPENDING	保留			VECTPENDING[5:4]		
15	14	13	12	11	10	9	8
VECTPENDING[3:0]				保留			
7	6	5	4	3	2	1	0
保留		VECTACTIVE[7:0]					

Bits	R/W	描述	
[31]	R/W	NMIPENDSET	<p>触发NMI中断 写: 0 = 无作用 1 = 改变NMI异常状态为未处理. 读: 0 = 没有NMI 异常等待处理 1 = 有NMI异常等待处理</p> <p>注: 设置该位将激活NMI, 由于NMI是最高优先级的异常, 正常情况下, 只要处理器检测到写'1'到这个位, 将马上进入中断处理函数, 然后这个位就会被自动清成'0'. 这就意味着如果在中断处理函数中读这个位返回'1', 就是NMI中断再次被置。</p>
[30:29]		保留	保留
[28]	R/W	PENDSVSET	<p>触发PendSV中断 写: 0 = 无作用 1 = 改变PendSV异常状态为未处理. 读: 0 = 没有PendSV异常等待处理 1 = 有PendSV异常等待处理</p> <p>注: 触发PendSV 中断。写"1"到这个位是唯一触发PendSV的方式。</p>

[27]	W	PENDSVCLR	<p><b>清除PendSV中断</b></p> <p>写:</p> <p>0 = 无作用 1 = 清除PendSV 中断.</p> <p>这个位是只写的, 当要清除PendSV中断的时候, 必须同时对PENDSVSET写"0", PENDSVCLR写"1"</p>
[26]	R/W	PENDSTSET	<p><b>触发SysTick中断</b></p> <p>写:</p> <p>1 = 触发SysTick中断. 0 = 无作用</p> <p>读:</p> <p>1 = 有SysTick中断等待处理 0 = 没有SysTick中断等待处理。</p>
[25]	W	PENDSTCLR	<p><b>清除SysTick中断</b></p> <p>写:</p> <p>1 = 清除等待处理的SysTick.中断 0 = 无作用</p>
[23]	R	ISRPREEMPT	<p><b>中断抢占位</b></p> <p>如果置位, 当从调试状态退出时, 未处理的异常将被处理.</p>
[22]	R	ISR PENDING	<p><b>有中断等待处理标志位, 不包括NMI和HardFault中断(只读)</b></p> <p>0 = 没有中断等待处理 1 = 有中断等待处理</p> <p>这个位是只读的.</p>
[21:18]		保留	保留
[17:12]	R	VECTPENDING	<p><b>等待处理的最高优先级异常号</b></p> <p>0 = 没有中断等待处理 非0 = 等待处理的最高优先级异常号</p>
[8:0]	R	VECTACTIVE	<p><b>包含当前正在处理的异常号</b></p> <p>0 = 线程模式 非0: 当前正在处理的异常号</p>

应用中断和复位控制寄存器(AIRCR)

寄存器	偏移量	R/W	描述	复位后的值
AIRCR	SCS_BA+0x00C	R/W	应用中断和复位控制寄存器	0xFA05_0000



Bits	描述		
[31:16]	<b>VECTORKEY</b>	寄存器访问值 写该寄存器时，该域应该是0x05FA，否则写动作将被忽略。VECTORKEY域用来防止复位或者异常状态清除时意外写该寄存器	
[15:3]	保留	保留	
[2]	<b>SYSRESETREQ</b>	系统复位请求 该位写1，产生复位信号给芯片表示有复位请求。 该位是只写的，在复位时自动清零。	
[1]	<b>VECTCLRACTIVE</b>	异常活动状态清除位 预留用于调试。写该寄存器时，用户必须将该位写0，否则行为无法预期	
[0]	保留	保留	



### 系统控制寄存器(SCR)

寄存器	偏移量	R/W	描述	复位后的值
SCR	SCS_BA 0x010	+R/W	系统控制寄存器	0x 00000000

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			SEVONPEND	保留	SLEEPDEEP	SLEEPONEXIT	保留

Bits	描述	
[4]	<b>SEVONPEND</b>	未处理位发送事件  0 = 只有使能的中断或者事件可以唤醒处理器，禁止的中断不能唤醒 1 = 使能的事件和所有中断，包括禁止的中断，可以唤醒处理器  当事件或者中断进入未处理状态时，将把CPU从WFE指令唤醒。如果处理器没有正在等待一个事件，这个事件将被注册，在下一个WFE指令时起作用。 执行SEV指令或者外部事件也可以唤醒处理器
[3]	预留	预留
[2]	<b>SLEEPDEEP</b>	控制处理器进入睡眠还是深度睡眠模式  0 = 睡眠 1 = 深度睡眠。
[1]	<b>SLEEPONEXIT</b>	退出时睡眠使能控制  当从Handler模式返回Thread模式时，是否进入睡眠模式  0: 返回Thread模式时，不要进入睡眠 1: 当从中断处理函数返回Thread模式时进入睡眠或者深度睡眠模式  设置这个比特为"1"，可以使能一个中断驱动的应用，避免从中断返回时进入一个空的主函数
[0]	预留	预留



### 系统处理函数优先级寄存器2 (SHPR2)

寄存器	偏移量	R/W	描述	复位后的值
SHPR2	SCS_BA +0x01C	R/W	系统处理函数优先级寄存器2	0x 00000000

31	30	29	28	27	26	25	24
<b>PRI_11</b>		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							

Bits	描述	
[31:30]	<b>PRI_11</b>	系统处理函数11的优先级 – SVCall “0”表示最高优先级& “3”表示最低优先级



### 系统处理函数优先级寄存器3 (SHPR3)

寄存器	偏移量	R/W	描述	复位后的值
SHPR3	SCS_BA 0x020	+R/W	系统处理函数优先级寄存器3	0x 00000000

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

Bits	描述	
[31:30]	<b>PRI_15</b>	系统处理函数15的优先级- SysTick “0” 表示最高优先级 & “3” 表示最低优先级
[23:22]	<b>PRI_14</b>	系统处理函数14的优先级 – PendSV “0” 表示最高优先级 & “3” 表示最低优先级

## 6.3 模拟比较器 (ACMP)

### 6.3.1 概述

NuMicro M05xxBN/DN/DE最多有4个比较器，可以在不同的配置下使用。当正端输入大于负端输入时，比较器输出逻辑“1”，否则输出“0”。当比较器输出值改变，每个比较器可以配置发生中断。

### 6.3.2 特性

- 模拟输入电压范围: 0~AV<sub>DD</sub>
- 支持迟滞功能
- 每个模拟比较器负端可以选择输入内部参考电压
- 4/2个比较器共享2/1个中断向量

	M05xxBN	M05xxDN/DE
ACMP数量	2	4
ACMP 输出反转功能	-	●

表 6-5 M05xxBN 和 M05xxDN/DE 比较器功能比较 (ACMP)

## 6.3.3 框图

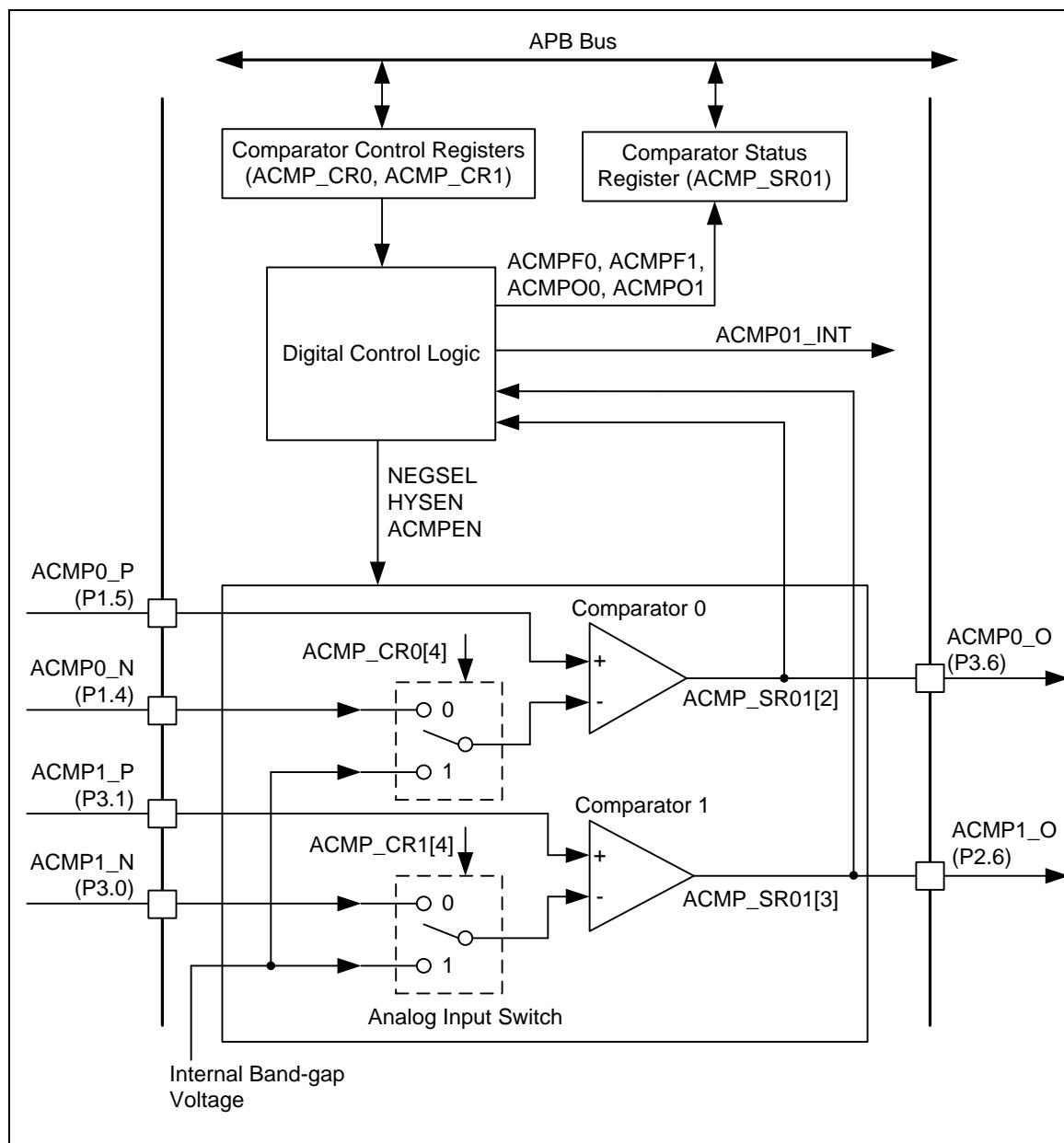


图 6-3 模拟比较器 0/1 框图

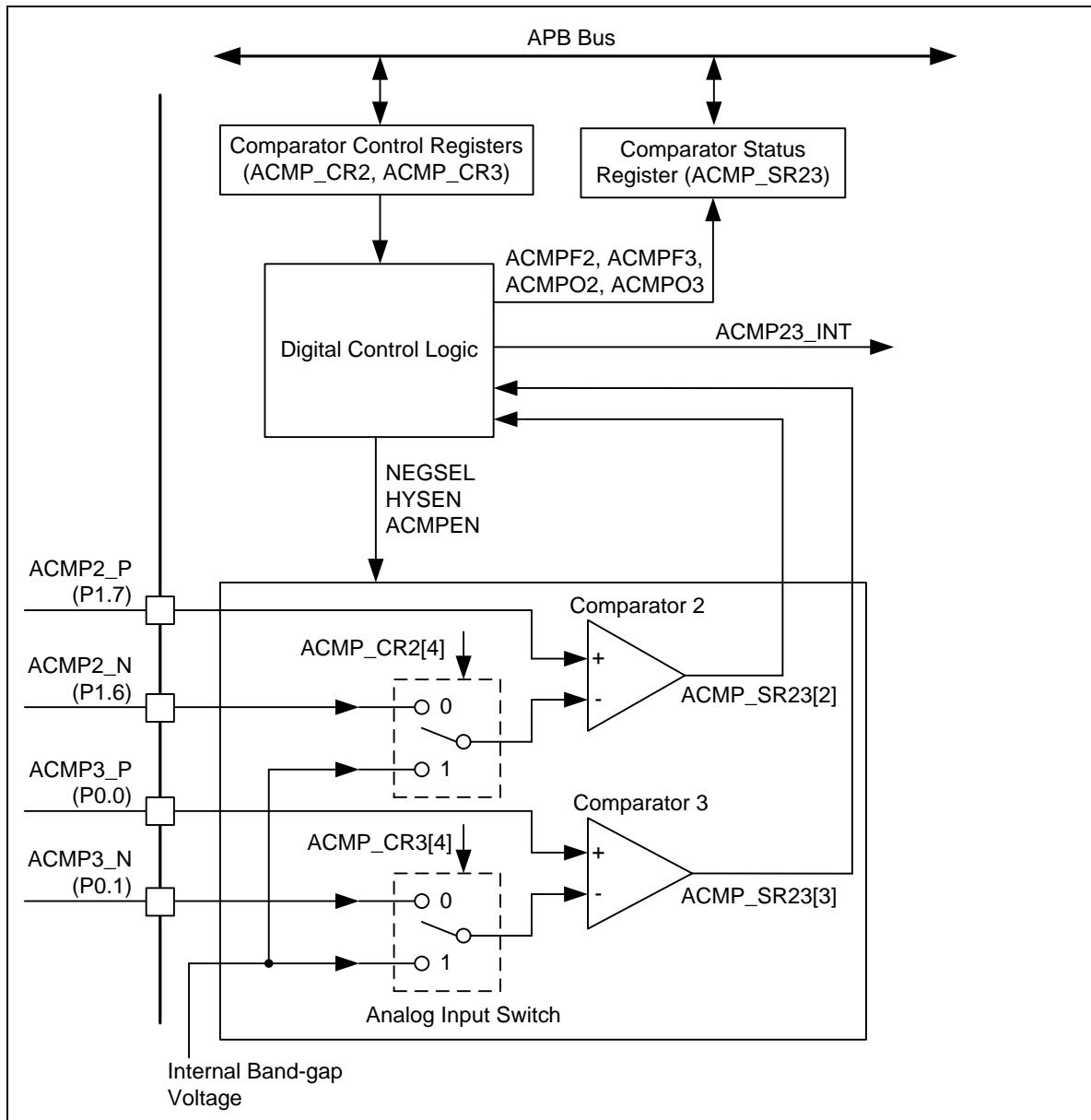


图 6-4 模拟比较器 2/3 框图 (M05xxDN/DE Only)

### 6.3.4 基本配置

ACMP引脚功能使用寄存器P0\_MFP, P1\_MFP, P2\_MFP 和 P3\_MFP来配置。推荐关闭模拟输入引脚的数字通路，以避免漏电。数字通路可以使用P0\_OFFD, P1\_OFFD 和 P3\_OFFD寄存器来关闭。如果某个GPIO脚被配置为ACMP输入引脚，Px\_PMD寄存器中该引脚不应该被设为推挽输出模式，输入模式是最安全的配置。如果选择使用开漏输出模式或者准双向模式，不要输出0。缺省GPIO输入值是1。复位之后，M05xxBN中默认Px\_PMD设定是准双向的，M05xxDN/DE中默认Px\_PMD设定由用户配置区决定

ACMP01 和 ACMP23外设时钟使能位是APBCLK[30] 和 APBCLK[31]

### 6.3.5 功能描述

#### 6.3.5.1 中断源

比较器的输出由 PCLK 采样，反应在 ACMP\_SR01/23 寄存器的 ACMPO<sub>x</sub> 位。如果寄存器 ACMP\_CR<sub>x</sub> 中的 ACMPIE 设为 1，当比较器输出的值改变将导致比较器标志位 ACMPF<sub>x</sub> 被设，比较器中断将发生。软件可以写 1 到 ACMPF<sub>x</sub> 来使该位清 0.

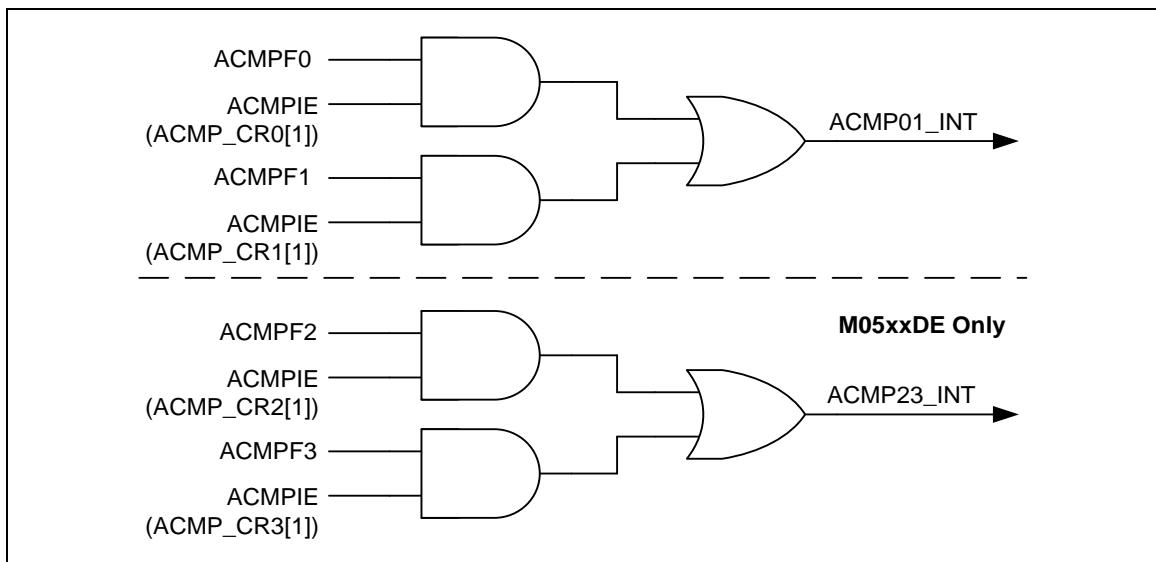


图 6-5 比较器中断源

#### 6.3.5.2 迟滞功能

模拟比较器提供迟滞功能使比较输出转变更稳定。如果当前比较器输出为 0，正端输入电压超过负端输入电压大于迟滞电压，比较器输出才会变成 1。相似的，如果当前比较器输出为 1，负端输入电压低于正端输入电压大于迟滞电压，比较器输出才会变成 0。

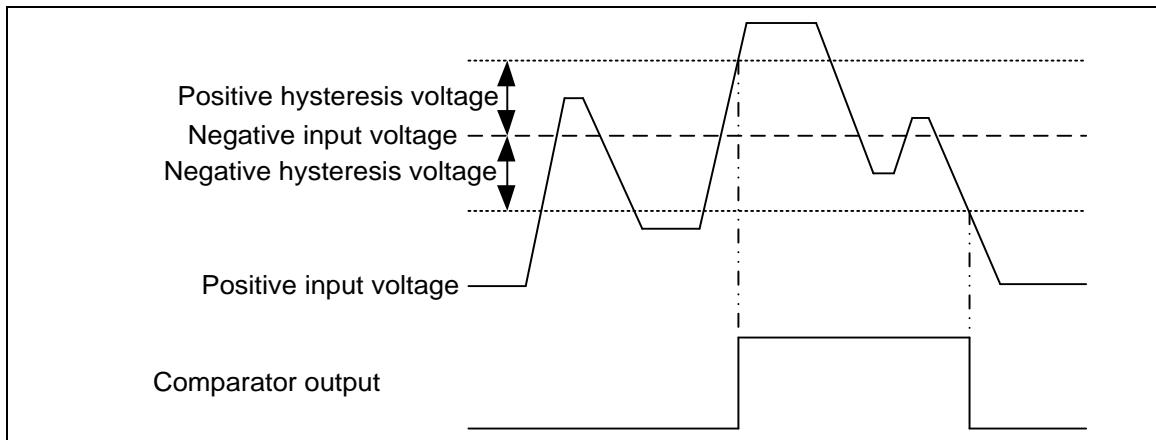


图 6-6 比较器迟滞功能

### 6.3.6 寄存器映射

R: 只读, W: 只写, R/W: 读/写

寄存器	偏移	R/W	描述	复位后的值
<b>ACMP 基地址:</b>				
ACMP01_BA = 0x400D_0000				
ACMP23_BA = 0x401D_0000 (只有M05xxDN/DE)				
ACMP_CR0	ACMP01_BA+0x00	R/W	比较器0 控制寄存器	0x0000_0000
ACMP_CR1	ACMP01_BA+0x04	R/W	比较器1 控制寄存器	0x0000_0000
ACMP_SR01	ACMP01_BA+0x08	R/W	比较器0/1状态寄存器	0x0000_0000
ACMP_CR2	ACMP23_BA+0x00	R/W	比较器2控制寄存器(只有M05xxDN/DE)	0x0000_0000
ACMP_CR3	ACMP23_BA+0x04	R/W	比较器3 控制寄存器(只有M05xxDN/DE)	0x0000_0000
ACMP_SR23	ACMP23_BA+0x08	R/W	比较器2/3状态寄存器(只有M05xxDN/DE)	0x0000_0000

### 6.3.7 寄存器描述

#### 模拟比较器0控制寄存器 (ACMP\_CRO)

寄存器	偏移	R/W	描述	复位后的值
ACMP_CRO	ACMP01_BA+0x00	R/W	比较器0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	ACMPOINV	保留	NEGSEL	保留	HYSEN	ACMPIE	ACMPEN

Bits	描述	
[31:7]	保留	保留
[6]	ACMPOINV	比较器0输出反转使能控制 (只有M05xxDN/DE) 0 = 禁止比较器输出反转。 1 = 使能比较器输出反转。
[5]	保留	保留
[4]	NEGSEL	比较器0负端输入选择 1 = 负端选择输入内部band-gap电压 0 = 负端选择从ACMP0_N引脚输入
[3]	保留	保留
[2]	HYSEN	比较器0迟滞使能控制 1 = 使能迟滞功能。 0 = 关闭迟滞功能。
[1]	ACMPIE	比较器0中断使能控制 1 = 使能中断 0 = 禁止中断功能 如果ACMPIE 等于 1, 比较器转换结果发生反转时, 中断将发生
[0]	ACMPEN	比较器0使能控制 1 = 使能 0 = 禁止 注: ACMPEN 置位以后, 比较器输出需要等2 us 的稳定时间

**比较器1 控制寄存器(ACMP CR1)**

寄存器	偏移	R/W	描述	复位后的值
ACMP_CR1	ACMP01_BA +0x04	R/W	比较器1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	ACMPOINV	保留	NEGSEL	保留	HYSEN	ACMPIE	ACMPEN

Bits	描述	
[31:7]	保留	保留
[6]	ACMPOINV	比较器1 输出反转使能控制 (只有M05xxDN/DE) 0 = 禁止比较器输出反转 1 = 使能比较器输出反转.
[5]	保留	保留
[4]	NEGSEL	比较器1负端输入选择 1 = 负端选择输入内部band-gap电压 0 = 负端选择从ACMP1_N引脚输入
[3]	保留	保留
[2]	HYSEN	使能比较器1迟滞 功能 1 = 使能迟滞功能. 0 = 关闭迟滞功能
[1]	ACMPIE	使能比较器1的中断 1 = 使能中断 0 = 禁止中断功能 如果ACMPIE 等于 1, 比较器转换结果发生反转时, 中断将发生.
[0]	ACMPEN	使能比较器1 1 = 使能 0 = 禁止 注: ACMPEN 置位以后, 比较器输出需要等2 us 的稳定时间

**比较器0/1状态寄存器 (ACMP\_SR01)**

寄存器	偏移	R/W	描述	复位后的值
ACMP_SR01	ACMP01_BA +0x08	R/W	比较器0/1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				ACMPO1	ACMPO0	ACMPF1	ACMPF0

Bits	描述	
[31:4]	保留	保留
[3]	ACMPO1	<b>比较器1的输出</b> 这个bit显示当前比较器的比较结果，用户可以通过读这个bit查看比较器的输出。当比较器被关闭时(ACMP_CR1[0]= 0)此bit将被清0。 0 = 比较器1输出0 1 = 比较器1输出1
[2]	ACMPO0	<b>比较器0的输出</b> 这个bit显示当前比较器的比较结果，用户可以通过读这个bit查看比较器的输出。当比较器被关闭时(ACMP_CR0[0]= 0)此bit将被清0。 0 = 比较器0输出0 1 = 比较器0输出1
[1]	ACMPF1	<b>比较器1标志位</b> 当比较器1输出状态改变时，这个bit将被置位。如果ACMP_CR1[1]使能，中断将发生。 0 = 比较器1输出没有发生改变 1 = 从上次该位清0之后，比较器1的输出发生了改变 注：这个bit写1清0。
[0]	ACMPF0	<b>比较器0标志位</b> 当比较器0输出状态改变时，这个bit将被置位。如果ACMP_CR0[1]使能，中断将发生。 0 = 比较器0输出没有发生改变 1 = 从上次该位清0之后，比较器0的输出发生了改变 注：这个bit写1清0。

模拟比较器2控制寄存器 (ACMP\_CR2)

寄存器	偏移	R/W	描述	复位后的值
ACMP_CR2	ACMP23_BA+0x00	R/W	比较器2控制寄存器(只有M05xxDN/DE)	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	ACMPOINV	保留	NEGSEL	保留	HYSEN	ACMPIE	ACMPEN

Bits	描述	
[31:7]	保留	保留
[6]	ACMPOINV	比较器 2输出反转使能控制 0 = 禁止比较器输出反转. 1 = 使能比较器输出反转.
[5]	保留	保留
[4]	NEGSEL	比较器2负端输入选择 1 = 负端选择输入内部band-gap电压 0 = 负端选择从ACMP2_N引脚输入
[3]	保留	保留
[2]	HYSEN	比较器2迟滞 使能控制 1 = 使能迟滞功能. 0 = 关闭迟滞功能.
[1]	ACMPIE	比较器2中断使能控制 1 = 使能中断 0 = 禁止中断功能 如果ACMPIE 等于 1, 比较器转换结果发生反转时, 中断将发生
[0]	ACMPEN	比较器2使能控制 1 = 使能 0 = 禁止 注: ACMPEN 置位以后, 比较器输出需要等2 us 的稳定时间

比较器3 控制寄存器(ACMP CR3)

寄存器	偏移	R/W	描述	复位后的值
ACMP_CR3	ACMP23_BA +0x04	R/W	比较器3控制寄存器(只有M05xxDN/DE)	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	ACMPOINV	保留	NEGSEL	保留	HYSEN	ACMPIE	ACMPEN

Bits	描述	
[31:7]	保留	保留
[6]	ACMPOINV	比较器3输出反转使能控制 0 = 禁止比较器输出反转 1 = 使能比较器输出反转.
[5]	保留	保留
[4]	NEGSEL	比较器3负端输入选择 1 = 负端选择输入内部band-gap电压 0 = 负端选择从ACMP3_N引脚输入
[3]	保留	保留
[2]	HYSEN	使能比较器3迟滞 功能 1 = 使能迟滞功能. 0 = 关闭迟滞功能
[1]	ACMPIE	使能比较器3的中断 1 = 使能中断 0 = 禁止中断功能 如果ACMPIE 等于 1, 比较器转换结果发生反转时, 中断将发生.
[0]	ACMPEN	使能比较器3 1 = 使能 0 = 禁止 注: ACMPEN 置位以后, 比较器输出需要等2 us 的稳定时间

比较器2/3状态寄存器 (ACMP\_SR23)

寄存器	偏移	R/W	描述	复位后的值
ACMP_SR23	ACMP23_BA +0x08	R/W	比较器2/3状态寄存器(只有M05xxDN/DE)	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				ACMPO3	ACMPO2	ACMPF3	ACMPF2

Bits	描述	
[31:4]	保留	保留
[3]	ACMPO3	<b>比较器3的输出</b> 这个bit显示当前比较器的比较结果，用户可以通过读这个bit查看比较器的输出。当比较器被关闭时(ACMP_CR3[0]= 0)此bit将被清0。 0 = 比较器3输出0 1 = 比较器3输出1
[2]	ACMPO2	<b>比较器2的输出</b> 这个bit显示当前比较器的比较结果，用户可以通过读这个bit查看比较器的输出。当比较器被关闭时(ACMP_CR2[0]= 0)此bit将被清0。 0 = 比较器2输出0 1 = 比较器2输出1
[1]	ACMPF3	<b>比较器3标志位</b> 当比较器3输出状态改变时，这个bit将被置位。如果ACMP_CR3[1]使能，中断将发生。 0 = 比较器3输出没有发生改变 1 = 从上次该位清0之后，比较器3的输出发生了改变 <b>注：</b> 该位写1清0。
[0]	ACMPF2	<b>比较器2标志位</b> 当比较器2输出状态改变时，这个bit将被置位。如果ACMP_CR3[1]使能，中断将发生。 0 = 比较器2输出没有发生改变 1 = 从上次该位清0之后，比较器3的输出发生了改变 <b>注：</b> 该位写1清0。

## 6.4 模拟数字转换(ADC)

### 6.4.1 概述

NuMicro™ M05xxBN/DN/DE系列包含一个8通道12位的SAR型模拟 – 数字转换器 (SAR A/D转换器). A/D 转换器支持四种工作模式: 单次转换模式、突发转换模式、单周期扫描模式和连续扫描模式. A/D 转换可以通过软件，外部STADC/P3.2引脚启动或者PWM触发(只有M05xxDN/DE)。

### 6.4.2 特性

- 模拟输入电压范围:  $0 \sim AV_{DD}$ .
- 12位分辨率和10位精度保证.
- 最多 8 路单端模拟输入通道或4路差分模拟输入通道.
- 最大 ADC 时钟频率 16MHz.
- 高达760k SPS 采样速率.
- 四种操作模式
  - ◆ 单次转换模式: A/D在指定通道完成一次转换.
  - ◆ 突发模式: A/D转换在指定单个通道连续进行，并将结果顺序地存入FIFO.
  - ◆ 单周期扫描模式: A/D 转换在所有指定通道完成一次转换（从低序号通道到高序号通道）.
  - ◆ 连续扫描模式: A/D 转换连续执行单周期扫描模式直到软件停止A/D转换.
- A/D转换开始条件
  - ◆ 软件向ADST 位写1
  - ◆ 外部引脚STADC触发
  - ◆ PWM触发, 启动延迟可以配置(只有M05xxDN/DE)
- 每个通道的转换结果存储在相应数据寄存器内，并带有有效和溢出标志.
- 转换结果可以和指定的值相比较，当转换结果和比较寄存器的设定值相匹配时，用户可设定是否产生中断请求.
- 通道 7 支持 3种输入源: 外部模拟电压, 内部带隙电压和内部温度传感器的输出.

## 6.4.3 ADC框图

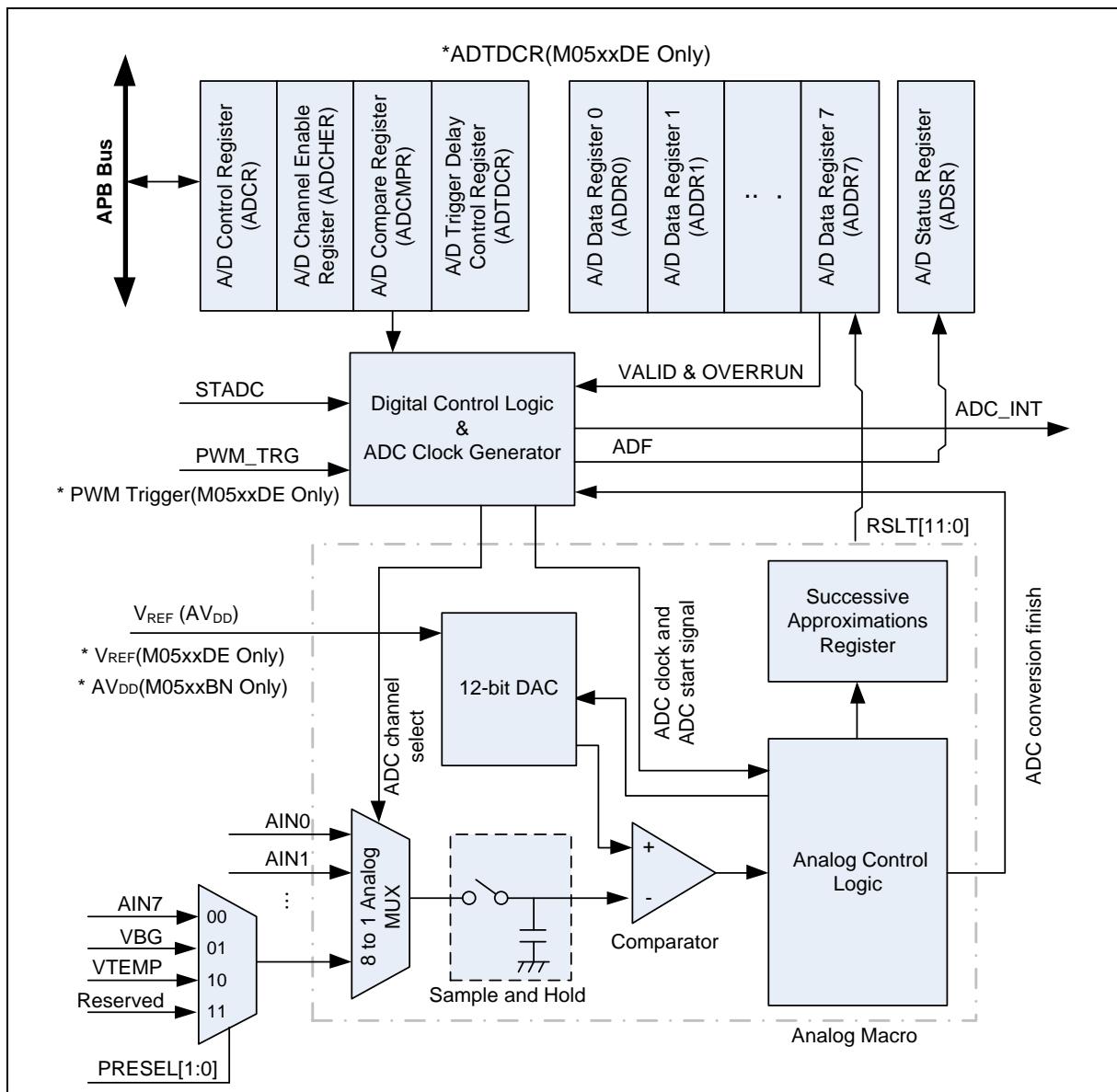


图 6-7 ADC 控制器框图

#### 6.4.4 基本配置

ADC引脚通过P1\_MFP寄存器配置。推荐关闭模拟输入引脚的数字通路，以避免漏电。用户可以通过配置P1\_OFFD寄存器关闭数字通路。

ADC外设时钟通过APBCLK[28]使能，ADC时钟源通过CLKSEL1[3:2]设定。时钟预分频由CLKDIV[23:16]决定。

#### 6.4.5 ADC功能描述

A/D转换器通过逐次逼近的方式运行，分辨率为12位。ADC共有4种操作模式：单次转换模式、突发转换模式、单周期扫描模式和连续扫描模式。当改变工作模式或使能的模拟输入通道时，为了防止错误的操作，改变之前软件需清ADST位为0(ADCR寄存器)。

##### 6.4.5.1 ADC时钟发生器

最大采样率达760K。ADC有4个时钟源可选，可由ADC\_S(CLKSEL1[3:2])选择，ADC时钟频率由一个8位分频器按如下公式进行8位预分频：

ADC时钟频率 = (ADC时钟源频率) / (ADC\_N+1)；  
8位ADC\_N在寄存器CLKDIV[23:16]中。

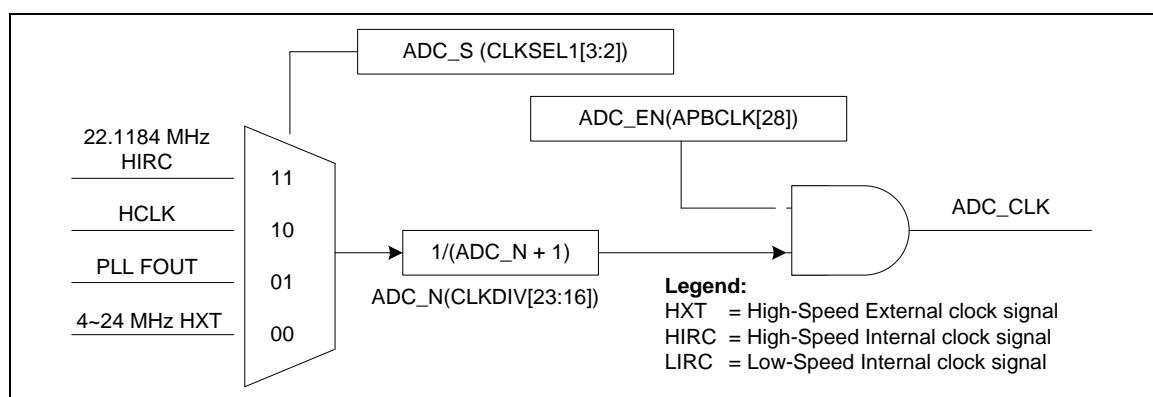


图 6-8 ADC 时钟控制

#### 6.4.5.2 单次转换模式

在单次转换模式下, A/D 转换只在指定的通道上执行一次, 操作流程如下:

1. 当通过软件或外部触发输入使 ADCR 的 ADST =1 时, A/D 转换开始.
2. 当 A/D 转换完成, A/D 转换的结果将存储于相应通道的 A/D 数据寄存器中
3. A/D 转换完成, ADSR 的 ADF 位置 1. 若此时 ADCR 寄存器的 ADIE 位置 1, 将产生 ADC 中断.
4. A/D 转换期间, ADST 位保持为 1。A/D 转换结束, ADST 位自动清 0 , A/D 转换器进入空闲状态。

**注1:** 如果软件使能超过1个通道, 只有最小编号的通道被选中, 其他通道被忽略

**注2:** 在硬件清除ADST bit之后, M05xxBN中ADST bit必须维持在0至少1个ADC时钟周期, 才能再次将ADST 置为1, 否则A/D转换器可能不工作。M05xxDN/DE没有这个限制

**注3:** 当ADC正在转换时, 如果ADST被清成0, BUSY位将被立即清成0, M05xxBN中ADC完成当前转换, 将结果保存到 ADDR<sub>x</sub>寄存器。但是M05xxDN/DE中ADC不完成当前转换, A/D转换直接进入空闲状态

通过ADC驱动, 单次转换模式设定流程如下

```
/* 设定ADC 操作模式为单次模式, 输入模式为单端模式, 并使能ADC 转换器 */
ADC->ADCR = (ADC_ADCR_ADMD_SINGLE
                | ADC_ADCR_DIFFEN_SINGLE_END
                | ADC_ADCR_ADEN_CONVERTER_ENABLE);

/* 使能模拟输入通道 2 */
_ADC_SET_CHANNEL(1<<2);

/* 安全起见清除A/D 中断标志 */
ADC->ADSR = ADC_ADSR_ADF_Msk;

/* 使能ADC 中断 */
_ADC_ENABLE_ADC_INT();
NVIC_EnableIRQ(ADC_IRQn);

/* 启动 A/D 转换 */
_ADC_START_CONVERT();
```

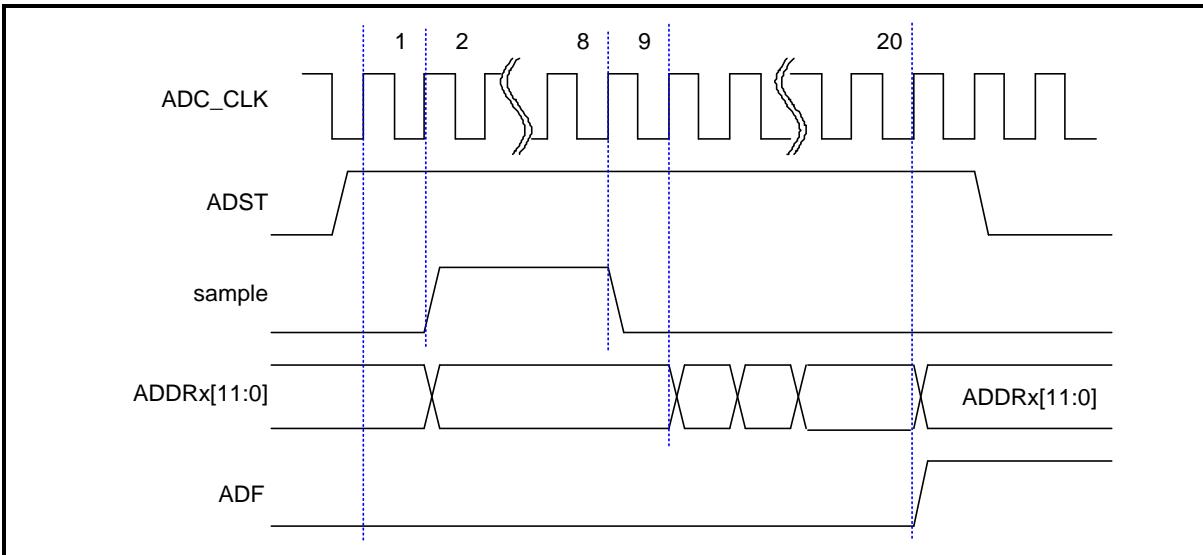


图 6-9 单次转换模式时序图

#### 6.4.5.3 突发模式

在突发模式下，A/D 转换采样并转换指定的单个通道，将转换结果按序存放到FIFO中（最多8次采样）。操作步骤如下：

1. 软件或外部触发置ADCR的ADST位为1，在序号最小的通道上A/D 转换开始。
2. 当指定通道的A/D转换完成后，结果按序存入FIFO，可以从A/D数据寄存器0读到。
3. A/D转换完成将结果存放到FIFO中时，如果发现FIFO中存储了大于等于4个采样值，ADSR的ADF位置1。如果此时ADIE位等于1，就会发生ADC中断。
4. 只要ADST位保持为1，步骤2到步骤3会一直重复。当ADST位清零时，M05xxBN中ADC控制器将完成当前转换并进入空闲状态。但是M05xxDN/DE中ADC不完成当前转换，而是立即停止，A/D转换器直接进入空闲状态。

**注1：**在突发模式下，如果软件使能多个通道，则序号最小的通道将被转换，其他通道忽略。

**注2：** M05xxBN中突发模式下，ADC不支持2的补码输出格式，DMOF位必须清成0；M05xxDN/DE中没有这个限制，突发模式下支持输出2的补码。

通过ADC驱动，突发模式设定流程如下

```
/* 设定 ADC 操作模式为突发模式，输入模式为单端模式并使能ADC转换器*/
ADCR = (ADC_ADCR_ADMD_BURST
         | ADC_ADCR_DIFFEN_SINGLE_END
         | ADC_ADCR_ADEN_CONVERTER_ENABLE);

/* 使能通道2 */
_ADC_SET_CHANNEL(1<<2);

/* 安全起见清除 A/D 中断 */
ADSR = ADC_ADSR_ADF_Msk;
```

```

/* 使能 ADC 中断 */
    _ADC_ENABLE_ADC_INT();
    NVIC_EnableIRQ(ADC_IRQn);

/* 启动A/D 转换 */
    _ADC_START_CONVERT();

```

#### 6.4.5.4 单周期扫描模式

在单周期扫描模式下，从被使能的最小序号通道到最大序号通道按序都将进行一次A/D转换，具体流程如下：

1. 软件或外部触发使ADCR 寄存器的 ADST 位置1，从最小序号通道开始 A/D 转换。
2. 每个使能的通道 A/D 转换完成后，A/D 转换结果将依次存放到相应的数据寄存器中。
3. 当所有使能的通道转换都完成后，ADSR的ADF位置1，如果ADC中断使能，则ADC中断发生。
4. ADC完成一轮转换后，ADST 位自动清 0，A/D 转换器进入空闲状态。如果在所有被使能通道完成转换前 ADST 清 0，M05xxBN中ADC将完成当前转换，并且序号最小的通道的结果将不可预知。M05xxDN/DE中ADC 不完成当前转换，A/D 转换器直接进入空闲状态。**注意：**M05xxBN中在硬件清除ADST bit之后，ADST bit必须维持0至少1个ADC时钟周期，才能再次将ADST置为1，否则，A/D转换器可能不工作

通过ADC驱动，单次循环模式设定流程如下

```

/* 设定 ADC 操作模式为单周期扫描模式, 输入模式为单端模式并使能ADC转换器 */
    ADC->ADCR = (ADC_ADCR_ADMD_SINGLE_CYCLE
                  | ADC_ADCR_DIFFEN_SINGLE_END
                  | ADC_ADCR_ADEN_CONVERTER_ENABLE);

/*使能通道 0, 1, 2 和 3 */
    _ADC_SET_CHANNEL(0xF);

/*安全起见清除 A/D 中断*/
    ADC->ADSR = ADC_ADSR_ADF_Msk;

/*启动A/D 转换*/
    _ADC_START_CONVERT();

/* 等待转换完成*/
    _ADC_WAIT_CONVERSION_DONE();

```

使能通道(0, 2, 3 and 7) 的单周期扫描模式时序图如下：

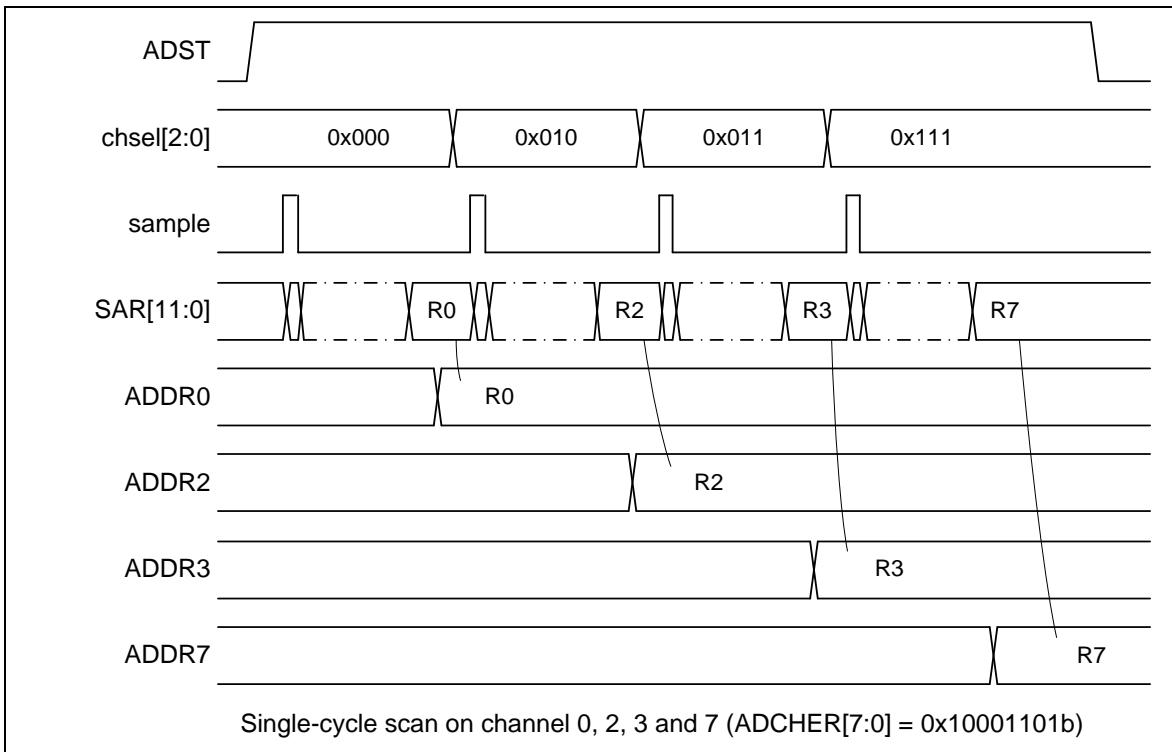


图 6-10 使能通道上单周期扫描模式时序图

#### 6.4.5.5 连续扫描模式

在连续扫描模式下，A/D转换在通过ADCHER寄存器中的CHEN位使能的通道上顺序进行(最多8个ADC通道)。操作步骤如下：

1. 通过软件或外部触发使ADCR 寄存器的 ADST 位置位，从最小序号通道到最大序号通道的 A/D 转换开始。
2. 每路 A/D 转换完成后，A/D 转换结果将存储到相应的数据寄存器中。
3. 当被使能的所有通道都完成了一次转换后，ADF 位（ADSR 寄存器）置1. 如果ADC中断使能，则ADC中断发生。如果软件没有清零ADST位，则在使能的最小通道号上的转换又一次开始。
4. 只要ADST位保持为1，步骤2到步骤3会一直重复.当ADST清0，M05xxBN 中ADC 控制器将完成当前转换，被使能的最小序号ADC通道的结果将不可预料。M05xxDN/DE中ADC不完成当前转换，ADC直接进入空闲模式

通过ADC驱动，连续循环模式设定流程如下

```
/* 设定 ADC 操作模式为单周期扫描模式, 输入模式为单端模式并使能ADC转换器 */
ADC->ADCR = (ADC_ADCR_ADMD_CONTINUOUS
              | ADC_ADCR_DIFFEN_SINGLE_END
              | ADC_ADCR_ADEN_CONVERTER_ENABLE);

/*使能通道 0, 1, 2 和 3 */
_ADC_SET_CHANNEL(0xF);
```

```

/*安全起见清除 A/D 中断*/
ADC->ADSR = ADC_ADSR_ADF_Msk;

/*启动A/D 转换*/
_ADC_START_CONVERT();

/* 等待转换完成*/
_ADC_WAIT_CONVERSION_DONE();

/* 停止 A/D 转换 */
_ADC_STOP_CONVERT();

```

使能通道(0, 2, 3和7) 连续扫描模式时序图如下:

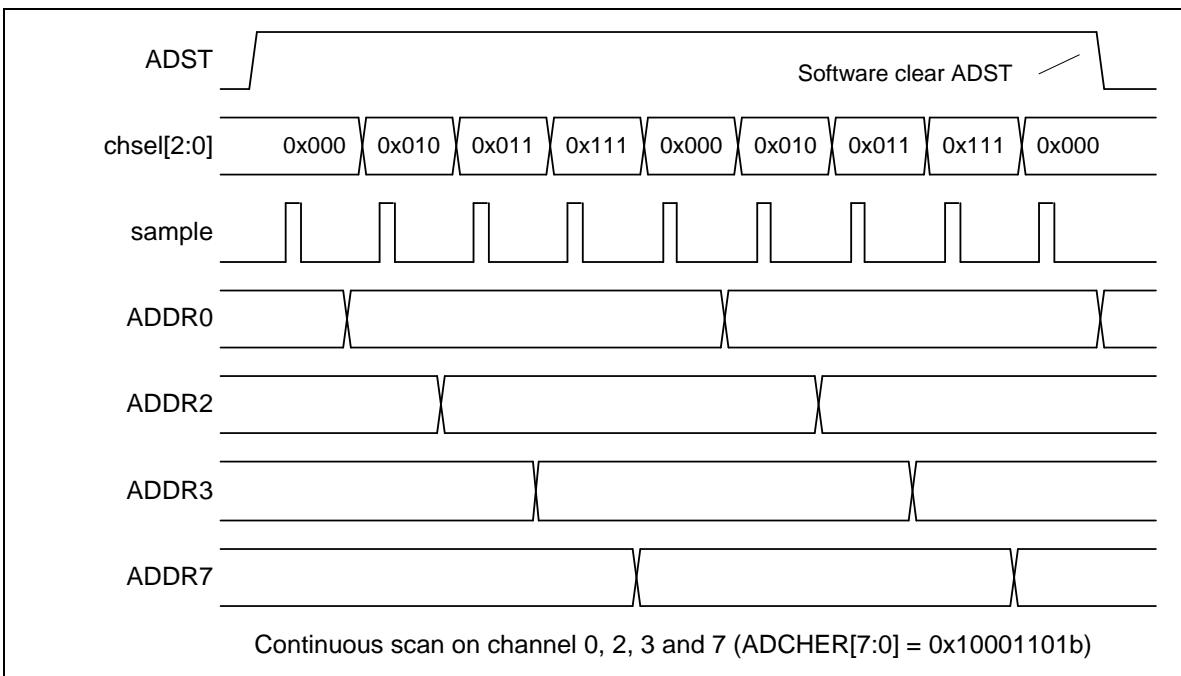


图6-11 使能通道的连续扫描时序图

#### 6.4.5.6 外部触发输入采样和 A/D 转换时间

单周期扫描模式下, A/D 转换可通过外部引脚触发。当 ADCR寄存器的TRGEN 置位, 使能 ADC 外部触发功能以后, 配置 TRGS[1:0] 位为 00b选择来自 STADC 引脚的外部触发输入. 软件可以设定 TRGCOND[1:0] 选择触发方式为上升沿/下降沿 或低电平/高电平触发。若选择 电平触发, STADC 需保持定义的电平状态至少 8 个PCLK周期. 在第9个PCLK时钟时ADST 位置位, 开始转换, 电平触发模式下, 如果外部触发输入保持为有效状态, 转换连续进行. 仅当外部触发条件消失才停止。若选择边沿触发模式, 高或低电平状态至少需保持 4 个PLCKs周期. 脉冲短于该值时, 将被忽略。

**注:** 使能ADC外设时钟之后, 用户必须使能外部触发功能或者使能ADC至少4个PCLKs时钟时间

#### 6.4.5.7 PWM触发

单周期扫描模式下，ADC转换可以由PWM触发(只有M05xxDN/DE)。当TRGEN设为高使能ADC外部硬件触发功能时，TRGS[1:0]设为11b选择来自PWM触发的外部硬件触发输入。当PWM触发使能时，设定PTDT[7:0]可以在PWM触发和ADC开始转换之间插入延迟时间。

#### 6.4.5.8 比较模式下 AD转换结果监控

NuMicro™ M05xxBN/DN/DE 系列ADC控制器提供2个比较寄存器 ADCMPR0 和ADCMPR1，来监控最多两个指定通道的转换结果。可通过软件设定CMPCH(ADCMPRx[5:0])选择监控的通道，CMPCOND位用于决定比较条件：转换结果小于或大于等于在CMPD[11:0]中指定的值。如果CMPCOND=0，当转换结果小于CMPD[11:0]中指定的值时，内部匹配计数器将增加1。如果CMPCOND=1时，当转换结果大于或者等于CMPD[11:0]中指定的值时，内部匹配计数器将增加1。当CMPCH指定的通道完成转换时，比较就被自动触发且执行一次。当比较结果和设定值相匹配，比较匹配计数器将加1，否则比较匹配计数器就清0。当计数器的值和设定值(CMPMATCNT+1)匹配，CMF 位将置1，如果CMPIE 置位 将产生ADC\_INT 中断请求. 在扫描模式下，软件可使用该功能来监控外部模拟输入引脚电压变化而不会增加程序负载。具体逻辑框图如下所示。

**注：**如果用户使能差分输入模式，转换结果可以用二进制无符号或者2的补码(有符号数)表示。软件可以设定ADSR寄存器的DMOF[31]位来选择输出格式。M05xxBN 中当差分输入模式使能时，数字比较功能不支持2的补码输出格式， DMOF位必须被清成0。

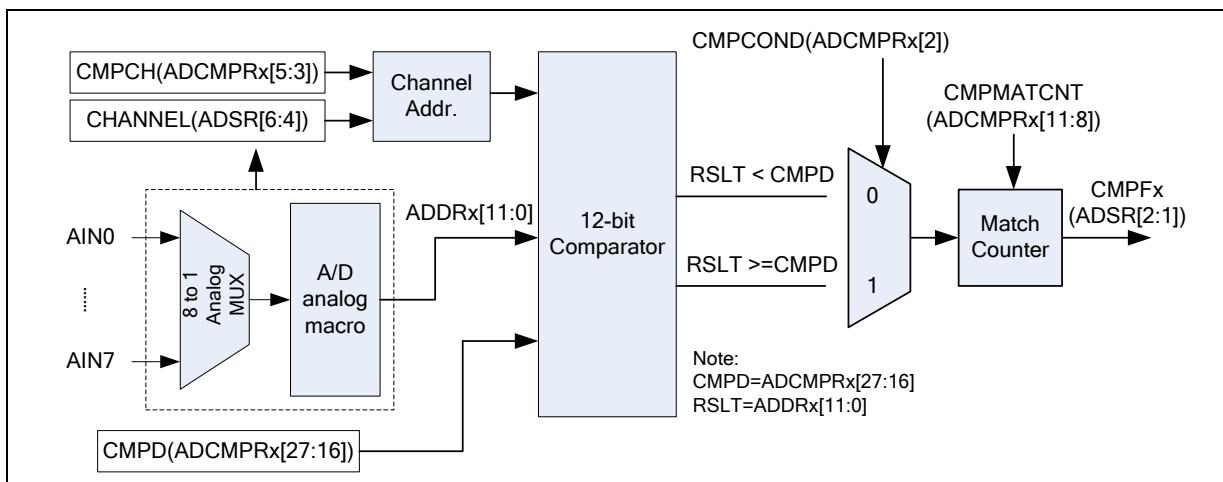


图 6-12 A/D 转换结果监控框图

#### 6.4.5.9 中断源

ADC中断有三个中断源，A/D 转换结束时，A/D转换结束标志ADF将会被置位。CMF0和CMF1是比较功能的比较标志，当转换结果满足ADCMPR0/1的设定值，相应的标志将被置位。当ADF，CMF0和CMF1这三个标志位其中一个置位，且相应的中断使能位ADCR寄存器的ADIE位，或者ADCMPR0/1中的CMPIE位被置位，ADC中断将会产生。软件可清零中断标志来撤销中断。

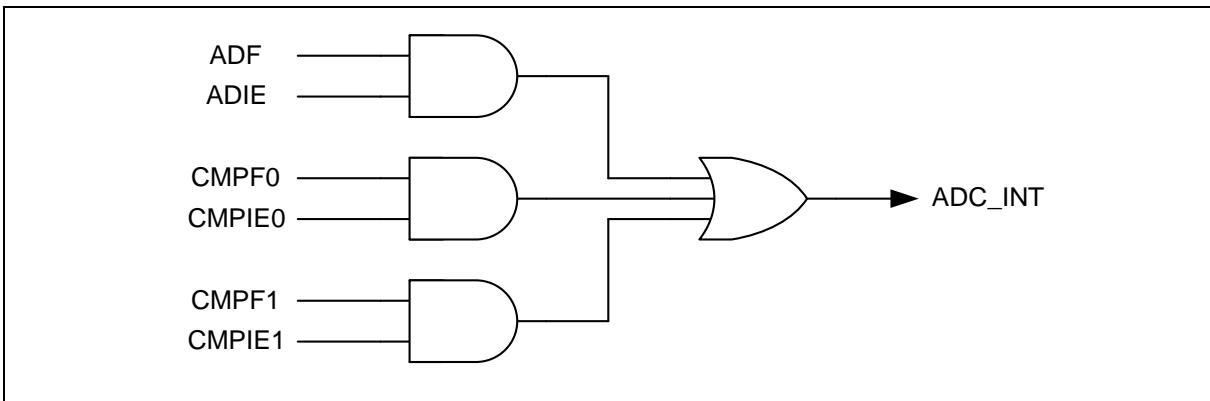


图 6-13 A/D 控制器中断



#### 6.4.6 ADC 寄存器映射

R: 只读, W: 只写, R/W: 可读写, C:仅在值为0时可写

寄存器	偏移量	R/W	描述	复位后的值
<b>ADC基址:</b>				
<b>ADC_BA = 0x400E_0000</b>				
<b>ADDR0</b>	ADC_BA+0x00	R	ADC数据寄存器 0	0x0000_0000
<b>ADDR1</b>	ADC_BA+0x04	R	ADC数据寄存器 1	0x0000_0000
<b>ADDR2</b>	ADC_BA+0x08	R	ADC数据寄存器 2	0x0000_0000
<b>ADDR3</b>	ADC_BA+0x0C	R	ADC数据寄存器 3	0x0000_0000
<b>ADDR4</b>	ADC_BA+0x10	R	ADC数据寄存器 4	0x0000_0000
<b>ADDR5</b>	ADC_BA+0x14	R	ADC数据寄存器 5	0x0000_0000
<b>ADDR6</b>	ADC_BA+0x18	R	ADC数据寄存器 6	0x0000_0000
<b>ADDR7</b>	ADC_BA+0x1C	R	ADC数据寄存器 7	0x0000_0000
<b>ADCR</b>	ADC_BA+0x20	R/W	ADC控制寄存器	0x0000_0000
<b>ADCHER</b>	ADC_BA+0x24	R/W	ADC 通道使能寄存器	0x0000_0000
<b>ADCMR0</b>	ADC_BA+0x28	R/W	ADC比较寄存器0	0x0000_0000
<b>ADCMR1</b>	ADC_BA+0x2C	R/W	ADC比较寄存器1	0x0000_0000
<b>ADSR</b>	ADC_BA+0x30	R/W	ADC状态寄存器	0x0000_0000
<b>ADTDCR</b>	ADC_BA+0x44	R/W	ADC触发延迟控制寄存器(M05xxDN/DE)	0x0000_0000

### 6.4.7 ADC 寄存器描述

#### ADC数据寄存器 (ADDR0 ~ ADDR7)

寄存器	偏移量	R/W	描述	复位后的值
ADDR0	ADC_BA+0x00	R	A/D数据寄存器 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D数据寄存器 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D数据寄存器 2	0x0000_0000
ADDR3	ADC_BA+0x0c	R	A/D数据寄存器 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D数据寄存器 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D数据寄存器 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D数据寄存器 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D数据寄存器 7	0x0000_0000

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

Bits	描述	
[31:18]	保留	-
[17]	VALID	<b>有效标志位(只读)</b> 1 = RSLT[11:0] 中的数据 有效. 0 = RSLT[11:0] 中的数据 无效. 相应模拟通道 转换完成后，该位将置位，读ADDR 寄存器后，该位由硬件清0. <b>注：</b> 当ADC正在转换时，如果用户想监控某个通道的VALID标志，用户应该轮询ADSR寄存器的VALID位代替轮询该位（M05xxBN）；M05xxDN/DE没有这个限制。
[16]	OVERRUN	<b>溢出标志位(只读)</b> 1 = RSLT[11:0] 中的数据发生覆盖. 0 = RSLT[11:0] 中的数据没有发生覆盖. 新的转换结果装载至寄存器之前，若 RSLT[11:0] 的数据没有被读走，，OVERRUN 将置 1， 读ADDR 寄存器后，该位由硬件清0..
[15:12]	保留	-

[11:0]	RSLT	A/D 转换结果 包括 12 位AD 转换结果.
--------	------	-----------------------------

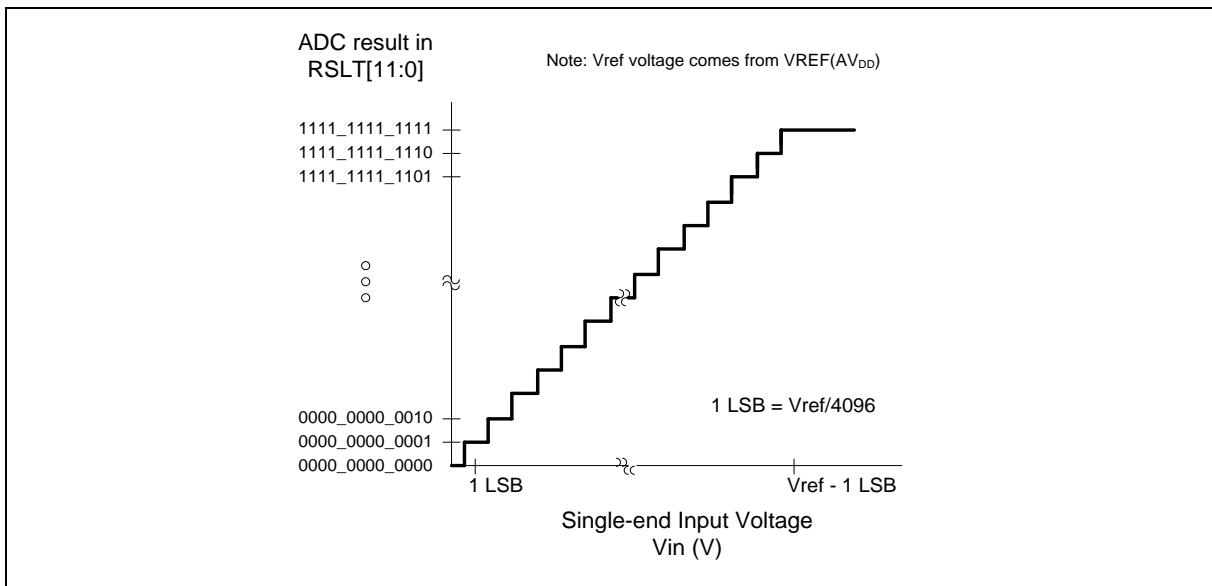


图 6-14 ADC 单端输入转换电压和转换结果映射图

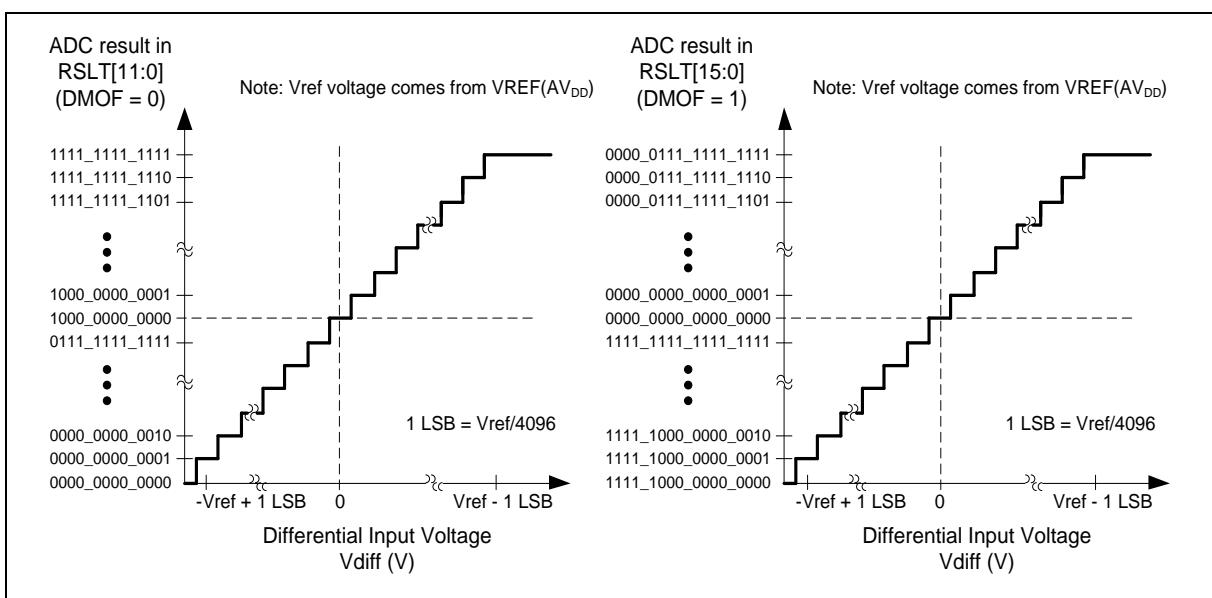


图 6-15 ADC 差分输入转换电压和转换结果映射图

**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	保留						
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留				ADST	DIFFEN	保留	TRGEN
7	6	5	4	3	2	1	0
TRGCOND		TRGS		ADMD		ADIE	ADEN

Bits	描述																					
[31]	DMOF	<b>差分输入模式输出格式</b> 1 = A/D 转换结果被存在ADDRx寄存器的RSLT中，格式为转换结果2的补码。 0 = A/D 转换结果被存在ADDRx寄存器的RSLT中，格式为无符号数(直接二进制格式) <b>注：</b> 突发模式和ADC比较功能不支持2的补码输出格式，M05xxBN中DMOF必须被清成0																				
[31:12]	保留	-																				
[11]	ADST	<b>A/D 转换开始</b> ADST 位等于1有下列 2 种方式：：软件设定和外部STADC引脚。单次转换模式和单周期扫描模式下，转换完成后，ADST 将被硬件自动清0。在连续扫描和突发模式下，A/D 转换将一直进行直到软件写0到该位或系统复位 1 = 转换开始。 0 = 转换结束，A/D进入空闲状态。 <b>注：</b> M05xxBN中，ADST位被清成0以后，用户必须等待至少1个ADC时钟周期以后才能再次将ADST设为1																				
[10]	DIFFEN	<b>A/D 差分输入模式使能</b> 1 = A/D为差分输入模式 0 = A/D为单端输入模式 <table border="1" data-bbox="682 1707 1347 1939"> <tr> <td colspan="2">差分输入配对通道</td> <td colspan="2">模拟输入</td> </tr> <tr> <td colspan="2"></td> <td>V<sub>plus</sub></td> <td>V<sub>minus</sub></td> </tr> <tr> <td colspan="2">0</td> <td>AIN0</td> <td>AIN1</td> </tr> <tr> <td colspan="2">1</td> <td>AIN2</td> <td>AIN3</td> </tr> <tr> <td colspan="2">2</td> <td>AIN4</td> <td>AIN5</td> </tr> </table>	差分输入配对通道		模拟输入				V <sub>plus</sub>	V <sub>minus</sub>	0		AIN0	AIN1	1		AIN2	AIN3	2		AIN4	AIN5
差分输入配对通道		模拟输入																				
		V <sub>plus</sub>	V <sub>minus</sub>																			
0		AIN0	AIN1																			
1		AIN2	AIN3																			
2		AIN4	AIN5																			

		3 AIN6 AIN7
		差分输入电压( $V_{diff}$ ) = $V_{plus} - V_{minus}$ , $V_{plus}$ 和 $V_{minus}$ 为差分输入两个引脚上的电压 <b>注:</b> 在差分输入模式下,两个相关的通道中只有偶数通道需要在ADCHER中使能. 转换结果将放置于相应的使能通道的寄存器里。
[9]	保留	-
[8]	TRGEN	<b>外部触发使能控制</b> 使能或禁止外部STADC引脚触发A/D 转换。如果外部触发被使能，ADST位可以由选中的硬件触发源设为1 1= 使能外部触发 0= 禁用外部触发 <b>注:</b> 只有在单周期扫描模式下, 才支持外部STADC引脚触发A/D转换
[7:6]	TRGCOND	<b>外部触发条件</b> 该 2 位 决定外部STADC 引脚触发为电平触发还是边沿触发。电平触发时, 该信号必须保持至少8 个PCLKs的稳定时间; 边沿触发下, 至少保持4 个PCLKs 的高电平或低电平状态。 00 = 低电平 01 = 高电平 10 = 下降沿 11 = 上升沿
[5:4]	TRGS	<b>硬件触发源</b> 00 = A/D 转换由外部STADC引脚启动 11 = A/D转换由PWM触发启动 其它 =保留 改变 TRGS 前, 软件需要先关闭TRGEN 和 ADST. <b>注:</b> M05xxBN不支持PWM触发ADC转换.
[3:2]	ADMD	<b>A/D 转换操作模式控制</b> 00 = 单次转换 01 = 突发转换 10 = 单周期扫描 11 = 连续扫描 <b>注1:</b> 当改变操作模式时, 软件要首先清除 ADST 位 <b>注2:</b> 在突发模式下, A/D转换结果总是存储在数据寄存器0中
[1]	ADIE	<b>A/D 中断使能控制</b> 1 = 使能 A/D 中断功能 0 = 禁用 A/D 中断功能 如果ADIE=1, A/D 转换结束将产生中断请求.
[0]	ADEN	<b>A/D 转换器使能</b> 1 = 使能 0 = 禁用 <b>注:</b> 开始 A/D 转换前, 该位需 置位。该位为 0 将关闭 A/D 转换模拟电路的电源以省电.



### 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
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留						PRESEL[1:0]	
7	6	5	4	3	2	1	0
CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0

Bits	描述	
[31:10]	保留	-
[9:8]	PRESEL[1:0]	模拟输入通道7输入选择 00= 外部模拟输入 01= 内部band-gap电压 10= 内部温度传感器 11= 保留 注：当软件选择内部band-gap电压作为ADC通道7的输入源时，ADC时钟频率需要小于300KHz，使用内部温度传感器需将VTEMP_EN致能
[7]	CHEN7	模拟输入通道 7使能 1 = 使能 0 = 禁用
[6]	CHEN6	模拟输入通道 6使能 如果DIFFEN位=1，只有偶数通道需要使能 1 = 使能 0 = 禁用
[5]	CHEN5	模拟输入通道 5使能 1 = 使能 0 = 禁用
[4]	CHEN4	模拟输入通道 4使能 如果DIFFEN位=1，只有偶数通道需要使能 1 = 使能 0 = 禁用
[3]	CHEN3	模拟输入通道 3使能

		1 = 使能 0 = 禁用
[2]	CHEN2	模拟输入通道 2使能 如果DIFFEN位=1，只有偶数通道需要使能 1 = 使能 0 = 禁用
[1]	CHEN1	模拟输入通道 1使能 1 = 使能 0 = 禁用
[0]	CHENO	模拟输入通道 0使能 如果DIFFEN位=1，只有偶数通道需要使能 1 = 使能 0 = 禁用

ADC 比较寄存器 0/1 (ADCMPR0/1)

寄存器	偏移量	R/W	描述	复位后的值
ADCMPR0	ADC_BA+0x28	R/W	ADC比较寄存器0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	ADC比较寄存器1	0x0000_0000

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

Bits	描述	
[31:28]	保留	-
[27:16]	<b>CMPD</b>	<b>比较数值</b> 此12位数值将和指定通道的转换结果相比较, 注: 差分输入模式下, ADC比较器将CMPD和无符号数的转换结果比较(M05xxBN), CMPD应该填无符号数。
[15:12]	保留	保留
[11:8]	<b>CMPPATCNT</b>	<b>比较匹配次数</b> 当指定A/D通道的转换值和比较条件CMPCOND[2]相匹配, 内部计数器将相应的加1.当内部计数器的值达到设定值(CMPPATCNT +1)时, 硬件将置位CMPEF位.
[5:3]	<b>CMPCNT</b>	<b>Compare 通道选择</b> 000 = 选择比较通道0 的转换结果. 001 = 选择比较通道1 的转换结果. 010 = 选择比较通道2 的转换结果. 011 = 选择比较通道3 的转换结果. 100 = 选择比较通道4 的转换结果. 101 = 选择比较通道5 的转换结果. 110 = 选择比较通道6 的转换结果. 111 = 选择比较通道7 的转换结果.
[2]	<b>CMPCOND</b>	<b>比较条件</b> 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(ADCMPRx[27:16])时, 内部匹配计数器加1. 0= 设置比较条件为当12位A/D转换结果小于12位CMPD(ADCMPRx[27:16])时, 内部

		匹配计数器加1. 注: 当内部计数器的值达到(CMPMATICNT +1), CMPFx 置位.
[1]	<b>CMPIE</b>	<b>比较中断使能</b> 如果使能比较功能, 且比较条件与CMPCOND和CMPMATICNT的设置匹配, CMPF位有效, 同时, 如果 <b>CMPIE</b> 置1, 产生比较中断请求. 1 = 使能比较功能中断 0 = 禁用比较功能中断
[0]	<b>CMPEN</b>	<b>比较使能</b> 1 = 使能比较. 0 = 禁用比较. 当转换结果放到ADDR 寄存器时, 该位置位将使能ADC控制器将CMPD[11:0]与特定通道的转换值进行比较。

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	undefined

31	30	29	28	27	26	25	24
保留							
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
保留	CHANNEL			BUSY	CMPF1	CMPF0	ADF

Bits	描述	
[31:24]	保留	-
[23:16]	<b>OVERRUN</b>	<b>溢出标志 (只读)</b> 为ADDRx的OVERRUN位的镜像 当ADC工作于突发模式时，若FIFO溢出，OVERRUN[7:0] 全部置1.
[15:8]	<b>VALID</b>	<b>数据有效标志位(只读)</b> 为ADDRx的VALID位的镜像 当ADC工作于突发模式时，若FIFO有效，VALID [7:0] 全部置1.
[7]	保留	保留
[6:4]	<b>CHANNEL</b>	<b>当前转换通道</b> 这3位在BUSY=1时 表示正在进行转换中的通道. 当BUSY=0, 表示下次进行转换的通道. 只读位.
[3]	<b>BUSY</b>	<b>忙/空闲(只读)</b> 该位是ADST 位(ADCR). 的镜像 1 = A/D 转换器忙碌 0 = A/D 转换器空闲 只读位.
[2]	<b>CMPF1</b>	<b>比较标志位1</b> 当选择的 A/D通道转换结果和ADCMR1相匹配时，该位置1。写 1 清该位. 1 = ADDR 转换结果和 ADCMR1的设定相匹配 0 = ADDR 转换结果和 ADCMR1不匹配
[1]	<b>CMPF0</b>	<b>比较标志位0</b> 当选择的 A/D通道转换 结果和ADCMR0相匹配时，该位置1。写 1 清该位.

		1 = ADDR 转换结果和 ADCMP0的设定相匹配 0 = ADDR 转换结果和 ADCMP0不匹配
[0]	<b>ADF</b>	<b>A/D转换结束标志位</b> 状态标志位 指示A/D 转换结束。该标志写1清除为零. ADF 在下列三个条件时置位: 1. 单次转换模式下A/D转换结束时 2. 单次扫描模式和连续扫描模式下所有指定通道A/D转换结束时. 3. 突发模式下, FIFO存储多于4个转换结果

ADC 触发延迟控制寄存器 (ADTDCR)

寄存器	偏移量	R/W	描述	复位后的值
ADTDCR	ADC_BA+0x44	R/W	ADC 触发延迟控制寄存器 (只有M05xxDN/DE)	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
PTDT							

Bits	描述	
[31:8]	保留	保留.
[7:0]	PTDT	<b>PWM 触发延迟时间</b> PWM触发后，设置该域将延迟ADC启动转换的时间. $\text{PWM 触发延迟时间} = (4 * \text{PTDT}) * \text{system clock}$

## 6.5 时钟控制器

### 6.5.1 概述

时钟控制器为整个芯片提供时钟，包括系统时钟和所有外设时钟。时钟控制器还利用独立的时钟ON/OFF控制、时钟源选择和时钟分频器，实现电源控制功能。在CPU置位掉电使能位PWR\_DOWN\_EN (PWRCON[7])和PD\_WAIT\_CPU (PWRCON[8])且Cortex-M0执行WFI指令之后，芯片会进入掉电模式，在那之后，芯片等待唤醒中断源被触发以离开掉电模式。在掉电模式下，时钟控制器关闭外部4~24 MHz高速晶振(HXT)和内部22.1184MHz高速振荡器(HIRC)，以降低整个系统的功耗。下图显示时钟发生器和时钟源控制。

时钟发生器由如下4个时钟源组成：

- 一个外部 4~24 MHz 高速晶振(HXT)
- 一个内部 22.1184 MHz RC 高速振荡器(HIRC)
- 一个可编程的 PLL FOUT(PLL时钟源可以选择外部4~24MHz高速晶振(HXT)或者内部22.1184MHz (HIRC)高速振荡器) (PLL FOUT)
- 一个内部 10KHz 低速振荡器(LIRC)

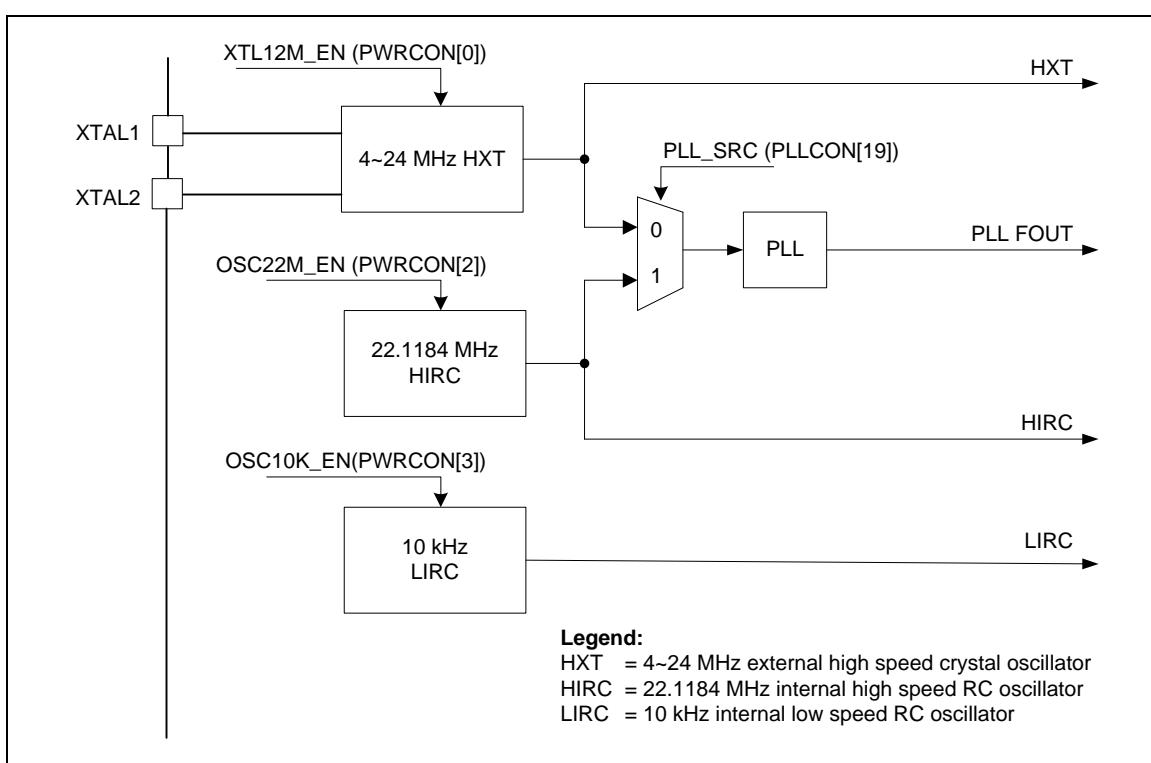


图 6-16 时钟发生器框图

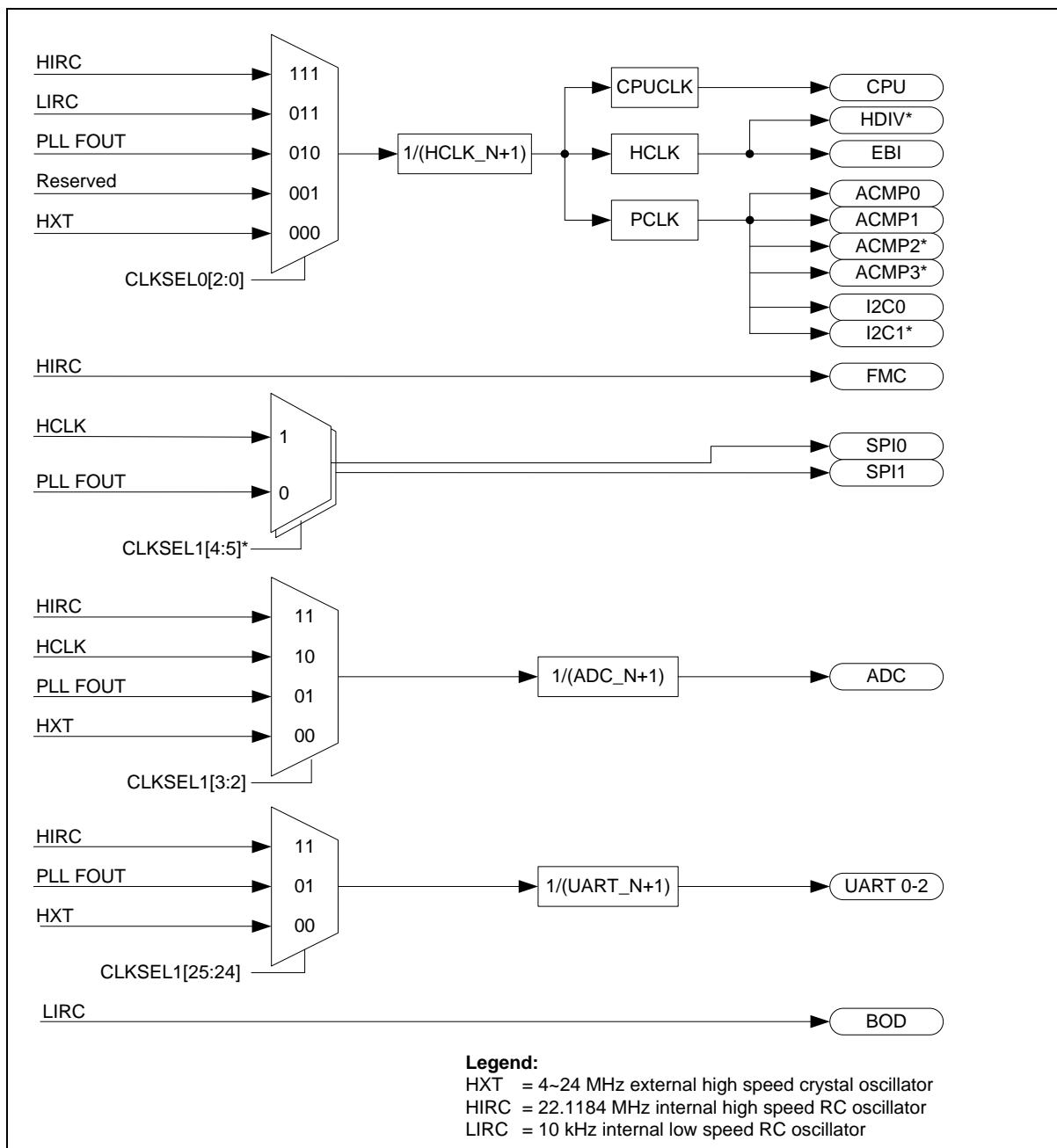


图6-17 时钟源控制器(1/2)

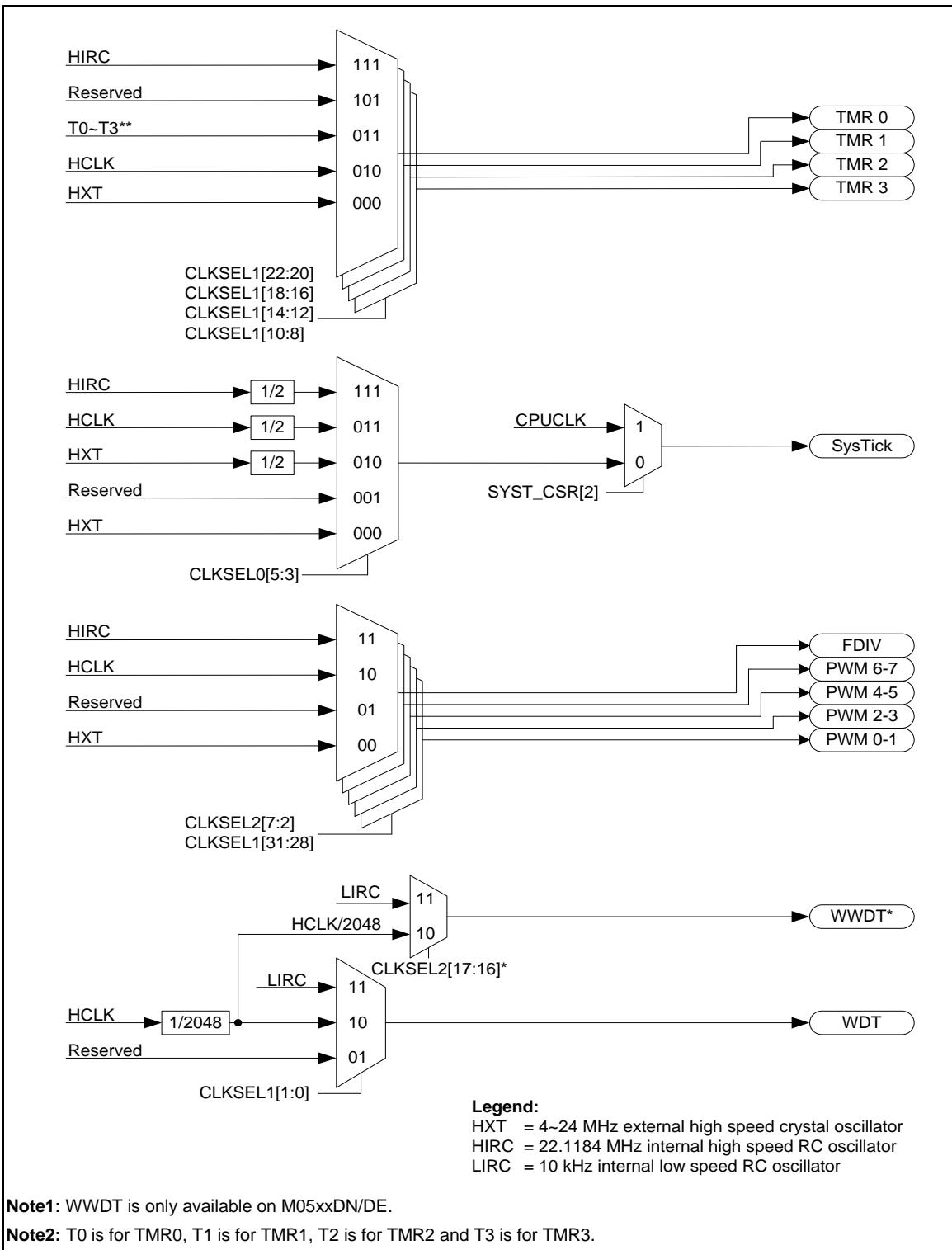


图 6-18 时钟源控制器 (2/2)

### 6.5.2 系统时钟 & SysTick 时钟

系统时钟有4个时钟源，由时钟发生器模块产生。使用寄存器HCLK\_S(CLKSEL0[2:0])可以切换不同的时钟，系统时钟框图如下所示。

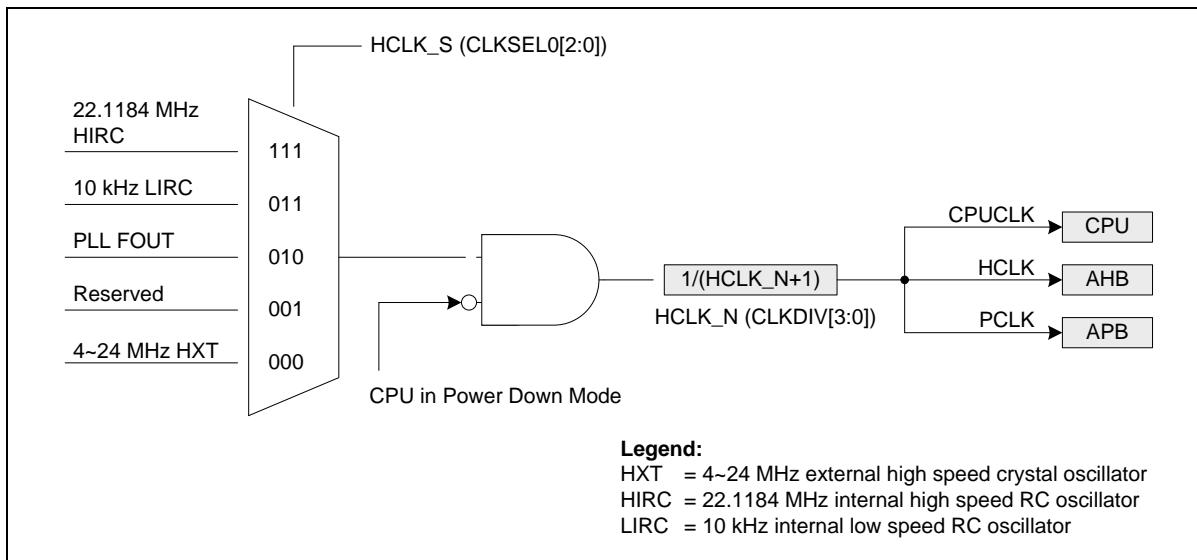


图6-19 系统时钟框图

在Cortex-M0核中的SysTick的时钟源可以使用CPU时钟或者外部时钟(SYST\_CSR[2])。如果使用外部时钟，SysTick时钟(STCLK)有4个时钟源。时钟源切换使用寄存器STCLK\_S(CLKSEL0[5:3])的设置。SysTick时钟框图如下所示。

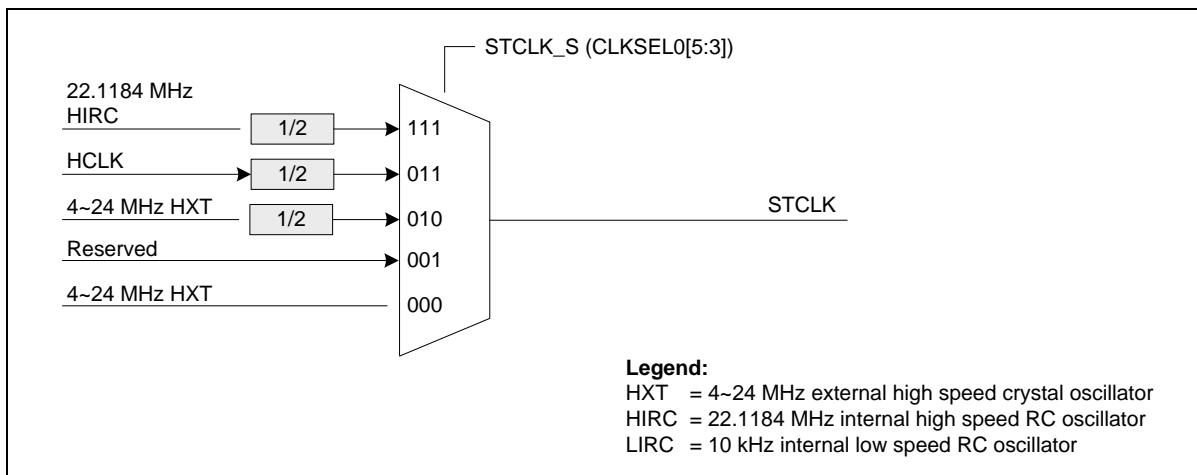


图6-20 SysTick时钟控制框图

### 6.5.3 掉电模式（深度睡眠模式）时钟

当芯片进入掉电模式后，大部分时钟源、外设时钟和系统时钟将会被禁用，如果在CPU进入掉电模式之前没有关闭内部10K，它将保持有效，一些选择10K做时钟源的外设仍可以处于激活状态。

如下外设仍然可以保持激活(当这些IP采用内部10KHz低速振荡器作时钟源时)

- 看门狗时钟
- Timer 0/1/2/3 时钟
- PWM 时钟

### 6.5.4 分频器输出

该设备包含一个由16级2分频移位寄存器组成的分频器。其中哪一级的值被输出由一个16选1的多路转换器选择，并被映射到CKO引脚P3.6输出。所以有16种以2为幂的时钟分频选择，频率从  $F_{in}/2^1$  到  $F_{in}/2^{16}$ ，其中  $F_{in}$  为输入到时钟分频器的时钟频率。

输出公式：  $F_{out} = F_{in}/2^{(N+1)}$ ，其中  $F_{in}$  为输入时钟频率， $F_{out}$  为时钟分频输出频率，N 为FSEL (FRQDIV[3:0]) 中的4位值

当写1到DIVIDER\_EN(FRQDIV[4])时，链计数器开始计数，当写0到DIVIDER\_EN(FRQDIV[4])时，链计数器持续计数直到分频时钟达到低状态并停留在低状态。

M05xxDN/DE中，如果 DIVIDER1(FRQDIV[5]) = 1，时钟分频器 (FRQDIV\_CLK) 将绕过2的倍数频率分频器。频率分配器的时钟将由CKO引脚直接输出.

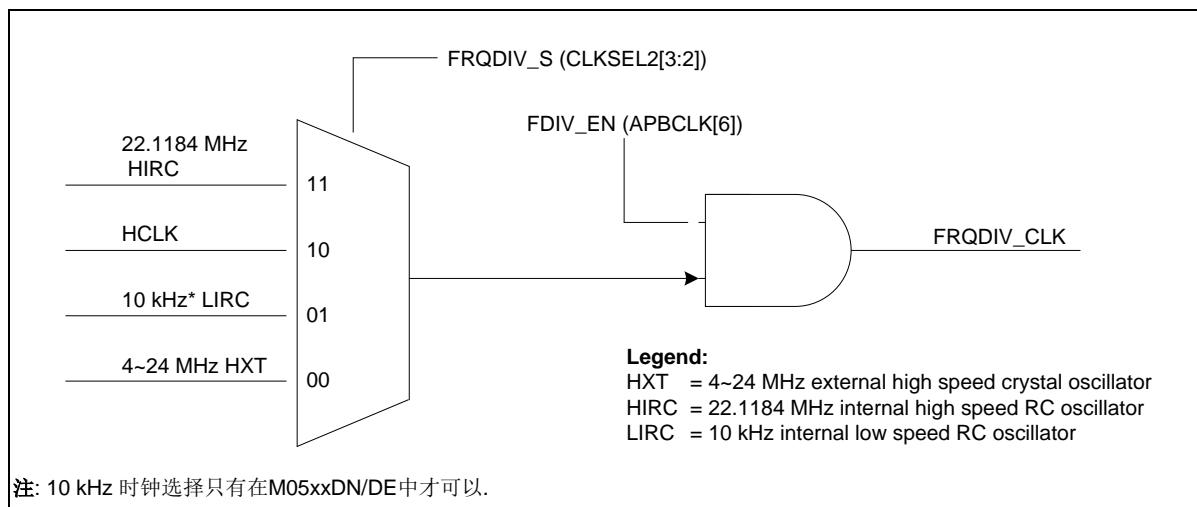


图6-21 分频器的时钟源

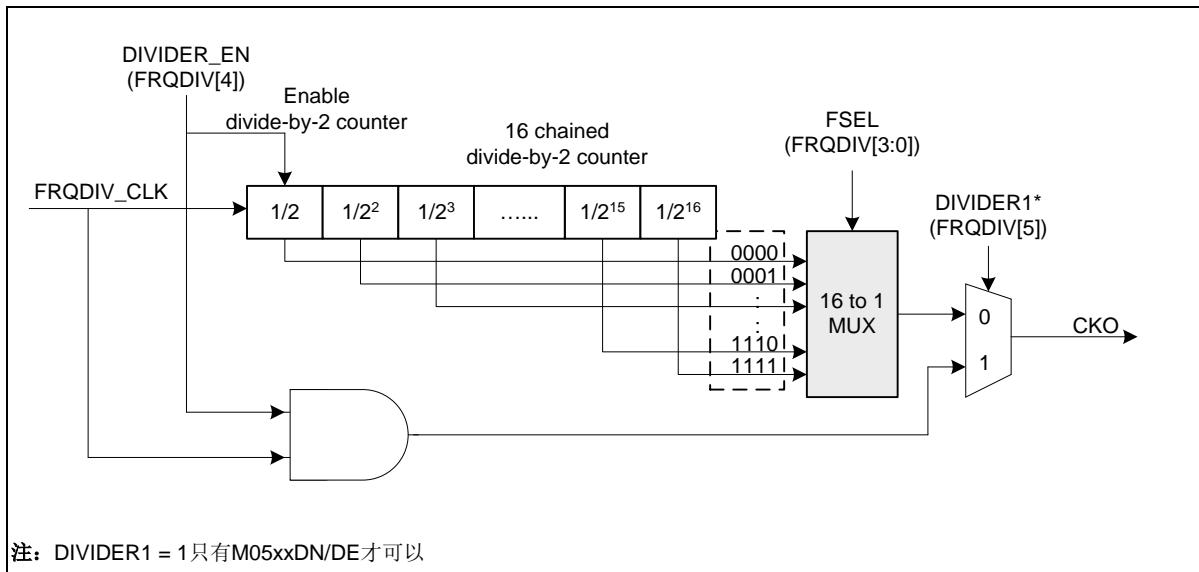


图 6-22 分频器框图

### 6.5.5 时钟控制寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>CLK基地址:</b>				
<b>CLK_BA = 0x5000_0200</b>				
PWRCON	CLK_BA + 00	R/W	系统掉电控制寄存器	0x0000_000X
AHBCLK	CLK_BA + 04	R/W	AHB 设备时钟使能控制寄存器	0x0000_000D
APBCLK	CLK_BA + 08	R/W	APB 设备时钟使能控制寄存器	0x0000_000X
CLKSTATUS	CLK_BA + 0C	R/W	时钟状态监控寄存器	0x0000_00XX
CLKSEL0	CLK_BA + 10	R/W	时钟源选择控制寄存器0	0x0000_003X
CLKSEL1	CLK_BA + 14	R/W	时钟源选择控制寄存器1	0xFFFF_FFFF
CLKDIV	CLK_BA + 18	R/W	时钟分频数目寄存器	0x0000_0000
CLKSEL2	CLK_BA + 1C	R/W	时钟源选择控制寄存器2	0x0000_00FF
PLLCON	CLK_BA + 20	R/W	PLL控制寄存器	0x0005_C22E
FRQDIV	CLK_BA + 24	R/W	分频器控制寄存器	0x0000_0000

### 6.5.6 寄存器描述

#### 掉电控制寄存器 (PWRCON)

除BIT[6]外, PWRCON的其他位都受保护。要编程这些被保护的位需要向地址0x5000\_0100依次写入"59h", "16h", "88h"以禁用寄存器保护。参考寄存器REGWRPROT, 地址GCR\_BA + 0x100

寄存器	偏移量	R/W	描述	复位后的值
PWRCON	CLK_BA +0x 00	R/W	系统掉电控制寄存器	0x0000_001X

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							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	保留	XTL12M_EN

Bits	描述	
[31:9]	保留	保留
[8]	PD_WAIT_CPU	控制进入掉电模式的条件(写保护) 1 = PWR_DOWN_EN置1并且CPU执行WFI指令时, 芯片进入掉电模式。 0 = PWR_DOWN_EN 置1时, 芯片进入掉电模式
[7]	PWR_DOWN_EN	使能系统掉电模式(写保护) 该位置“1”, 使能芯片的掉电模式, 掉电行为取决于PD_WAIT_CPU 位 (a) PD_WAIT_CPU 为“0”, 置位PWR_DOWN_EN 后, 芯片马上进入掉电模式。 (b) PD_WAIT_CPU 为“1”, 置位PWR_DOWN_EN 后, 芯片不马上进入掉电模式, 要等CPU 执行WFI指令之后, 芯片才进入掉电模式  芯片由掉电模式唤醒时, 该位由硬件自动清零, 在下次掉电时, 用户需要重新置位该位。 掉电 模式下, 4~24MHz外部高速晶振 (HXT) 与22.1184 MHz内部高速振荡器(HIRC)被关闭, 内部10 KHz低速振荡器(LIRC)不受该位控制  掉电时, PLL 与系统时钟也被关闭, ,时钟源选择被忽略. 如果外设以10KHz低速振荡器作为时钟, 则其时钟不受该位控制 1 = 芯片立即进入掉电模式 或等待CPU休眠命令WFI 0 = 芯片工作于正常模式或CPU执行WFI指令进入空闲模式
[6]	PD_WU_STS	掉电唤醒状态标志 芯片从掉电模式下被唤醒时, 该位置位 例如: 如果GPIO, UART, WDT, ACMP或者BOD唤醒CPU, 该标志置位

		注: 这个比特只有在 PD_WU_INT_EN (PWRCON[5]) 被置为 1的时候才会被置位写1清除为零.
[5]	PD_WU_INT_EN	<p><b>掉电模式唤醒中断使能(写保护)</b>            0 = 禁用            1 = 使能。从掉电唤醒时，产生中断。            注：当 PD_WU_STS 和 PD_WU_INT_EN 都为高时中断将发生</p>
[4]	PD_WU_DLY	<p><b>唤醒延迟计数器使能(写保护).</b>            当芯片从掉电模式唤醒时，该域可以控制延迟一定时钟周期以等待系统时钟稳定。            当芯片工作于外部4~24MHz高速晶振（HXT）时，延迟时间为4096 个时钟周期，工作于内部22.1184MHz(HIRC)时，延迟256 个时钟周期。            1 = 使能时钟周期延迟            0 = 禁用时钟周期延迟</p>
[3]	OSC10K_EN	<p><b>内部 10KHz 低速振荡器使能控制 (写保护)</b>            1 = 使能10KHz 低速振荡器            0 =禁用10KHz 低速振荡器</p>
[2]	OSC22M_EN	<p><b>内部 22.1184MHz高速振荡器使能控制 (写保护)</b>            1 = 使能22.1184MHz高速振荡器            0 =禁用22.1184MHz高速振荡器</p>
[1]	保留	保留
[0]	XTL12M_EN	<p><b>外部 12MHz晶振使能控制 (写保护)</b>            该位的默认值由flash用户配置寄存器config0 [26:24] 设置。当默认时钟源为外部高速晶振（4~24MHz时）。该位自动置1            1 = 使能晶振            0 =禁用晶振</p>

指令模式	寄存器/SLEEPDEEP (SCR[2])	PWR_DOWN_EN (PWRCON[8])	PD_WAIT_CPU (PWRCON[7])	CPU 运行 WFE/WFI 指令	时钟关闭
正常运行模式	0	0	0	NO	软件通过控制寄存器关闭所有时钟
IDLE 模式 (CPU 进入休眠模式)	0	X	0	YES	仅CPU内部时钟关闭
Power_down Mode (CPU 进入深度休眠模式)	1	1	1	YES	大部分时钟关闭，仅外部10K与WDT/Timer/PWM/ADC可能仍然处于激活状态

表 6-6 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唤醒CPU。在设定PWR\_DOWN\_EN位(PWRCON[7])之前用户应该使能相关中断和NVIC IRQ使能位(NVIC\_ISER)，以确保进入掉电模式以后可以成功唤醒。

AHB设备时钟使能控制寄存器 (AHBCLK)

该寄存器中各位用于使能/禁用系统、HDIV与EBI时钟

寄存器	偏移量	R/W	描述				复位后的值
AHBCLK	CLK_BA + 04	R/W	AHB设备时钟使能控制寄存器				0x0000_000D

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			HDIV_EN	EBI_EN	ISP_EN	保留	保留

Bits	描述	
[31:5]	保留	保留
[4]	HDIV_EN	除法器时钟使能控制 (只有M05xxDN/DE) 0 = 除法器时钟关闭 1 = 除法器时钟使能.
[3]	EBI_EN	EBI 控制器时钟使能控制. 1 = 使能 EBI控制器时钟. 0 = 禁用 EBI控制器时钟.
[2]	ISP_EN	Flash ISP 控制器时钟使能控制. 1 = 使能 Flash ISP控制器时钟. 0 = 禁用 Flash ISP控制器时钟.
[1:0]	保留	保留



### APB 设备时钟使能控制寄存器 (APBCLK)

该寄存器中的各位用于使能/禁用外设时钟.

寄存器	偏移量	R/W	描述	复位后的值			
APBCLK	CLK_BA +0x 08	R/W	APB设备时钟使能控制寄存器	0x0000_000X			

31	30	29	28	27	26	25	24
ACMP23_EN	ACMP01_EN	保留	ADC_EN	保留			
23	22	21	20	19	18	17	16
PWM67_EN	PWM45_EN	PWM23_EN	PWM01_EN	保留		UART1_EN	UART0_EN
15	14	13	12	11	10	9	8
保留		SPI1_EN	SPI0_EN	保留		I2C1_EN	I2C0_EN
7	6	5	4	3	2	1	0
保留	FDIV_EN	TMR3_EN	TMR2_EN	TMR1_EN	TMR0_EN	保留	WDT_EN

Bits	描述	
[31]	ACMP23_EN	模拟比较器 2/3 时钟使能控制 (M051xxDE Only) 1 = 使能 ACMP 2/3时钟. 0 = 禁用ACMP 2/3时钟.
[30]	ACMP_EN	模拟比较器0/1时钟使能控制 1 = 使能 ACMP 0/1时钟 0 = 禁用 ACMP 时钟
[29]	保留	保留
[28]	ADC_EN	ADC时钟使能控制 1 = 使能 ADC 时钟 0 = 禁用 ADC 时钟
[27:24]	保留	保留
[23]	PWM67_EN	PWM_67 时钟使能控制 1 = 使能 PWM67 时钟 0 = 禁用 PWM67 时钟
[22]	PWM45_EN	PWM_45时钟使能控制 1 = 使能 PWM45 时钟 0 = 禁用 PWM45 时钟
[21]	PWM23_EN	PWM_23时钟使能控制 1 = 使能 PWM23 时钟 0 = 禁用 PWM23 时钟

[20]	<b>PWM01_EN</b>	<b>PWM_01时钟使能控制</b> 1 = 使能 PWM01时钟 0 = 禁用 PWM01 时钟
[19:18]	保留	保留
[17]	<b>UART1_EN</b>	<b>UART1 时钟使能控制</b> 1 = 使能 UART1 时钟 0 = 禁用 UART1 时钟
[16]	<b>UART0_EN</b>	<b>UART0 时钟使能控制</b> 1 = 使能 UART0 时钟 0 = 禁用 UART0 时钟
[15:14]	保留	保留
[13]	<b>SPI1_EN</b>	<b>SPI1 时钟使能控制</b> 1 = 使能 SPI1 时钟 0 = 禁用 SPI1 时钟
[12]	<b>SPI0_EN</b>	<b>SPI0 时钟使能控制</b> 1 = 使能 SPI0 时钟 0 = 禁用 SPI0 时钟
[11:10]	保留	保留
[9]	<b>I2C1_EN</b>	<b>I2C1 时钟使能控制(只有M051xxDN/DE )</b> 1 = 使能 I2C1 时钟 0 = 禁用 I2C1 时钟
[8]	<b>I2C0_EN</b>	<b>I2C0 时钟使能控制</b> 1 = 使能 I2C0 时钟 0 = 禁用 I2C0 时钟
[7]	保留	保留
[6]	<b>FDIV_EN</b>	<b>分频器输出时钟使能控制</b> 0 = 禁用FDIV时钟 1 = 使能FDIV时钟
[5]	<b>TMR3_EN</b>	<b>Timer3 时钟使能控制</b> 0 = 禁用定时器3时钟 1 = 使能定时器3时钟
[4]	<b>TMR2_EN</b>	<b>Timer2 时钟使能控制</b> 0 = 禁用定时器2时钟 1 = 使能定时器2时钟
[3]	<b>TMR1_EN</b>	<b>Timer1 时钟使能控制</b> 0 = 禁用定时器1时钟 1 = 使能定时器1时钟

[2]	<b>TMR0_EN</b>	Timer0 时钟使能控制 0 = 禁用定时器0时钟 1 = 使能定时器0时钟
[1]	保留	保留
[0]	<b>WDT_EN</b>	看门狗 时钟使能控制 (写保护) 0 = 禁用看门狗时钟 1 = 使能看门狗时钟 <b>注:</b> 该位是写保护的, 对该位编程时, 需要向0x5000_0100 依次写入“59h”, “16h”, “88h”来解除寄存器写保护, 参考寄存器REGWRPROT, 地址GCR_BA+0x100 默认值由flash控制器用户可配置寄存器congig0 bit[31]设置

**时钟状态寄存器（CLKSTATUS）**

该寄存器各位用于监控芯片时钟是否稳定，时钟切换是否失败。

寄存器	偏移量	R/W	描述	复位后的值
CLKSTATUS	CLK_BA + 0C	R/W	时钟状态监控寄存器	0x0000_00XX

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
CLK_SW_FAIL	保留		OSC22M_STB	OSC10K_STB	PLL_STB	保留	XTL12M_STB

Bits	描述	
[31:8]	保留	保留-
[7]	CLK_SW_FAIL	<p>时钟切换失败标志 1 = 时钟切换失败 0 = 时钟切换成功</p> <p><b>注1：</b>当软件切换系统时钟源时，该位被更新。如果切换目标时钟源稳定，该位清0；如果切换目标时钟源不稳定，该位置位</p> <p><b>注2：</b> M05xxBN中，写1清除为零。</p> <p><b>注3：</b> M05xxDN/DE中，该位是只读的。选择的时钟源稳定后，硬件将自动切换系统时钟到选择的时钟源，CLK_SE_FAIL将被硬件自动清0</p>
[6:5]	保留	保留
[4]	OSC22M_STB	<p>内部 22.1184 MHz 高速振荡器(HIRC)稳定标志 (只读) 1 = 内部 22.1184 MHz 高速振荡器(HIRC)稳定 0 = 内部 22.1184 MHz 高速振荡器(HIRC)不稳定或没有使能</p>
[3]	OSC10K_STB	<p>内部10KHz低速震荡器(LIRC)稳定标志 (只读) 1 = 内部10KHz低速震荡器(LIRC)稳定 0 = 内部10KHz低速震荡器(LIRC)不稳定或没有使能</p>
[2]	PLL_STB	<p>PLL 时钟稳定标志 (只读) 1 = PLL 时钟稳定 0 = PLL 时钟不稳定或没有使能</p>
[1]	保留	-
[0]	XTL12M_STB	<p>外部4-24MHz高速晶振稳定标志 (只读) 1 = 外部4-24MHz高速晶振稳定</p>



		0 = 外部4-24MHz高速晶振不稳定或没有使能
--	--	---------------------------

时钟源选择控制寄存器0 (CLKSEL0)

寄存器	偏移量	R/W	描述	复位后的值
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
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		<b>STCLK_S</b>			<b>HCLK_S</b>		

Bits	描述	
[31:6]	保留	保留
[5:3]	<b>STCLK_S</b>	<p><b>Cortex®-M0 SysTick 时钟源选择 (写保护) .</b></p> <p>如果SYST_CSR[2] = 1, SysTick 时钟源来自HCLK.</p> <p>如果SYST_CSR[2] = 0, SysTick 时钟源由如下设定决定</p> <p>000 = 外部高速晶振HXT (4~24MHz)</p> <p>001 = 保留</p> <p>010 =外部高速晶振 (4~24MHz) HXT/2分频</p> <p>011 = 时钟源来自HCLK/2</p> <p>111 = 内部 22.1184MHz 高速振荡器HIRC/2分频.</p> <p>其它 = 保留</p> <p><b>注1:</b> 该位是受保护位, 对该位编程时, 需要向0x5000_0100 依次写入“59h”, “16h”, “88h” 来解除寄存器写保护, 参考寄存器REGWRPROT, 地址为GCR_BA + 0x100</p> <p><b>注2:</b> 如果SysTick 时钟源不是来自HCLK(也就是SYST_CSR[2]=0), SysTick时钟源必须小于或者等于HCLK/2</p>

[2:0]	HCLK_S	<p><b>HCLK</b> 时钟源选择（写保护）。</p> <p>000 = 外部高速晶振HXT (4~24MHz)</p> <p>001 = 保留</p> <p>010 = 时钟源来自PLL 时钟</p> <p>011 = 内部 10KHz 低速振荡器时钟LIRC</p> <p>111 = 内部 22.1184MHz 高速振荡器时钟HIRC</p> <p>Others = 保留.</p> <p><b>注1：</b>在时钟切换之前，相关时钟源(当前和新选)都必须打开并且稳定</p> <p><b>注2：</b>任何复位后，默认值将从用户配置寄存器<b>CFOSC(CONFIG0[26:24])</b> 加载，所以默认值可为000b 或 111b.</p> <p><b>注3：</b>该位是写保护位，对该位编程时，需要向0x5000_0100 依次写入“59h”，“16h”，“88h” 来解除寄存器写保护，参考寄存器REGWRPROT，地址为GCR_BA + 0x100</p>
-------	--------	---

**时钟源选择控制寄存器1 (CLKSEL1)**

在时钟切换前，相关的时钟源（当前和新选），必须都打开。

寄存器	偏移量	R/W	描述				复位后的值
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
<b>PWM23_S</b>		<b>PWM01_S</b>			保留		<b>UART_S</b>
23	22	21	20	19	18	17	16
保留	<b>TMR3_S</b>			保留	<b>TMR2_S</b>		
15	14	13	12	11	10	9	8
保留	<b>TMR1_S</b>			保留	<b>TMR0_S</b>		
7	6	5	4	3	2	1	0
保留		<b>SPI1_S</b>	<b>SPI0_S</b>	<b>ADC_S</b>		<b>WDT_S</b>	

Bits	描述	
[31:30]	<b>PWM23_S</b>	<b>PWM3 与 PWM2的时钟源选择.</b> PWM3 与 PWM2使用相同的时钟源 和相同的预分频 00 = 外部高速晶振HXT (4~24MHz) 01 = 内部10K振荡器LIRC(只有M051xxDN/DE) 10 = 时钟源来自HCLK 11 = 内部 22.1184MHz 高速振荡器HIRC
[29:28]	<b>PWM01_S</b>	<b>PWM1 与 PWM0的时钟源选择.</b> PWM1 与 PWM0使用相同的时钟源 和相同的预分频 00 = 外部高速晶振HXT (4~24MHz) 01 = 内部10K振荡器LIRC(只有M051xxDN/DE) 10 = HCLK 11 = 内部22.1184MHz 高速振荡器HIRC
[27:26]	保留	保留
[25:24]	<b>UART_S</b>	<b>UART时钟源选择.</b> 00 =外部高速晶振HXT (4~24MHz) 01 = PLL 10 = 保留 11 = 内部22.118422MHz 振荡器HIRC
[23]	保留	保留

[22:20]	<b>TMR3_S</b>	<b>TIMER3 时钟源选择.</b> 000 = 外部高速晶振HXT (4~24MHz) 010 = HCLK 011 = 时钟来自外部T3引脚(只有M051xxDN/DE) 101 = 时钟来自内部10K振荡器LIRC(只有M051xxDN/DE) 111 = 内部 22.1184MHz 高速振荡器HIRC Others = 保留
[19]	保留	保留
[18:16]	<b>TMR2_S</b>	<b>TIMER2 时钟源选择.</b> 000 = 外部高速晶振HXT (4~24MHz) 010 = HCLK 011 = 时钟来自外部T3引脚(只有M051xxDN/DE) 101 = 时钟来自内部10K振荡器LIRC(只有M051xxDN/DE) 111 = 内部 22.1184MHz 高速振荡器HIRC Others = 保留
[15]	保留	保留
[14:12]	<b>TMR1_S</b>	<b>TIMER1 时钟源选择.</b> 000 = 外部高速晶振HXT (4~24MHz) 010 = HCLK 011 = 时钟来自外部T3引脚(只有M051xxDN/DE) 101 = 时钟来自内部10K振荡器LIRC(只有M051xxDN/DE) 111 = 内部 22.1184MHz 高速振荡器HIRC Others = 保留
[11]	保留	保留
[10:8]	<b>TMR0_S</b>	<b>TIMER0 时钟源选择.</b> 000 = 外部高速晶振HXT (4~24MHz) 010 = HCLK 011 = 时钟来自外部T3引脚(只有M051xxDN/DE) 101 = 时钟来自内部10K振荡器LIRC(只有M051xxDN/DE) 111 = 内部 22.1184MHz 高速振荡器HIRC Others = 保留
[7:6]	保留	保留
[5]	<b>SPI1_S</b>	<b>SPI1 时钟源选择 (只有M051xxDN/DE)</b> 0 = 时钟来自PLL. 1 = 时钟来自HCLK.
[4]	<b>SPI0_S</b>	<b>SPI0 时钟源选择 (只有M051xxDN/DE)</b> 0 = 时钟来自PLL. 1 = 时钟来自HCLK

[3:2]	<b>ADC_S</b>	<p><b>ADC 时钟源选择.</b></p> <p>00 =外部高速晶振HXT (4~24MHz)      01 = PLL      10 = HCLK      11 =内部 22.1184 MHz 振荡器HIRC</p>
[1:0]	<b>WDT_S</b>	<p><b>看门狗 时钟源选择(写保护).</b></p> <p>00 =保留      01 =保留      10 = HCLK/2048      11 = 内部 10KHz低速振荡器LIRC</p> <p><b>注:</b> 受保护位, 对该位编程时, 需要向0x5000_0100 依次写入“59h”, “16h”, “88h” 来解除寄存器写保护, 参考寄存器REGWRPROT, 地址为GCR_BA + 0x100</p>

时钟源选择控制寄存器 (CLKSEL2)

在时钟切换前，相关的时钟源(当前和新选的)，必须都打开.

寄存器	偏移量	R/W	描述				复位后的值
CLKSEL2	CLK_BA + 0x1C	R/W	时钟源选择控制寄存器 2				0x0000_00FF

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

Bits	描述	
[31:18]	保留	保留
[17:16]	WWDT_S	窗看门狗时钟源选择(只有M051xxDN/DE) 10 = 时钟源来自HCLK/2038 11 = 时钟源来自LIRC
[15:8]	保留	保留
[7:6]	PWM67_S	<b>PWM6 与 PWM7的时钟源选择.</b> PWM6 与 PWM7使用相同的时钟源 和相同的预分频器 00 = 外部高速晶振HXT (4~24MHz) 01 = 时钟源来自LIRC(只有M051xxDN/DE) 10 = HCLK 11 = 内部 22.1184MHz 高速振荡器HIRC
[5:4]	PWM45_S	<b>PWM4 与 PWM5的时钟源选择.</b> PWM4 与 PWM5使用相同的时钟源 和相同的预分频器 00 = 外部高速晶振HXT (4~24MHz) 01 = 时钟源来自LIRC(只有M051xxDN/DE) 10 = HCLK 11 = 内部22.1184MHz 高速振荡器HIRC
[3:2]	FRQDIV_S	<b>时钟分频器时钟源选择</b> 00 = 外部高速晶振HXT (4~24MHz) 01 = 时钟源来自LIRC(只有M051xxDN/DE) 10 = HCLK 11 = 内部 22.1184MHz 高速振荡器HIRC
[1:0]	保留	保留

时钟分频寄存器(CLKDIV)

寄存器	偏移量	R/W	描述	复位后的值
CLKDIV	CLK_BA_+ 0x18	R/W	时钟分频寄存器	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:16]	ADC_N	ADC时钟频率=ADC时钟源频率/ (ADC_N + 1)
[15:12]	保留	保留
[11:8]	UART_N	UART时钟频率 = (UART时钟源频率) / (UART_N + 1)
[7:4]	保留	保留
[3:0]	HCLK_N	HCLK 时钟频率 = (HCLK时钟源频率) / (HCLK_N + 1)

**PLL控制寄存器 (PLLCON)**

PLL的参考时钟输入来自外部高速晶振时钟HXT（4~24MHz）输入或内部22.1184MHz高速振荡器HIRC，该寄存器用于控制PLL的输出频率和PLL的操作模式

寄存器	偏移量	R/W	描述	复位后的值
<b>PLLCON</b>	CLK_BA + 0x20	R/W	PLL控制寄存器	0x0005_C22E

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

Bits	描述	
[19]	<b>PLL_SRC</b>	<b>PLL时钟源选择</b> 0 = PLL时钟源为外部高速晶振HXT（4~24MHz） 1 = PLL 时钟源为22.1184 MHz 振荡器HIRC
[18]	<b>OE</b>	<b>PLL OE (FOUT enable)引脚控制</b> 0 = 使能 PLL FOUT 1 = PLL FOUT 固定为低
[17]	<b>BP</b>	<b>PLL 旁路控制</b> 0 = PLL 正常模式 (默认) 1 = PLL 时钟输出与时钟输入相同
[16]	<b>PD</b>	<b>掉电模式.</b> 如果设置PWRCON寄存器的PWR_DOWN_EN位被设为"1"，PLL也会进入掉电模式 0 = PLL正常模式 1 = PLL 掉电模式(默认)
[15:14]	<b>OUT_DV</b>	<b>PLL 输出分频控制</b> 参考下表的公式
[13:9]	<b>IN_DV</b>	<b>PLL输入分频控制</b> 参考下表的公式
[8:0]	<b>FB_DV</b>	<b>PLL反馈分频控制</b> 参考下表的公式



PLL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.  $4\text{MHz} < F_{IN} < 24\text{MHz}$
2.  $800\text{KHz} < \frac{F_{IN}}{2 * NR} < 7.5\text{MHz}$
3.  $100\text{MHz} < F_{CO} = F_{IN} \times \frac{NF}{NR} < 200\text{MHz}$   
 $120\text{MHz} < F_{CO}$  is preferred

符号	说明
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PLL频率设置

PLLCON默认值: 0xC22E

FIN = 12 MHz

NR = (1+2) = 3

NF = (46+2) = 48

NO = 4

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



### 频率分频器控制寄存器(FRQDIV)

寄存器	偏移量	R/W	描述	复位后的值
FRQDIV	CLK_BA+ 0x24	R/W	频率分频器控制寄存器	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		DIVIDER1	DIVIDER_EN	FSEL			

Bits	描述	
[31:6]	保留	保留
[5]	DIVIDER1	频率分频器1使能控制(M051xxDN/DE) 0 = 分频器输出频率由FSEL的值决定 1 = 分频器输出频率与输入频率相同
[4]	DIVIDER_EN	频率分频器使能控制 0 = 禁用频率分频 1 = 使能频率分频
[3:0]	FSEL	分频器输出频率选择位 输出频率的公式是 $F_{out} = F_{in}/2^{(N+1)}$ , $F_{in}$ 为输入时钟频率 $F_{out}$ 为分频器输出时钟频率 N 为FSEL[3:0]的值。

## 6.6 外部总线接口 (EBI)

### 6.6.1 概述

NuMicro™ M05xxBN/DN/DE 系列配备一个外部总线接口 (EBI) , 用来访问外部设备.

为节省外部设备与芯片的连接引脚数, EBI支持地址总线与数据总线复用模式. 且地址锁存使能 (ALE)信号支持区分地址与数据周期.

### 6.6.2 特性

外部总线接口有下列功能:

- 支持外部设备最大64K字节 (8位数据宽度)/128K字节(16位数据宽度)
- 外部总线基本时钟频率可调 (MCLK)
- 支持8位或 16 位数据宽度
- 数据访问时间 (tACC), 地址锁存使能时间(tALE) 和地址保持时间(tAHD) 可调
- 支持地址总线和数据总线复用以节省地址管脚
- 空闲周期可配置用于不同的访问条件: 写命令结束(W2X), 连续读(R2R)
- 读/写操作支持0访问保持时间, 写操作有写缓冲以增强读/写效率(M051xxDN/DE)

## 6.6.3 EBI 框图

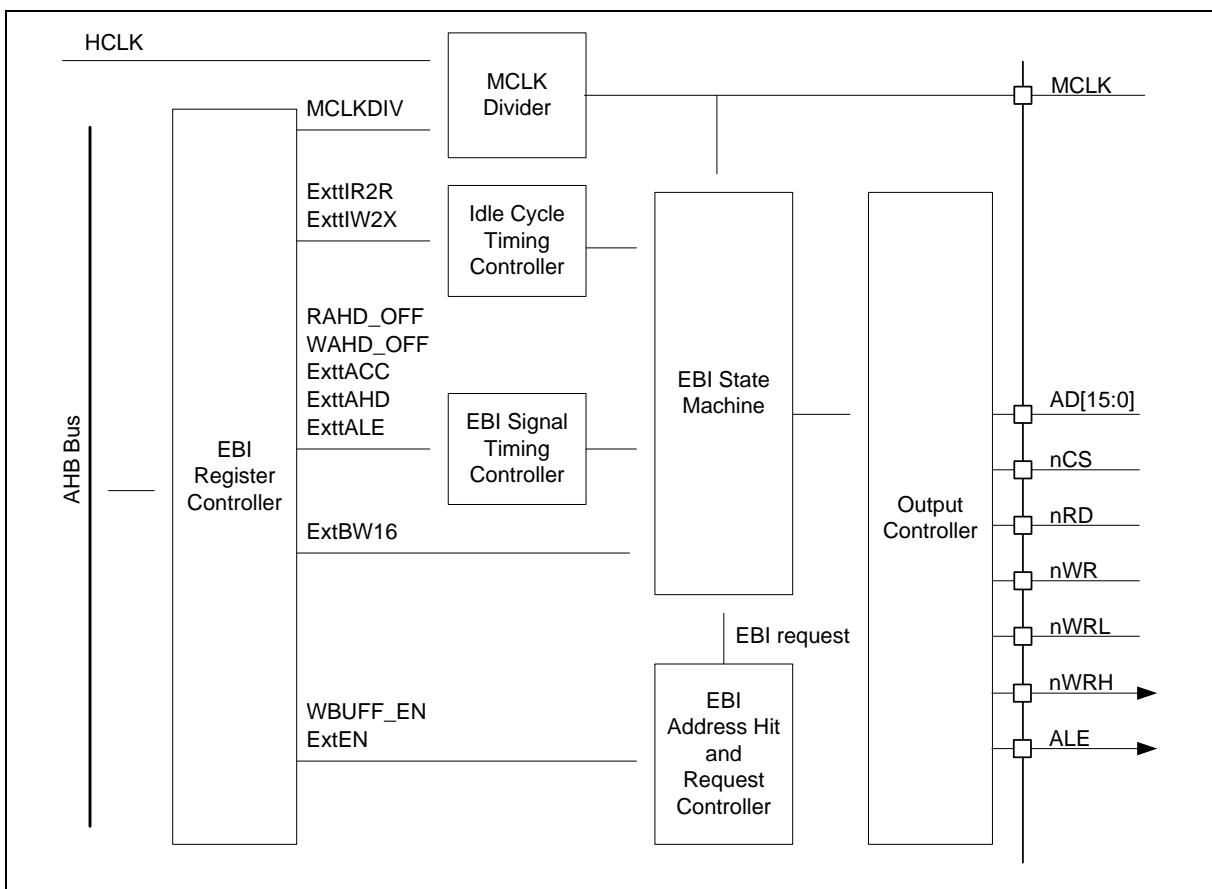


图6-23 EBI 框图

## 6.6.4 基本配置

EBI基本配置如下：

- EBI设备时钟在 AHBCLK[3]中使能
- AD[15:0]引脚功能在 P0\_MFP 和 P2\_MFP 寄存器中配置；nRD、nWR 和 MCLK 引脚功能在 P4\_MFP 寄存器中配置；nWRL 和 nWRH 引脚在 P1\_MFP 寄存器中配置

## 6.6.5 功能描述

### 6.6.5.1 EBI 区域和地址命中

EBI地址映射在0x6000\_0000 ~ 0x6001\_FFFF，最大存储器空间为128K字节。当系统请求的地址命中EBI的存储空间时，相应的EBI片选信号(nCS)有效，EBI状态机开始工作。

对于8位设备(64Kbyte)，EBI把该64K字节的设备同时映射到地址0x6000\_0000 ~ 0x6000\_FFFF 和 0x6001\_0000 ~ 0x6001\_FFFF。

对于16位设备(128Kbyte)，EBI把该128K字节的设备映射到地址0x6000\_0000 ~ 0x6001\_FFFF

### 6.6.5.2 EBI 数据宽度连接

EBI控制器支持地址总线和数据总线复用的设备。对于地址总线与数据总线分开的外部设备，与设备的连接需要额外的逻辑器件来锁存地址。这种情况下，ALE需要连接到锁存设备（如74HC373）以锁存地址。16位数据宽度引脚AD0 ~ AD15，8位数据宽度引脚AD0 ~ AD7作为锁存设备的输入引脚，锁存设备的输出引脚连到外部设备的Addr[15:0]。

对于16位设备，AD[15:0]由地址线与16位数据线共享。对于8位设备，仅AD[7:0]由地址线与8位数据线共享，AD [15:8]则只作为地址线直接与8位设备连接。

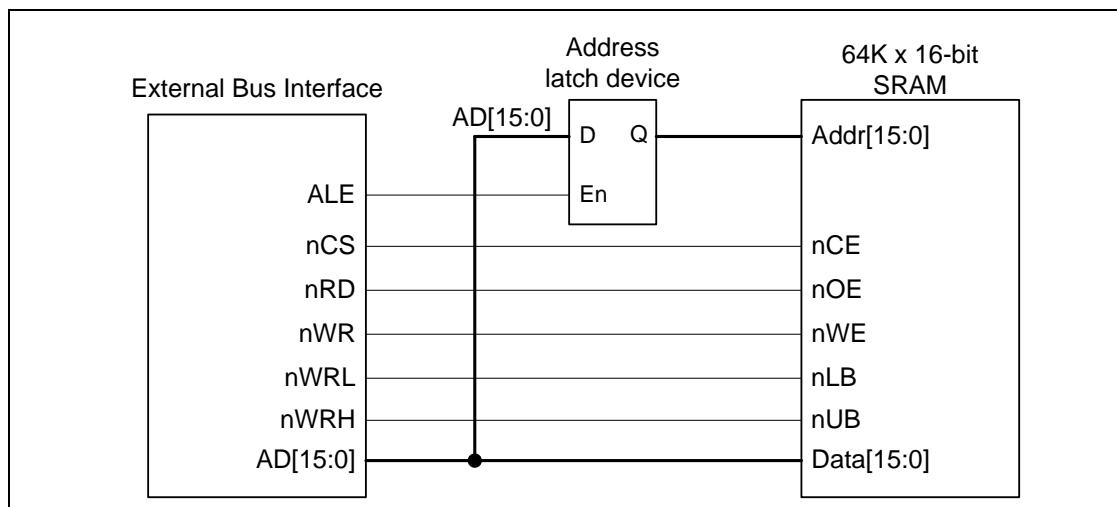


图6-24 16位EBI数据宽度与16位器件连接

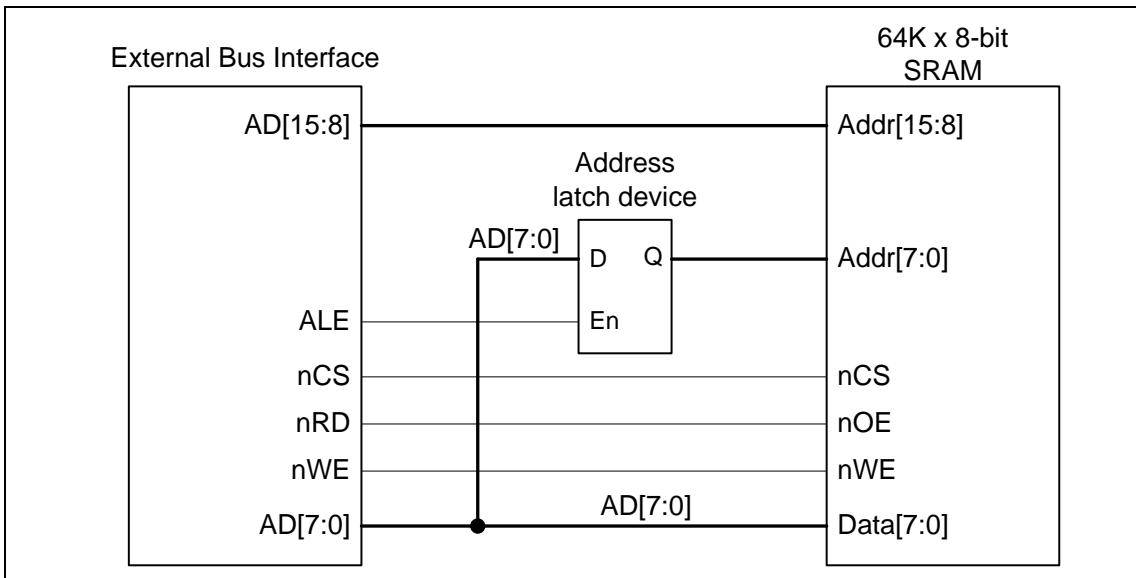


图6-25 8位EBI数据宽度与8位设备连接

当系统访问数据宽度大于EBI的数据宽度时(8-bit / 16 bit 数据宽度), EBI控制器通过多次执行EBI访问来完成操作. 例如, 如果系统通过EBI设备请求32位数据, 如果EBI为8位数据宽度, EBI控制器将访问4次来完成操作.

### 6.6.5.3 EBI 操作控制

#### MCLK 控制

NuMicro™ M051 系列中, EBI工作时, 通过MCLK同步所有EBI信号. 当芯片连接到工作频率较低的外部设备时, MCLK 可以通过设置寄存器EBICON 中的MCLKDIV[2:0]分频, 最小可达HCLK/32. 因此, EBI控制器 可以适用于宽频率范围的EBI 设备。如果 MCLK的频率等于HCLK, EBI 信号由MCLK的上升沿同步, 其他情况下, EBI信号由MCLK的下降沿同步.

#### 操作与访问时序控制

EBI开始访问时, 片选 (nCS)置低并等待一个MCLK时钟周期等待地址建立时间(tASU)以使地址稳定。然后地址稳定后地址锁存使能信号ALE置高 并保持一段时间 (tALE) 以用于地址锁存。地址锁存后, ALE 置低并等待一个MCLK 时钟周期等待地址锁存保持时间(tLHD) 和另一个MCLK 的周期 (tA2D)插在tLHD 后面 用于总线转换 (地址到数据)。然后当读操作时nRD 拉低或写操作时nWR拉低。在访问保持时间tACC(用于读取输出稳定或者写完成)之后nRD和nWR被拉高。之后, EBI 信号保持数据访问保持时间 (tAHD), 然后片选信号置高, 地址由当前访问控制释放。

EBI控制器提供如下表所示的灵活的EBI 时序控制以用于不同的外部设备. 在EBI 的时序控制中, tASU, tLHD 和 tA2D固定为1个MCLK周期, tAHD 可以通过设置寄存器EXTIME的ExttAHD[2:0]在1~8 个MCLK 周期内调节, M05xxDN/DE中通过设定RAHD\_OFF/WAHD\_OFF位, 读/写时tAHD可以为0个MCLK周期; tACC可以通过设置寄存器EXTIME的ExttACC[4:0]在1~32 个MCLK周期内调节; tALE可以通过寄存器EBICON的ExttALE[2:0]在1~8 个MCLK 周期内调节.

参数	值	单位	描述
----	---	----	----

tASU	1	MCLK	地址锁存建立时间.
tALE	1 ~ 8	MCLK	ALE 高电平时间. 由EBICON的ExttALE控制.
tLHD	1	MCLK	地址锁存保持时间.
tA2D	1	MCLK	地址到数据的延迟 (总线转换时间).
tACC	1 ~ 32	MCLK	数据访问时间. 由EXTIME的ExttACC[4:0]控制.
tAHD	1 ~ 8	MCLK	数据访问保持时间. 由EXTIME 的ExttAHD[2:0] 控制.
tAHD	0	MCLK	设地 RAHD_OFF/WAHD_OFF 位选择0个 tAHD (只有 M05xxDN/DE 支持).
IDLE	0 ~ 15	MCLK	空闲周期.由EXTIME 的ExtlR2R 和 ExtlW2X 控制.

表6-7 EBI 时序表

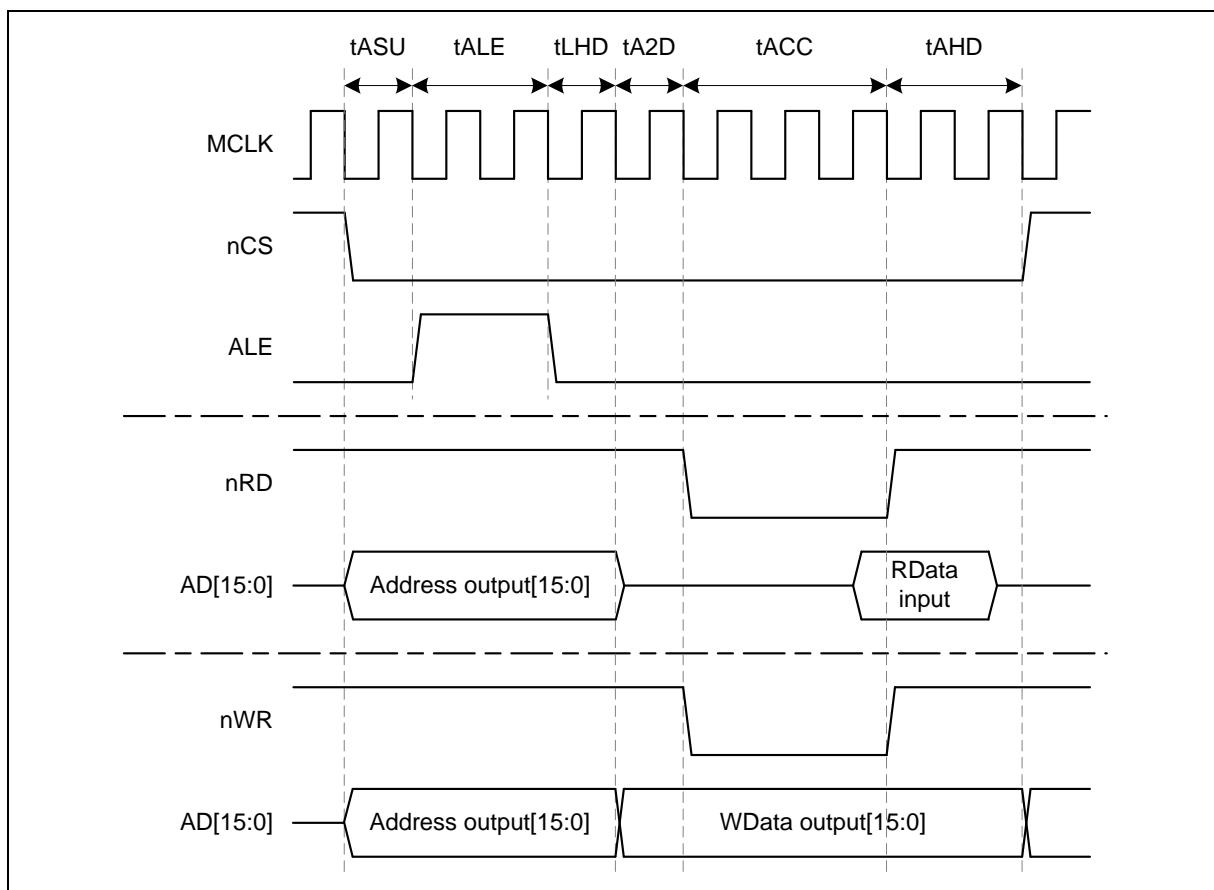


图6-26 16位数据宽度的EBI时序控制波形

上图的时序波形是以16位数据宽度为例. 此例中, AD0 ~ AD15 用作地址 [15:0] 和数据 [15:0]. 当 ALE 拉高时, AD0 ~ AD15为地址输出. 在地址锁存后(tLHD), ALE 拉低, AD0 ~AD15 总线转换成高阻以等待设备输出数据 (在读取访问操作时), 或用于写数据输出.

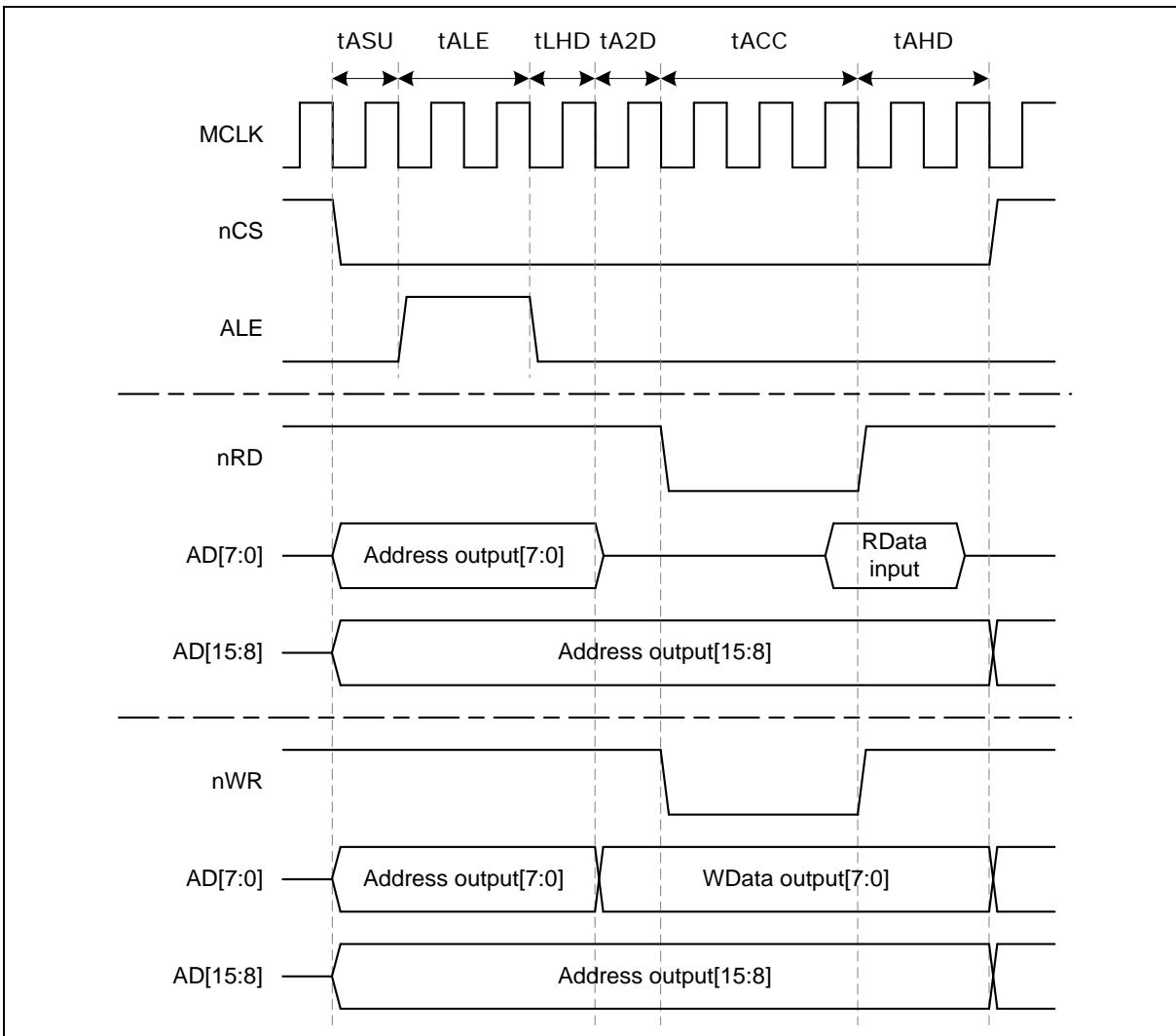


图6-27 8位数据宽度EBI时序控制波形

上图的时序波形是以8位数据宽度为例。与16位数据宽度不同的是AD[15:8]的使用。在8位数据宽度时，AD[15:8]固定为地址位 [15:8] 的输出，所以外部锁存仅需要8位宽度。

### 插入空闲周期

当EBI连续访问时，如果设备访问速度远低于系统工作速度，可能会发生总线冲突。EBI控制器 支持额外的空闲周期以解决该问题。在空闲周期，EBI的所有控制信号无效。下图为空闲周期波形图：

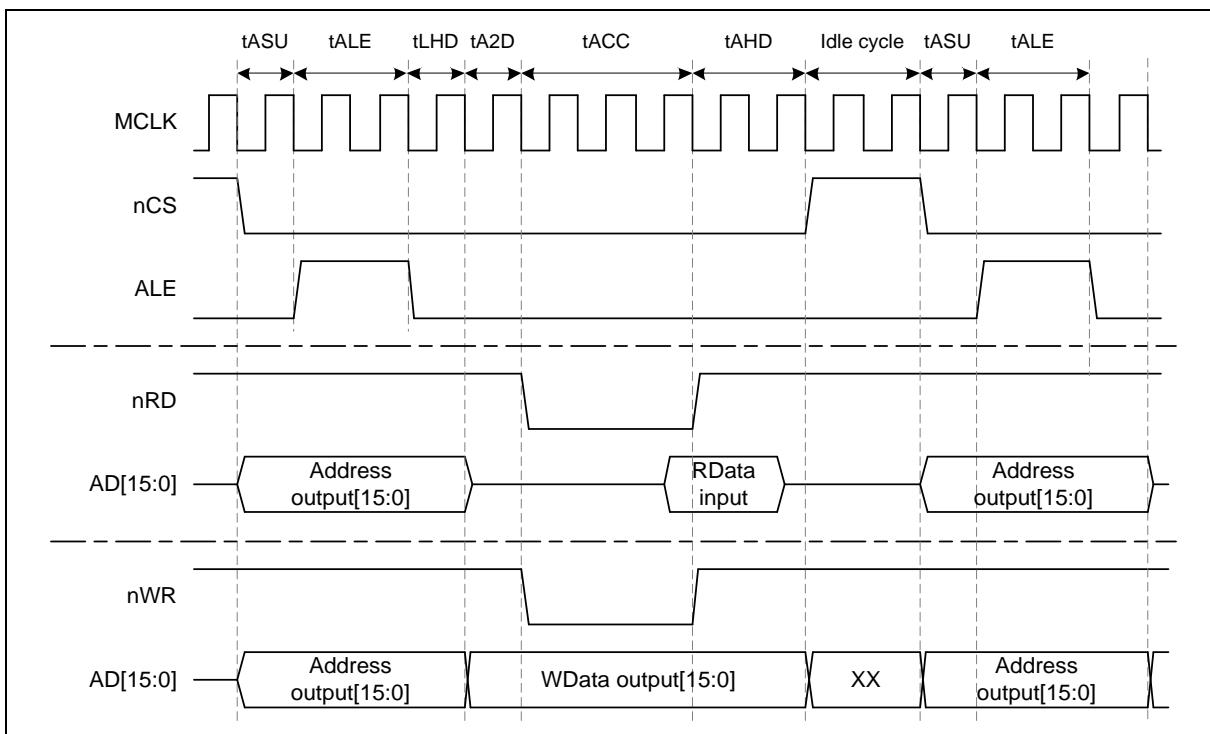


图6-28 插入空闲周期的EBI时序控制波形

在下面两种情况下，EBI可插入空闲周期：

- 写访问之后
- 读访问之后与下一个读访问之前

通过设置寄存器EXTIME的ExtIW2X[3:0], ExtIR2R[3:0]，空闲周期可设定为0~15 MCLK.

#### 写缓冲 (只有M05xxDN/DE)

当软件通过EBI总线写数据到外部设备的时候，EBI控制器将立即开始写动作，CPU一直等待EBI写操作完成。用户可以使能写缓冲功能提升CPU和EBI访问的性能。当EBI写缓冲功能使能时，EBI控制器处理写操作时，CPU可以继续执行其他指令。这种情况下，有一个异常条件，如果EBI正在执行写操作，此时CPU通过EBI又执行其他数据访问，CPU仍将等待。



### 6.6.6 寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>EBI基地址:</b>				
<b>EBI_BA = 0x5001_0000</b>				
<b>EBICON</b>	EBI_BA+0x00	R/W	外部总线接口通用控制寄存器	0x0000_0000
<b>EXTIME</b>	EBI_BA+0x04	R/W	外部总线接口时序控制寄存器	0x0000_0000
<b>EBICON2</b>	EBI_BA+0x08	R/W	外部总线接口通用控制寄存器2(只有M051xxDN/DE)	0x0000_0000

### 6.6.7 寄存器描述

#### 外部总线接口控制寄存器(EBICON)

寄存器	偏移量	R/W	描述	复位后的值
EBICON	EBI_BA+0x00	R/W	外部总线接口控制寄存器	0x0000_0000

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

Bits	描述																	
[31:19]	保留	保留																
[18:16]	ExttALE	<p><b>ALE的延长时间</b>            该域用来控制地址锁存ALE 脉冲宽度 (tALE)  <math>tALE = (ExttALE+1)*MCLK</math></p>																
[15:11]	保留	保留																
[10:8]	MCLKDIV	<p><b>外部输出时钟分频器</b>            由 MCLKDIV 控制 EBI 输出时钟的频率，见下表:</p> <table border="1"> <thead> <tr> <th>MCLKDIV</th> <th>输出频率 (MCLK)</th> </tr> </thead> <tbody> <tr> <td>000</td> <td>HCLK/1</td> </tr> <tr> <td>001</td> <td>HCLK/2</td> </tr> <tr> <td>010</td> <td>HCLK/4</td> </tr> <tr> <td>011</td> <td>HCLK/8</td> </tr> <tr> <td>100</td> <td>HCLK/16</td> </tr> <tr> <td>101</td> <td>HCLK/32</td> </tr> <tr> <td>11</td> <td>保留</td> </tr> </tbody> </table> <p>注: 默认输出时钟为HCLK/1</p>	MCLKDIV	输出频率 (MCLK)	000	HCLK/1	001	HCLK/2	010	HCLK/4	011	HCLK/8	100	HCLK/16	101	HCLK/32	11	保留
MCLKDIV	输出频率 (MCLK)																	
000	HCLK/1																	
001	HCLK/2																	
010	HCLK/4																	
011	HCLK/8																	
100	HCLK/16																	
101	HCLK/32																	
11	保留																	
[7:2]	保留	保留																
[1]	ExtBW16	<p><b>EBI 数据宽度为 16位/8 位</b>            该位配置数据总线是8位还是16位宽度.</p>																



		0 = EBI 数据宽度为8位 1 = EBI 数据宽度为16位
[0]	<b>ExtEN</b>	<b>EBI 使能控制</b> 该位使能EBI功能. 0 = 禁用EBI功能 1 = 使能EBI功能

外部总线接口时序控制寄存器(EXTIME)

寄存器	偏移量	R/W	描述	复位后的值
EXTIME	EBI_BA+0x04	R/W	外部总线接口时序控制寄存器	0x0000_0000

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

Bits	描述	
[31:28]	保留	保留
[27:24]	ExtIR2R	<p>读与读之间的空闲状态周期 ExtIR2R不等于0的时候，如果读操作完成且下一个动作也是读时，插入空闲状态周期且nCS信号拉高。 空闲状态周期 = (ExtIR2R * MCLK)</p>
[23:16]	保留	保留
[15:12]	ExtIW2X	<p>写之后的空闲状态周期 ExtIW2X不等于0的时候，写完成之后，将插入空闲状态且nCS拉高。 空闲状态周期 = (ExtIW2X * MCLK)</p>
[11]	保留	保留
[10:8]	ExttAHD	<p>EBI 数据访问保持时间 ExttAHD 定义数据访问保持时间(tAHD). <math>tAHD = (ExttAHD + 1) * MCLK</math></p>
[7:3]	ExttACC	<p>EBI 数据访问时间 ExttACC 定义数据访问时间 (tACC). <math>tACC = (ExttACC + 1) * MCLK</math></p>
[2:0]	保留	保留

外部总线接口控制寄存器2(EBICON2)

寄存器	偏移量	R/W	描述	复位后的值
EBICON2	EBI_BA+0x08	R/W	外部总线接口控制寄存器2(只有M05xxDN/DE)	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					WAHD_OFF	RAHD_OFF	WBUFF_EN

Bits	描述	
[31:3]	保留	保留
[2]	WAHD_OFF	写操作时访问保持时间关闭控制 0 = 通过EBI的写操作, tAHD由ExttAHD[2:0]控制 1 = 通过EBI的写操作, tAHD时间为0
[1]	RAHD_OFF	读操作时访问保持时间关闭控制 0 = 通过EBI的读操作, tAHD由ExttAHD[2:0]控制 1 = 通过EBI的读操作, tAHD时间为0
[0]	WBUFF_EN	EBI 写缓冲使能控制 使能该功能提升CPU和EBI的访问效率 0 = 禁用EBI写缓冲功能 1 = 使能EBI写缓冲功能

## 6.7 Flash内存控制器(FMC)

### 6.7.1 概述

M05xxBN 和 M05xxDN/DE 系列具有 64K/32K/16K/8K 字节的片上FLASH (APROM)，用于存储应用程，可以通过ISP寄存器更新。在系统编程 (ISP) 和在应用编程(IAP)允许用户更新焊接在PCB板上的芯片中的程序。**上电后，通过设置CONFIG0的启动选择位 (CBS) 决定 Cortex-M0 CPU 从 APROM 还是 LDROM 读取代码。**此外，M05xxBN和M05xxDN/DE为用户提供额外的4k字节的数据FLASH，以供用户在系统掉电之前存储数据。

M05xxDN/DE 在 CONFIG0 中提供更多设置，以支持高级功能，包括上电时 I/O 的状态，启动时使能 WDT，睡眠时使能 WDT 和 IAP 功能。下表列出 M05xxDN/DE 新添加的功能

	M05xxBN	M05xxDN/DE
CONFIG[6]	保留	支持IAP 功能和多引导选项
CONFIG[10]	保留	选择启动时 I/O 口的状态
CONFIG[30]	保留	当WDT默认启动使能时，支持睡眠模式下关闭WDT 时钟，还是WDT 时钟不由软件控制
CONFIG[31]	保留	支持启动时使能WDT，一旦使能如果 CONFIG[30]=0，软件无法停止WDT

表 6-8 M05xxBN 和 M05xxDN/DE FMC 功能比较表

### 6.7.2 特性

- 高达 50MHz 的零等待连续地址读访问
- 64/32/16/8KB 应用程序存储空间(APROM)
- 4KB 在系统编程 (ISP) 空间(LDROM)
- 固定的 4kB 数据FLASH
- 所有内部 Flash 页擦除单位为 512 字节
- 在系统编程(ISP)/在应用编程(IAP) 用于更新片上 Flash EPROM

### 6.7.3 FMC 框图

FLASH存储器控制器由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组成。FLASH存储器控制器框图如下所示：

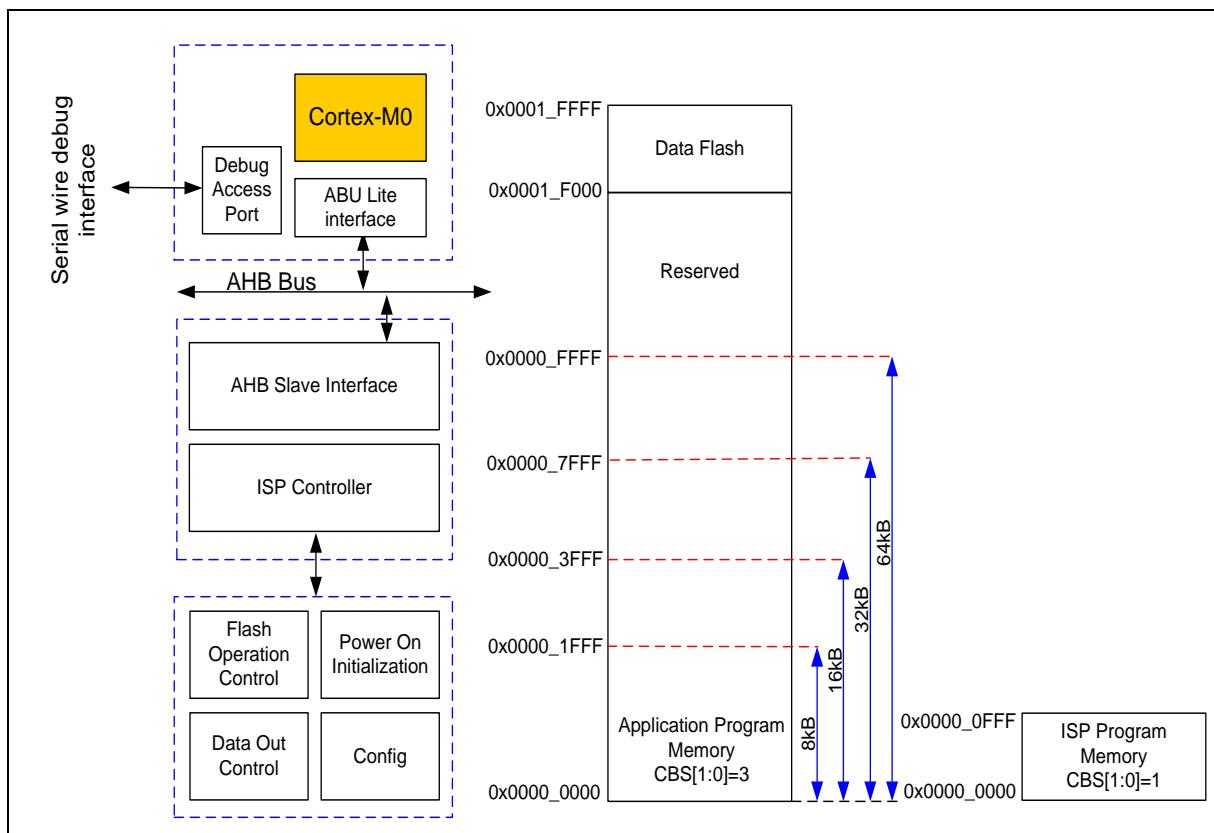


图 6-29 Flash 存储器控制器框图

#### 6.7.4 FMC组织结构

M05xxBN/DN/DE 的flash存储器由程序存储器(APROM)，数据FLASH，ISP加载程序存储器，和用户配置区共4块组成。

用户程序主要放在APROM中，用户将程序烧入APROM之后，设定系统从APROM启动就可以执行APROM中的程序了。

ISP启动程序空间LDROM用来实现在系统编程。LDROM空间和APROM空间是独立的，系统可以选择从LDROM启动或者从APROM启动。用户可以将ISP程序烧入LDROM中，然后选择从LDROM启动，这样LDROM中的程序就可以检查APROM中的程序是否被破坏，或者用户可以决定APROM中的程序是否需要更新。

数据Flash用来存储数据。可以使用ISP命令读或者直接读取，但是写操作一定要通过ISP命令。数据Flash的大小为4KB，每次擦除为512字节，基地址为0x0001\_F000。

用户配置区提供几个字节来控制系统逻辑，如flash安全加密，启动选择，欠压电平等。用户配置块的作用类似保险丝用于上电时的缺省配置，在上电期间，从FLASH存储器被加载到相应的控制寄存器中。

NuMicro™家族中，Flash内存映射和系统内存映射是不同的。Flash内存映射是当用户使用ISP命令读、写和擦除的时候使用。系统内存映射是CPU访问flash的时候使用，就是说系统地址是给CPU看的，Flash地址是给FMC看的。例如：CPU读代码或者数据的时候。如果CBS[1:0] = 1，系统选择从LDROM启动，CPU可以从系统地址0x0~ 0xFFFF(LDROM)读取代码，但是如果用户想使用ISP命令从LDROM读取数据的时候，地址应该为0x0010\_0000 ~ 0x0010\_0FFF。

下表和图显示了APROM、LDROM、数据Flash和用户配置区Flash内存映射地址

区块名称	大小	开始地址	结束地址
AP-ROM	8/16/32/64KB	0x0000_0000	0x0000_1FFF (8KB) 0x0000_3FFF (16KB) 0x0000_7FFF (32KB) 0x0000_FFFF (64KB)
Data Flash	4KB	0x0001_F000	0x0001_FFFF
LD-ROM	4KB	0x0010_0000	0x0010_0FFF
用户配置区	1 Words	0x0030_0000	0x0030_0000

表6-9 Flash存储器地址映射

Flash存储器组织结构如下所示:

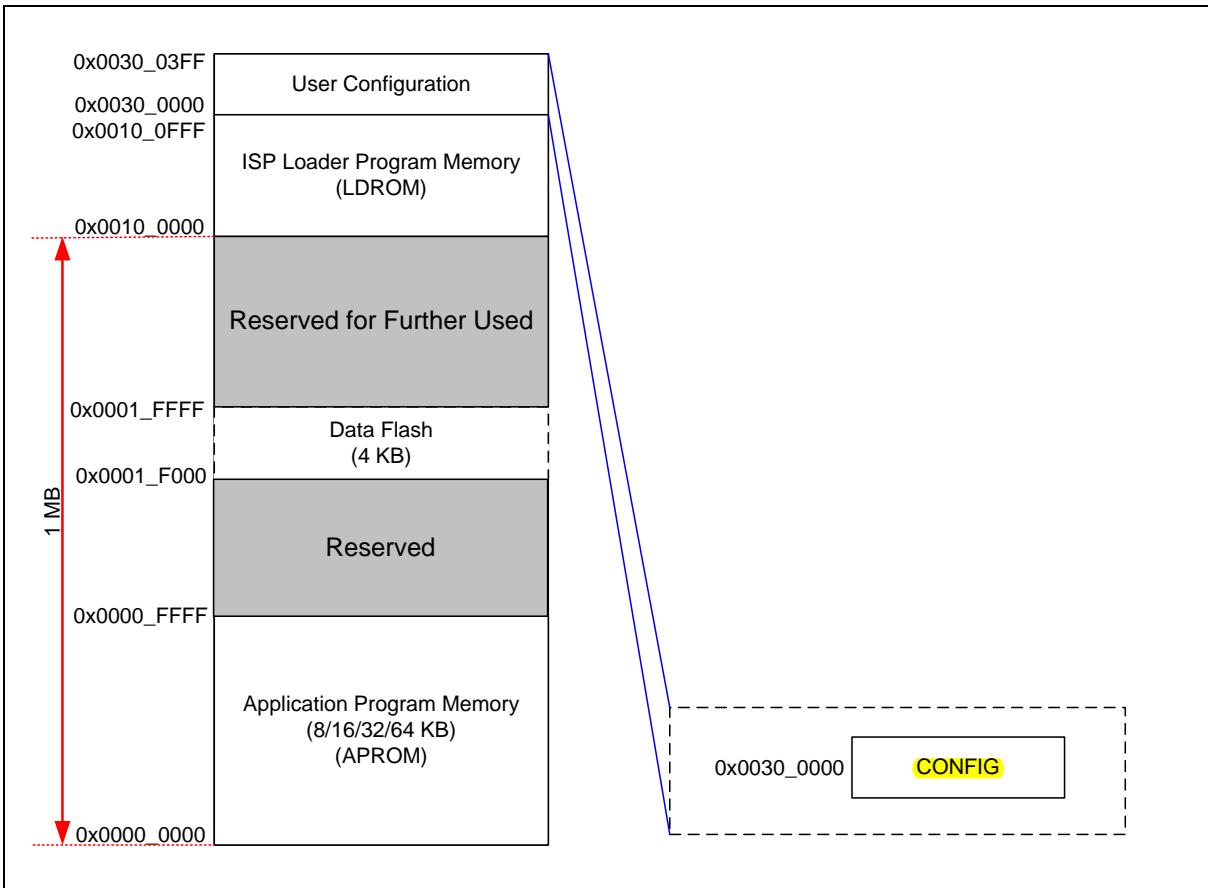


图 6-30 Flash 存储器组织结构

### 6.7.5 Data Flash

M05xxBN/DN/DE为用户提供数据FLASH用于存储数据。通过ISP寄存器读/写. 擦除单位为512字节. 若要改变一个字, 需要把所有128个字拷贝到另外页或SRAM中. 对于8/16/32/64KB的flash设备, 数据FLASH的大小为4KB, 开始地址固定在0x0001\_F000.

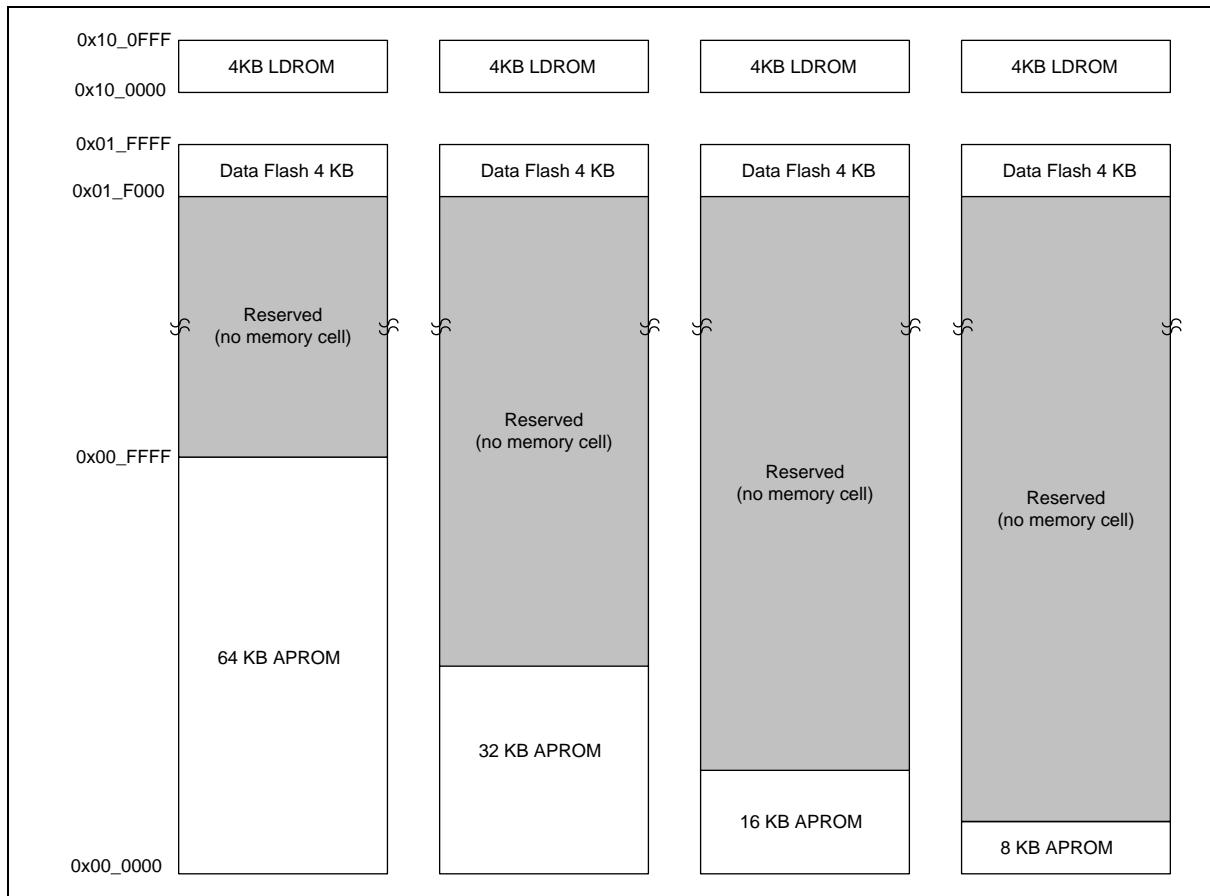


图 6-31 Flash 存储器结构

## 6.7.6 用户配置区

### 6.7.6.1 启动选择

用户配置区是内部可编程的配置区用于启动选择。用户配置区在Flash内存的地址0x300000，总共2个字。它的内容用于在系统启动时配置一些外设的寄存器。

**CONFIG (地址= 0x0030\_0000)**

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

Bits	描述									
[31]	<b>CWDTEN</b>	看门狗使能控制 0 = 芯片上电后使能看门狗定时器，并且迫使时钟源为内部10K振荡器。 1 = 上电时看门狗默认关闭.								
[30]	<b>CWDTPDEN</b>	看门狗时钟睡眠使能控制 0 = 看门狗的10K时钟源迫使总是使能的，软件无法关闭. 1 = 当芯片进入睡眠模式，执行WFI指令之前，看门狗的10K时钟源由OSC10K_EN (PWRCON[3])控制 注：该位只有在CWDTEN = 0时才工作								
[29:27]	保留	保留								
[26:24]	<b>CFOSC</b>	复位后CPU 时钟源选择 <table border="1"> <tr> <td>CFOSC[2:0]</td> <td>时钟源</td> </tr> <tr> <td>000</td> <td>外部晶振时钟 (4 ~ 24MHz)</td> </tr> <tr> <td>111</td> <td>内部 RC 22.1184 MHz 振荡器时钟</td> </tr> <tr> <td>其他</td> <td>保</td> </tr> </table> 复位发生后， CFOSC 的值将被加载到CLKSEL0.HCLK_S[2:0].	CFOSC[2:0]	时钟源	000	外部晶振时钟 (4 ~ 24MHz)	111	内部 RC 22.1184 MHz 振荡器时钟	其他	保
CFOSC[2:0]	时钟源									
000	外部晶振时钟 (4 ~ 24MHz)									
111	内部 RC 22.1184 MHz 振荡器时钟									
其他	保									
[23]	<b>CBODEN</b>	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]	<b>CBOV</b>	欠压电压选择 <table border="1"> <tr> <td>CBOV</td> <td>欠压电压</td> </tr> </table>	CBOV	欠压电压						
CBOV	欠压电压									

		<table border="1"> <tr><td>11</td><td>4.5V</td></tr> <tr><td>10</td><td>3.7V</td></tr> <tr><td>01</td><td>2.7V</td></tr> <tr><td>00</td><td>2.2V</td></tr> </table>	11	4.5V	10	3.7V	01	2.7V	00	2.2V	
11	4.5V										
10	3.7V										
01	2.7V										
00	2.2V										
[20]	<b>CBORST</b>	<b>欠压复位使能</b> 0 = 上电后使能欠压复位 1 = 上电后禁用欠压复位									
[19:11]	保留	保留									
[10]	<b>CIOINI</b>	<b>I/O 口初始状态选择</b> 0 = 上电后所有 GPIO 默认为输入三态模式. 1 = 上电后所有GPIO 默认为准双向.									
[9:8]	保留	保留									
[7:6]	<b>CBS</b>	<b>芯片启动选择</b> 00 = 芯片从LDROM启动, IAP功能使能 01 =芯片从LDROM启动, 没有IAP功能 10 = 芯片从APROM启动, IAP功能使能 11 =芯片从APROM启动, 没有IAP功能  M05xxBN 只支持 CBS[0] = 1, 意味着 M05xxBN 没有 IAP 功能。对于 M05xxDN/DE, 用户可以设定 CBS[0] = 0 以支持 IAP 功能。 <b>当 CBS[0] = 0 时, LDROM 映射到系统地址 0x100000 , APROM 映射到系统地址 0x0</b> 。不需要重新启动, CPU 可以访问 LDROM 和 APROM 的所有空间, 换句话说, 如果 IAP 功能使能, LDROM 和 APROM 中的函数可以互相调用。 <b>注1:</b> ISPCON 的 BS 位只有在 CBS[0] = 1 时才能用来控制启动切换 <b>注2:</b> 当CBS[0] = 0 是, VECMAP 命令只可以将某个地址映射到0x0~0x1ff									
[5:2]	保留	保留									
[1]	<b>LOCK</b>	<b>安全锁</b> 0 = Flash 数据锁定 1 = Flash 数据不锁定.  当锁定了flash数据时, 仅有设备ID、唯一ID和用户配置区 可以通过烧录器和ICP通过串行调试接口读出。其他数据锁定在0xFFFFFFFF。ISP 可以不管LOCK是否锁定都能读出数据。 为了解锁系统, 用户可以使用ISP命令关闭LOCK位或者通过ICP工具擦除整个芯片 (芯片擦除)									
[0]	保留	保留									

注: 用户区保留位应该维持为'1'

### 6.7.6.2 欠压检测

M05xxBN 和 M05xxDN/DE 都有欠压检测功能用于监控 $V_{DD}$ 引脚的电压。如果 $V_{DD}$ 电压低于 CBOV 的设定值，如果BOD功能使能，BOD 事件将被触发。用户可以通过使能CBORST决定欠压时BOD复位系统还是发生BOD中断。因为不论何时只要 $V_{DD}$ 电压低于CBOV 设定，BOD复位将马上发生，所以用户必须保证CBOV设定正确，以避免BOD一直产生复位。例如：如果 $V_{DD}$ 电压是3.3V，CBOV 不能为11'b 或者 10'b，否则，如果BOD复位使能，系统将被停在BOD复位状态。

### 6.7.7 启动选择

M05xxBN/DN/DE提供在系统编程 (ISP) 特性，允许用户直接更新PCB板上芯片中的程序。提供4kB 程序存储区专门用于存储ISP固件。用户设置CONFIG0的(CBS)以选择从APROM启动还是LDROM启动。

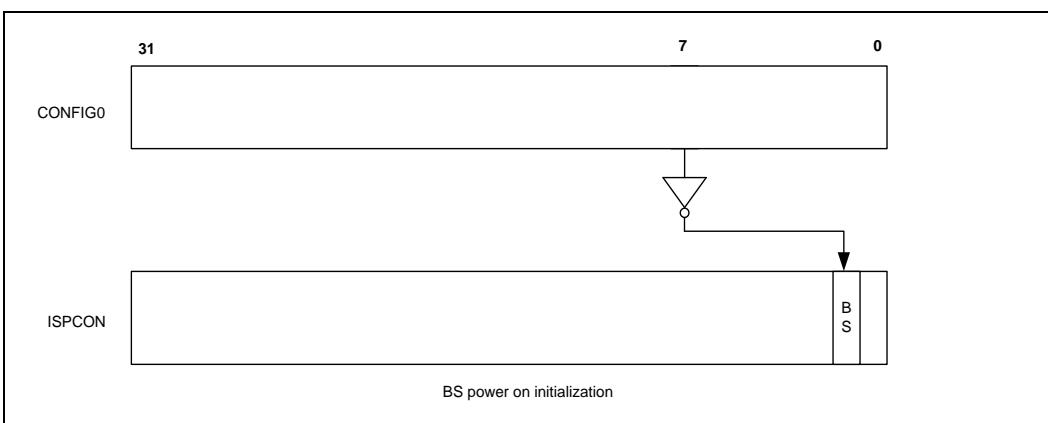


图6-32 上电时启动选择(BS)从CONFIG0加载

除了设定从APROM还是LDROM启动，CONFIG0的CBS还用来控制启动后系统内存映射I。当 CBS[0] = 1，并且设定 CBS[1] = 1 从APROM启动，APROM中的应用程序不能通过CPU读直接访问 LDROM。同样，当CBS[0] = 1 and CBS[1] = 0 系统从LDROM启动，LDROM中的程序也不能通过 CPU读访问APROM中的程序。下图显示了APROM启动和LDROM启动时的系统内存映射。

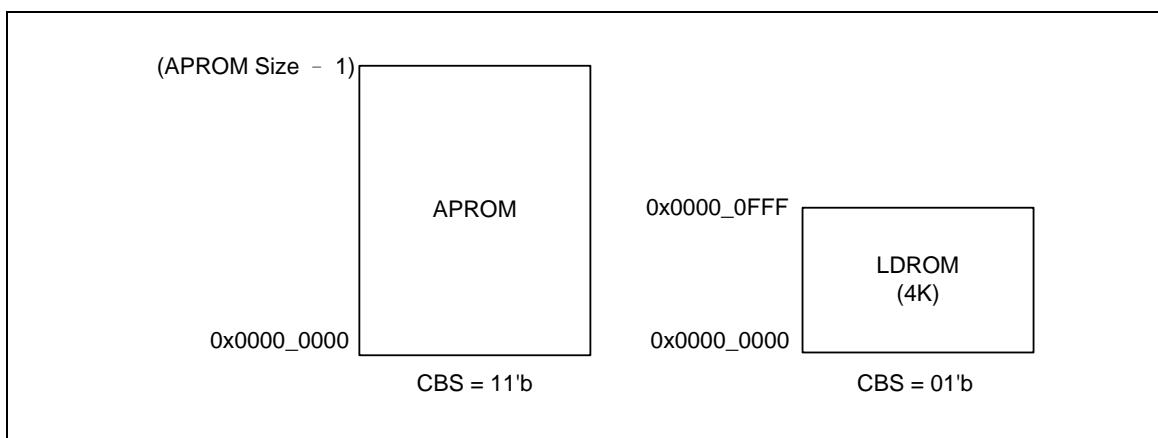


图 6-33 从 APROM 和 LDROM 启动时程序的执行范围

如果应用程序在APROM中执行需要调用LDROM中的函数或者程序在LDROM中执行需要调用APROM中的函数, CBS[0] 需要设为 0 , 这被称为在应用中编程(IAP)。只有M05xxDN/DE 支持IAP 功能。

CBS[1:0]	启动选择
00	<b>LDROM 启动支持IAP 功能 (只有M05xxDN/DE)</b> 芯片从LDROM启动, 程序执行范围包括LDROM 和大部分 APROM (除了第一页512个字节被LDROM映射). LDROM 地址范围0x0010_0000 ~ 0x0010_0FFF, 同时第一页512字节被映射到0x0000_0000 ~ 0x0000_01FF. 通过ISP命令, 地址0x0000_0000 ~ 0x0000_01FF 可以被重新映射到可执行范围的任何地方.
01	<b>LDROM 启动不支持 IAP 功能</b> 芯片从LDROM启动, 程序执行范围为 LDROM, APROM 中的程序不能直接访问必须通过ISP命令. 这种模式下, LDROM 是写保护的.
10	<b>APROM 启动支持IAP功能(只有M05xxDN/DE)</b> 芯片从APROM启动, 程序执行范围包括LDROM 和 APROM LDROM地址范围0x0010_0000~0x0010_0FFF 通过ISP命令, 地址0x0000_0000 ~ 0x0000_01FF可以被重新映射到可执行范围的任何地方.
11	<b>APROM 启动不支持IAP功能</b> 芯片从APROM 启动, 程序执行范围为 APROM, LDROM中的程序不能直接访问必须通过ISP命令. 这种模式下, APROM是写保护的.

表 6-10 启动选项

### 6.7.8 在应用中编程 (IAP)

M05xxDN/DE 提供在应用中编程(IAP)功能, 运行代码可以在APROM和LDROM之间切换而不用复位系统。用户通过设定CONFIG0中的芯片启动选项 (CBS[1:0]) 为10'b 或者 00'b, 然后重新复位芯片可以使能IAP功能。

芯片从APROM 启动, 支持 IAP 功能时(CBS[1:0] = 2), 执行范围包含所有的APROM 和 LDROM. APROM 系统地址空间和 LDROM 系统地址空间与 Flash 映射空间一致, 例如LDROM都在0x0010\_0000~ 0x0010\_0FFF.

芯片从 LDROM启动, 支持IAP功能时(CBS[1:0] = 0), 执行范围包含所有的LDROM和大部分 APROM(除了第一页)。用户不能访问APROM的第一页, 因为默认情况下, 可执行范围的第一页映射为LDROM的第一页了。同时LDROM的4KB地址空间仍映射在0x0010\_0000~0x0010\_0FFF。就是说此时, LDROM的第一页映射了两个地址: 地址0x0和0x100000

IAP地址映射请参考下图.

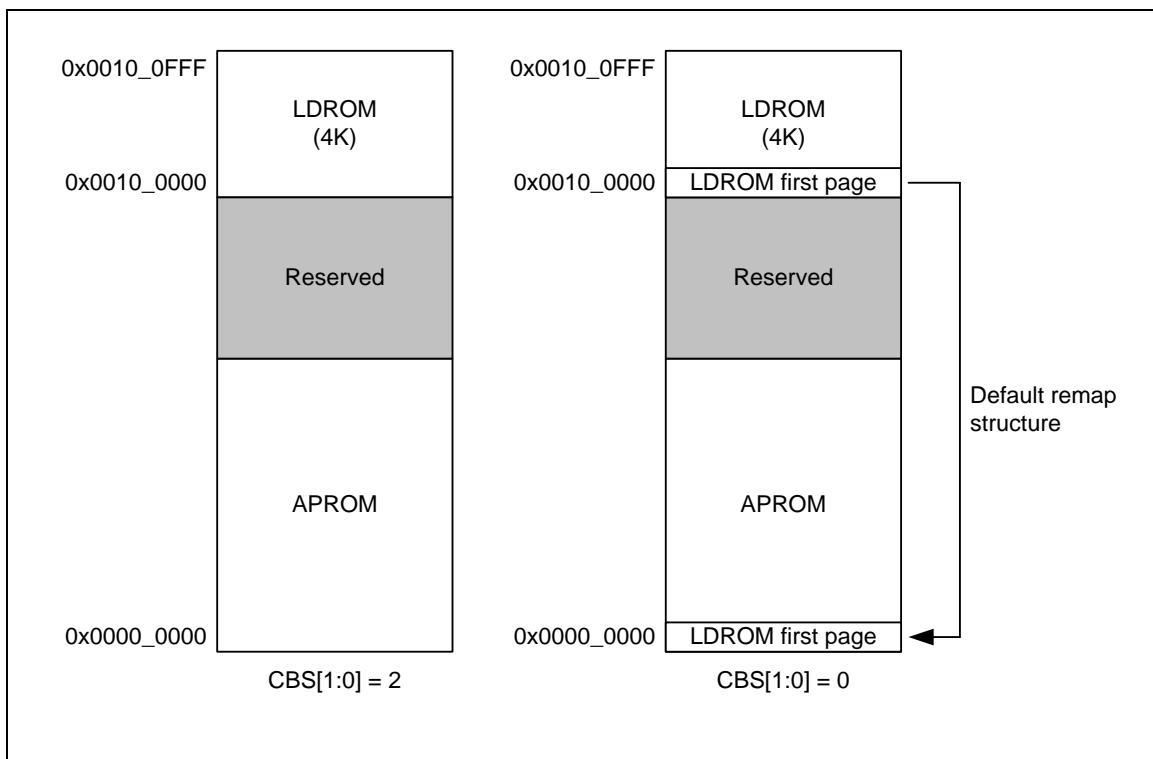


图 6-34 IAP 功能使能时可执行地址范围

当 IAP 功能使能时，可执行范围中的任何页都可以映射到第一页 (0x0000\_0000~0x0000\_01FF). 用户可以通过 ISP 向量页重新映射命令并将要映射的地址填到 ISPADR 寄存器就可以改变第一页的映射地址。通过读 ISPSTA 寄存器的 VECMAP 域，用户可以检查映射是否成功。

### 6.7.9 在系统编程(ISP)

M05xxBN/DN/DE 支持在系统编程，允许设备在软件的控制下重新升级程序。

为了支持在系统编程，M05xxBN/DN/DE 包含了 LDROM 和 ISP 控制器。用户可以实现他们自己的 ISP 加载固件放在 LDROM 中，该加载程序通过 ISP 命令可以编程用户程序(APROM)。换句话说，就是固件提供更新用户程序的功能。ISP 固件使用各种硬件外设接口可以很容易的接收到新的程序代码。ISP 最常用的方式是通过 UART。一般来说，PC 通过串口传输新的 APROM 程序，ISP 固件收到程序以后将其更新到 APROM 中。

### 6.7.10 ISP 寄存器控制过程

M05xxBN/DN/DE 从 APROM 还是 LDROM 启动由用户配置区(CBS)定义。**用户配置区改变之后需要重新复位系统才能生效**。如果用户不想改变用户配置区就能在 APROM 和 LDROM 之间切换，这就需要修改 ISPCON 寄存器的 BS 位，然后通过 AIRCR 寄存器或者 IPRSTC1 寄存器复位系统。修改 BS 位切换启动选项如下图所示。通过 BS 位切换只有在 CBS[0] = 1 时才有效。

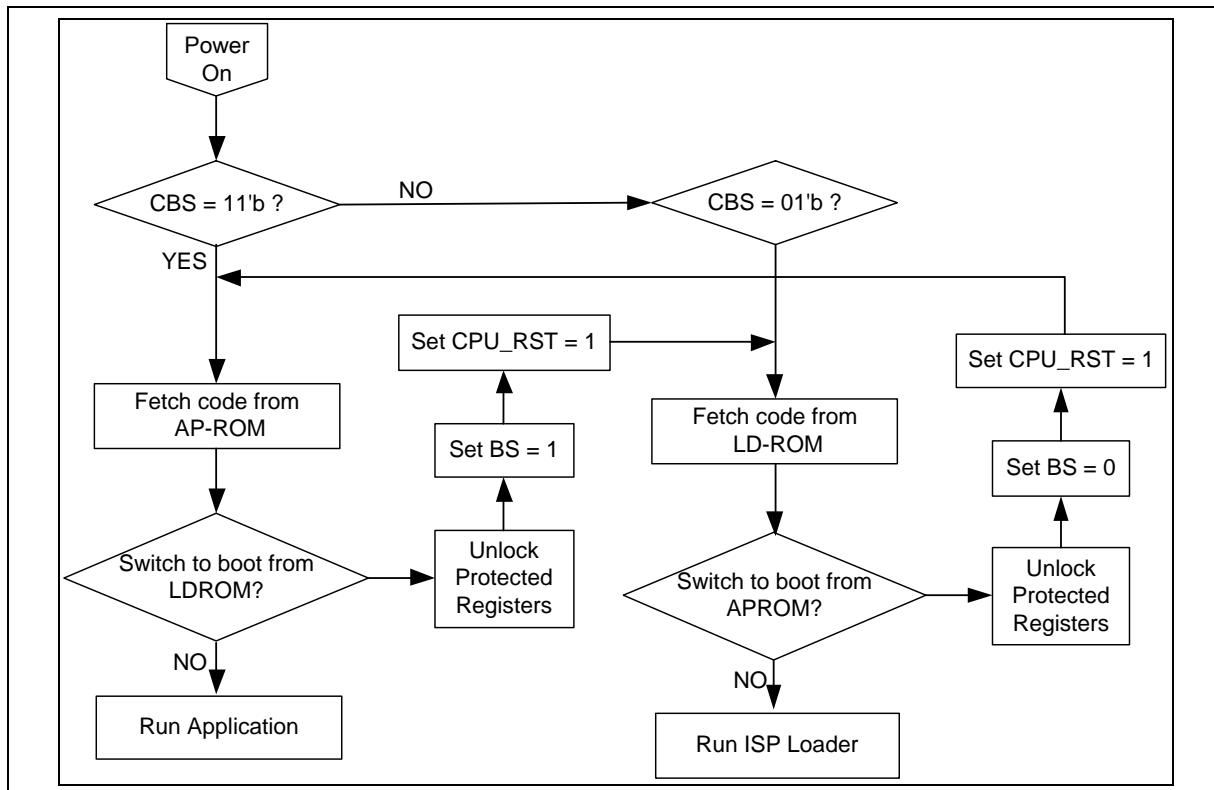


图6-35 CBS[0] = 1时通过BS位改变启动选项

ISP控制器支持读、写、擦除内嵌的Flash。ISP可采取的几个控制位是写保护的，因而在修改它们之前，软件需要向REGWRPROT寄存器（地址 0x5000\_0100）依次写入0x59, 0x16 和 0x88。如果解锁成功，REGWRPROT的值将为1，解锁序列一定不能被打断，否则解锁将失败。

解锁保护位置和，用户需要设置ISPCON控制寄存器决定更新LDROM，用户配置区，APROM和使能ISP控制器

一旦ISPCON寄存器设置正确，用户可以设置ISPCMD寄存器决定要擦除、读还是写。设置ISPADR地址(使用Flash内存映射地址)，ISPDAT用来设定要写的数据或者返回读到的数据

最后设定ISPTRG寄存器的ISPGO位启动操作，ISP操作完成之后，ISPGO将自清0.为了确保ISP功能正确完成，在CPU继续执行指令之前，ISPGO之后需要添加ISB指令。

启动ISPGO之后，需要检查几个错误条件. 如果错误条件产生，ISP操作失败，其失败标志置位，ISPFF标志由软件清零，而不会在下次ISP操作时被覆盖，即使ISPFF保持为“1”，下一次ISP也可以开始. 建议在每次ISP操作后，通过软件检查ISPFF位，如果ISPFF被被置为1了，就将其清零。.

当ISPGO置位时，CPU将等待ISP操作结束，在此期间，外设仍然正常工作，如果有中断请求时，CPU仍然会先执行完ISP后再响应中断.. 当ISP操作完成之后，ISPGO由硬件自动清0。用户可以通过ISPGO位来检查是否ISP操作已经完成。用户应该在ISPGO之后添加ISB指令。

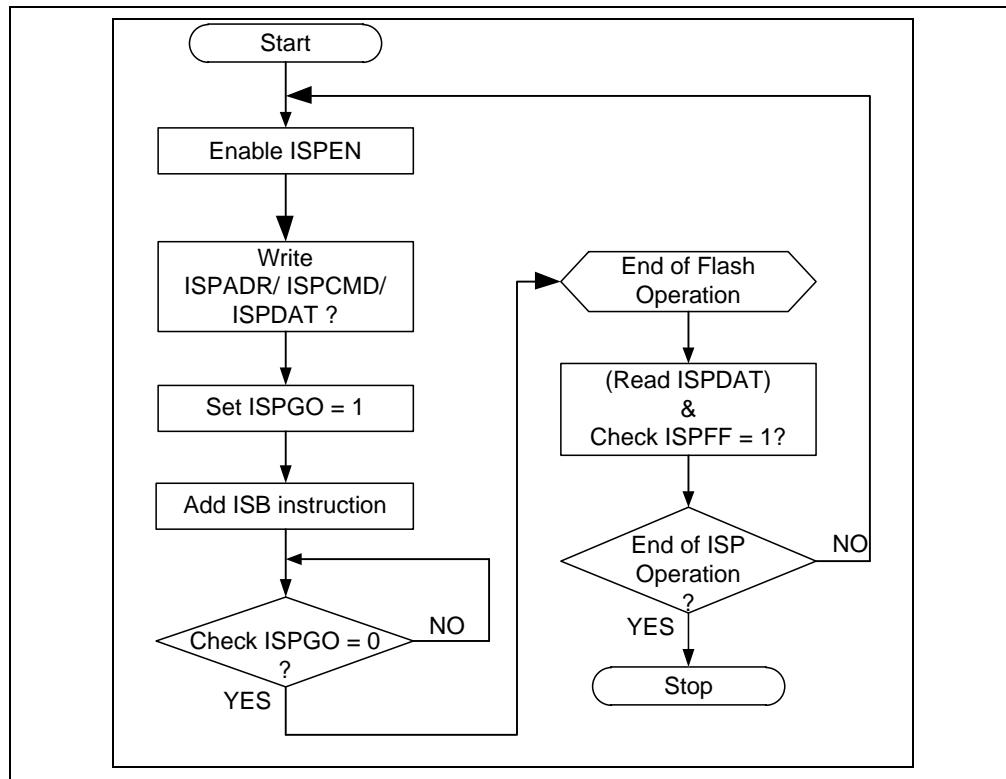


图6-36 ISP 软件编程流程

下表列出了M05xxBN/DN/DE支持的ISP命令

注：向量页重新映射命令只有M05xxDN/DE才支持

ISP 命令	ISPCMD	ISPADR	ISPDAT
FLASH Page Erase	0x22	Flash内存映射地址 必须是 512 字节对齐。	Don't care
FLASH Program	0x21	Flash内存映射地址	Programming Data
FLASH Read	0x00	Flash内存映射地址	Return Data
Read Unique ID	0x04	0x0000_0000	Unique ID Word 0
		0x0000_0004	Unique ID Word 1
		0x0000_0008	Unique ID Word 2
Read Company ID	0x0B	Don't care	Company ID (0xDA)
Vector Page Re-Map	0x2E	APROM 或者 LDROM中的某个页 必须是 512 字节对齐	Don't care

表 6-11 ISP 命令列表

### 6.7.11 通过向量重新映射多引导

M05xxDN/DE 通过向量重新映射支持从不同的地址启动。当 CBS[0] = 0, LDROM中所有页和

APROM中所有页都可以映射到地址 0x0。重新映射的地址可以从ISPSTA寄存器的VECMAP读到。当CBS[1:0] = 2时，上电时默认映射地址为0。这就意味着上电时从APROM启动，向量页映射到APROM的第一页。当CBS[1:0] = 0时，映射地址为0x100000。这就意味着上电从LDROM启动，向量页映射到LDROM的第一页。

重新映射地址可以通过向量页重新映射命令改变。通过向量页重新映射命令用户可以重新映射指定页为向量页，然后使用CPU\_RST 或者 SYSRESETREQ 来复位系统。CPU将从新的向量表取得堆栈指针和复位处理函数指针。

例如，如果用户有2个独立的应用程序都在APROM里面，分别命名为App0和App1。App0放在地址0，App1放在地址0x8000。CBS[1:0] = 0 从LDROM启动。上电时系统将执行LDROM中的代码。LDROM中的code将决定从App0启动还是从App1启动。如果从App0启动，LDROM中的代码将负责使能ISP并重新映射向量页到地址0，然后通过CPU\_RST（不复位I/O和外设）或者SYSRESETREQ（复位I/O和外设）来复位CPU从App0启动；如果从App1启动，中的代码将负责使能ISP并重新映射向量页到地址0x8000，然后通过CPU\_RST（不复位I/O和外设）或者SYSRESETREQ（复位I/O和外设）来复位CPU从App1启动。下表显示怎样通过向量表映射从不同的应用程序启动。

为了重新映射向量表，用户需要设置新的页地址到ISPADR，填重新映射命令0x2E 到ISPCMD寄存器，然后通过将ISPGO=1触发ISP。用户通过读ISPSTA寄存器确认性的向量表地址。

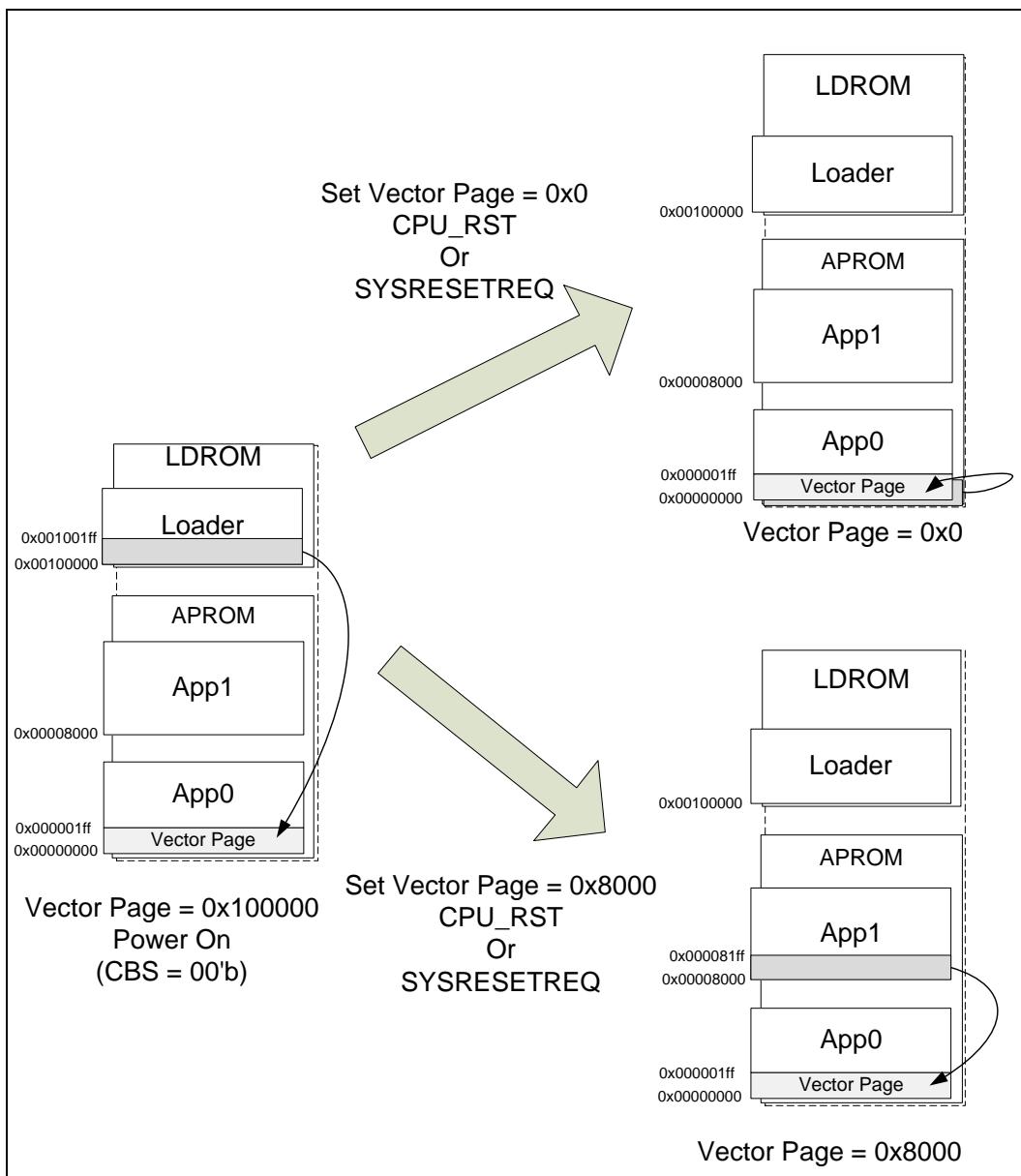


图6-37 通过向量重新映射多引导

**6.7.12 寄存器映射**

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>FMC基址:</b>				
<b>基地址 (FMC_BA) : 0x5000_C000</b>				
<b>ISPCON</b>	FMC_BA+0x00	R/W	ISP控制寄存器	0x0000_0000
<b>ISPADR</b>	FMC_BA+0x04	R/W	ISP地址寄存器	0x0000_0000
<b>ISPDAT</b>	FMC_BA+0x08	R/W	ISP数据寄存器	0x0000_0000
<b>ISPCMD</b>	FMC_BA+0x0C	R/W	ISP命令寄存器	0x0000_0000
<b>ISPTRG</b>	FMC_BA+0x10	R/W	ISP触发寄存器	0x0000_0000
<b>DFBADR</b>	FMC_BA+0x14	R	数据Flash 启始地址	0x0001_F000
<b>FATCON</b>	FMC_BA+0x18	R/W	FLASH访问时间控制寄存器	0x0000_0000
<b>ISPSTA</b>	FMC_BA+0x40	R/W	ISP状态寄存器 (只有M051xxDN/DE)	0x0000_0000



### 6.7.13 寄存器描述

#### ISP 控制寄存器(ISPCON)

寄存器	偏移量	R/W	描述	复位后的值
ISPCON	FMC_BA+0x00	R/W	ISP 控制寄存器	0x0000_0000

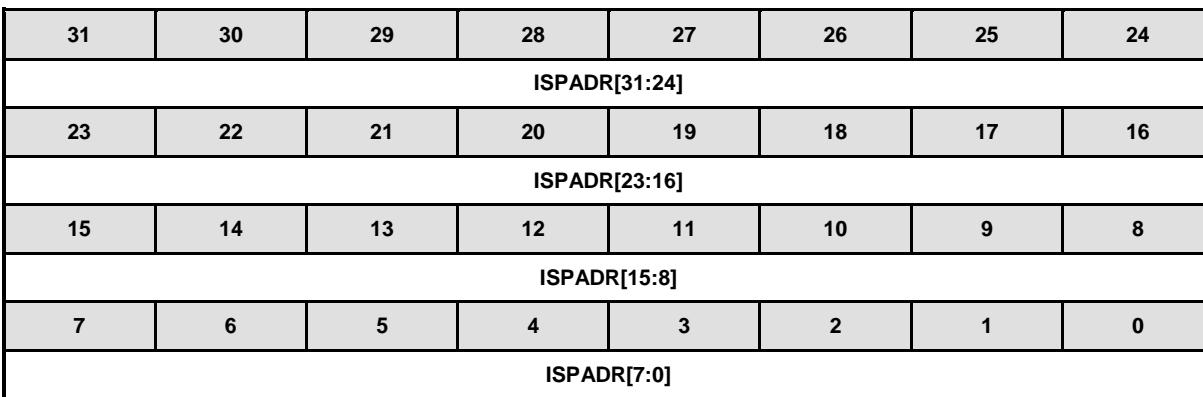
31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留	ISPFF	LDUEN	CFGUEN	APUEN	保留	BS	ISPEN

Bits	描述	
[31:7]	保留	保留
[6]	ISPFF	<b>ISP失败标志（写保护）</b> 当ISP满足下列条件时，该位由硬件置位： (1) APUEN =0时，APROM对自身写入。 (2) LDROM对自身写入。 (3) CFGUEN=0时，CONFIG 被修改 (4) 目标地址无效，如超过正常范围。 注：该位写 1 清零。
[5]	LDUEN	<b>LDROM更新使能控制（写保护）</b> LDROM 更新使能位。 1 = MCU在APROM中运行时，LDROM可以被更新。 0 = LDROM不能被更新
[4]	CFGUEN	<b>配置区更新使能控制（写保护）</b> 不管此时程序是运行在APROM还是LDROM，写1使能软件通过ISP更新配置区。 1 = 使能配置区更新 0 = 禁止配置区更新
[2]	保留	保留
[1]	BS	<b>启动选择（写保护）</b> 置位/清零该位选择下次复位时是由LDROM启动还是由APROM启动，该位可作为MCU启动状态标志，用于检查MCU是由LDROM还是APROM启动的。除了CPU复位(RSTS_CPU=1)和系统复位(RSTS_SYS)，任何复位后，该位初始值为CONFIG0的CBS的值取反。 1 = 由LDROM启动

		0 = 由 APROM启动
[0]	<b>ISPEN</b>	<b>ISP 使能 (写保护)</b> 该位是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 (ISPADR)**

寄存器	偏移量	R/W	描述	复位后的值
ISPADR	FMC_BA+ 0x04	R/W	ISP 地址寄存器	0x0000_0000



Bits	描述	
[31:0]	ISPADR	ISP 地址 M05xxBN/DN/DE 最大内置 16k x 32 的 flash (64 KB), 仅支持字编程. 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	复位后的值
ISPDAT	FMC_BA+ 0x08	R/W	ISP 数据寄存器	0x0000_0000

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

Bits	描述	
[31:0]	ISPDAT	<b>ISP 数据</b> 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 (ISPCMD)**

寄存器	偏移量	R/W	描述	复位后的值
ISPCMD	FMC_BA+ 0x0C	R/W	ISP命令寄存器	0x0000_0000

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

Bits	描述	
[31:6]	保留	保留
[5:0]	FOEN, FCEN, FCTRL	<p><b>ISP 命令</b>            ISP命令表如下:</p> <ul style="list-style-type: none"> <li>0x00 = 读</li> <li>0x04 = 读 唯一 ID</li> <li>0x0B = 读 公司 ID (0xDA)</li> <li>0x21 = 写</li> <li>0x22 = 擦除</li> <li>0x2E = 向量页重映射</li> </ul>

**ISP触发控制寄存器(ISPTRG)**

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

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

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

**数据FLASH基地址寄存器(DFBADR)**

寄存器	地址	R/W/C	描述	复位后的值
DFBADR	FMC_BA+ 0x14	R	数据FLASH基地址	0x0001_F000

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

Bits	描述	
[31:0]	DFBADR	<b>数据FLASH基地址</b> 该寄存器为数据FLASH基地址寄存器,只读. 对于 8/16/32/64KB flash 器件, 数据Flash的大小为4KB , 基地址固定为0x0001_F000.



### Flash访问时间控制寄存器 (FATCON)

寄存器	偏移量	R/W	描述	复位后的值
FATCON	FMC_BA + 0x18	R/W	Flash访问时间控制寄存器	0x0000_0000

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

Bits	描述	
[31:8]	保留	保留
[7]	保留	保留
[6:5]	保留	保留
[4]	<b>LFOM</b>	低频优化模式(写保护位) 当芯片操作频率低于25MHz时，通过设置该位为1，系统可以工作的更有效率 1 = 使能flash低频优化模式 0 = 禁用flash低频优化模式
[3:0]	保留	保留



### ISP 状态寄存器 (ISPSTA)

寄存器	偏移量	R/W	描述	复位后的值
ISPSTA	FMC_BA+0x40	R/W	ISP 状态寄存器(只有 M05xxDN/DE )	0x0000_0000

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

Bits	描述	
[31:21]	保留	保留.
[20:9]	VECMAP	向量页映射地址 (只读) Flash 地址空间 {VECMAP[11:0], 9'h000} ~ {VECMAP[11:0], 9'h1FF} 目前被映射到 0x0000_0000~0x0000_01FF
[8:7]	保留	保留.
[6]	ISPFF	<b>ISP 失败标志(写保护)</b> 触发ISP之后如果遇到如下情形, 该位由硬件置位: (1) APUEN =0时, APROM对自身写入. (2) LDROM对自身写入. (3) CFGUEN=0时, CONFIG 被修改 (4) 目标地址无效, 如超过正常范围.  该位写1清0. <b>注:</b> 这个位和ISPCON的bit[6]功能一样
[5:3]	保留	保留.
[2:1]	CBS	<b>芯片启动选择 (只读)</b> 该位与CONFIG0的CBS位功能相同.
[0]	ISPGO	<b>ISP 触发 (只读)</b> 写1开始 ISP 操作, 当ISP操作完成以后, 该位由硬件自动清0 1 = ISP 操作正在进行中. 0 = ISP 操作已经完成. <b>注:</b> 该位和ISPTRG的 bit0功能相同

## 6.8 通用 I/O (GPIO)

### 6.8.1 概述

NuMicro™ M05xxBN 和 M05xxDN/DE 最多有 40 个通用 I/O 引脚，这些引脚和其它功能共享。40 个引脚分为 5 个端口，分别命名为 P0, P1, P2, P3 和 P4，每个端口最多有 8 个引脚。每个引脚都是独立的，都有相应的寄存器来控制引脚工作模式与数据。

每个引脚的 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。复位后，M05xxBN 默认所有的 I/O 引脚处于准双向模式，M05xxDN/DE 默认 I/O 引脚的模式由 CONFIG0 的 CIOINI 位决定(准双向或者输入模式)，端口数据寄存器 Px\_DOUT[7:0] 的值为 0x000\_00FF。每个 I/O 引脚配有一个非常弱的独立的上拉电阻，VDD 从 5.0V 到 2.5V 时，内部弱上拉电阻阻值大约为 110KΩ~300KΩ。

### 6.8.2 特性

- 4 种 I/O 模式：
  - 输入模式带高阻
  - 推挽输出
  - 开漏输出
  - 准双向
- 准双向 TTL/Schmitt 触发输入模式由 Px\_MFP[23:16] 选择
- 每个 I/O 引脚都可以作为中断源，支持边沿/电平触发
- 准双向模式下，I/O 引脚内部上拉电阻被使能
- 引脚中断功能使能后，引脚的唤醒功能也将被使能
- 复位后，所有 I/O 引脚默认模式由 CIOINI(CONFIG[10]) 决定(只有 M05xxDN/DE)
  - CIOINI = 0, 复位后所有 GPIO 引脚为输入三态模式
  - CIOINI = 1, 复位后所有 GPIO 引脚为准双向模式

### 6.8.3 基本配置

GPIO 引脚功能由 P0\_MFP, P1\_MFP, P2\_MFP, P3\_MFP 和 P4\_MFP 寄存器配置。

## 6.8.4 功能描述

### 6.8.4.1 输入模式

设置 Px\_PMD(PMDn[1:0]) 为 00'b，Px.n 为输入模式，I/O 引脚为三态（高阻态），没有输出驱动能力。Px\_PIN 的值反映相应端口引脚的状态。

### 6.8.4.2 输出模式

设置 Px\_PMD(PMDn[1:0]) 为 01'b，Px.n 为输出模式，I/O 引脚支持数字输出功能，有拉电流/灌电流能力。Px\_DOUT 相应位的值被送到相应引脚上。

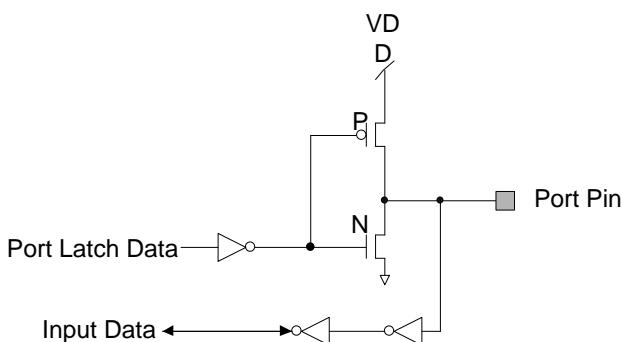


图 6-38 推挽输出

### 6.8.4.3 开漏输出模式

设置 Px\_PMD(PMDn[1:0]) 为 10'b，Px.n 为开漏模式，I/O 支持数字输出功能，但仅有灌电流能力，为了把 I/O 引脚拉到高电平状态，需要外接一颗上拉电阻。如果 Px\_DOUT 相应位的值为“0”，引脚上输出低电平。如果 Px\_DOUT 相应位的值为“1”，该引脚由外部上拉电阻控制输出高电平。

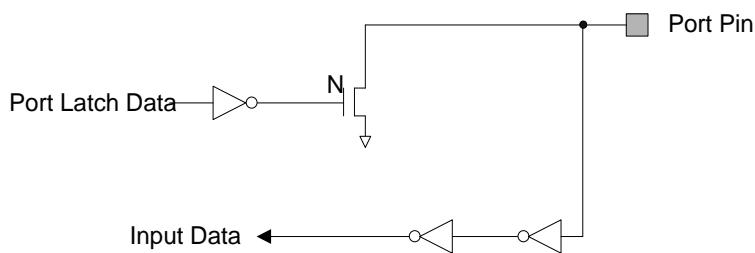


图 6-39 开漏输出

### 6.8.4.4 准双向模式

设置 Px\_PMD(PMDn[1:0]) 为 11'b，Px.n 引脚为准双向模式，I/O 同时支持数字输出和输入功能，但拉电流仅达数百 uA。要实现数字输入功能，需要先将 Px\_DOUT 相应位置 1。准双向输出是 80C51 及其派生

品所共有的模式。若Px\_DOUT相应位为“0”，引脚上输出为“低电平”。若Px\_DOUT相应位为“1”，该引脚将检查引脚值。若引脚值为高，没有任何动作，若引脚值为低，**该引脚将驱动2个时钟周期的强拉电流**，然后关闭强输出驱动，然后引脚状态由内部上拉电阻控制。

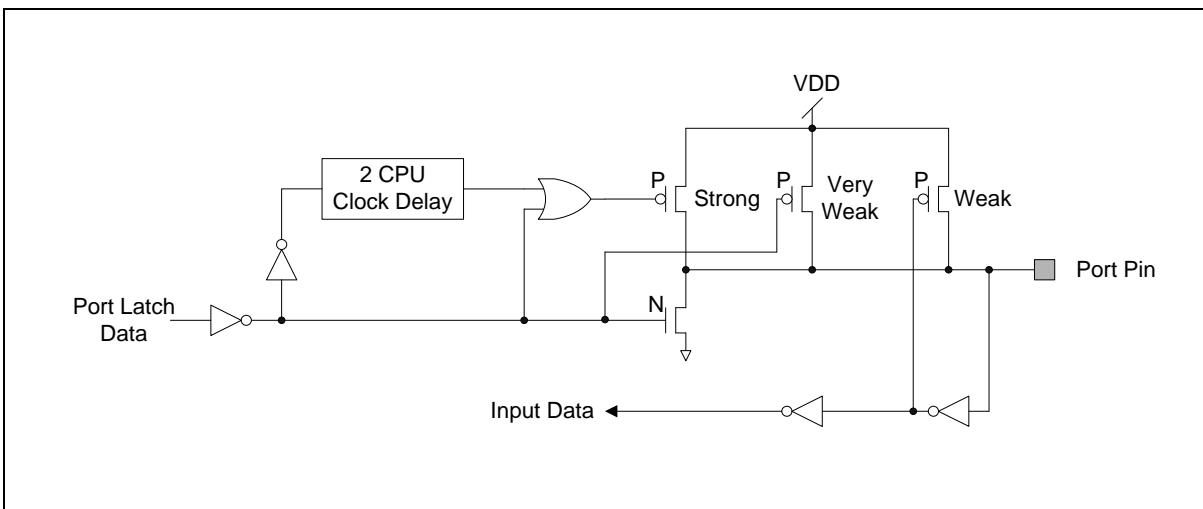


图 6-40 混双端 I/O 模式

### 6.8.5 GPIO 中断和唤醒功能

通过Px\_IEN 和 Px\_IMD寄存器，每个GPIO引脚都可以设置为中断源。有5种类型的中断条件可以选择：低电平触发、高电平触发、下降沿触发、上升沿触发、上升和下降沿都触发。对于边沿触发，用户可以使能输入信号去抖动功能来避免由干扰导致的不期望的中断发生。**de-bounce** 时钟源和采样周期可以由DBNCECON 寄存器设定。

当芯片进入空闲/睡眠模式时，GPIO也能唤醒芯片。如果使用GPIO唤醒芯片，唤醒触发条件和中断触发条件一致，但是有2件事情需要注意(只有M05xxBN)：

#### 1. 进入空闲/睡眠模式之前要确定 I/O 的状态

当时有GPIO反转来唤醒系统时，在进入空闲/睡眠模式之前，用户必须确认I/O口的状态。

如果设定的是高电平/上升沿唤醒，进入空闲/睡眠模式之前必须确认I/O为低电平；如果设定的是低电平/下降沿唤醒，进入空闲/睡眠模式之前必须确认I/O为高电平。

#### 2. 进入空闲/睡眠模式之前要关闭I/O去抖动功能

如果指定的I/O唤醒引脚使能了输入信号去抖动功能，当系统由GPIO唤醒时，系统将发生两次GPIO中断事件。一个中断事件由唤醒导致，另一个由 I/O 输入去抖动功能导致。使能空闲/睡眠功能之前，用户应该关闭去抖动功能，以避免两次中断事件。



### 6.8.6 寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>GPIO基地址</b>				
<b>GP_BA = 0x5000_4000</b>				
P0_PMD	GP_BA+0x000	R/W	P0 I/O模式控制	0x0000_XXXX
P0_OFFD	GP_BA+0x004	R/W	P0 关闭数字输入通路控制	0x0000_0000
P0_DOUT	GP_BA+0x008	R/W	P0数据输出值	0x0000_00FF
P0_DMASK	GP_BA+0x00C	R/W	P0数据输出写屏蔽	0x0000_0000
P0_PIN	GP_BA+0x010	R	P0管脚数值	0x0000_00XX
P0_DBEN	GP_BA+0x014	R/W	P0防抖动使能控制	0x0000_0000
P0_IMD	GP_BA+0x018	R/W	P0中断模式控制	0x0000_0000
P0_IEN	GP_BA+0x01C	R/W	P0中断使能控制	0x0000_0000
P0_ISRC	GP_BA+0x020	R/WC	P0 中断源标志	0x0000_0000
P1_PMD	GP_BA+0x040	R/W	P1 Bit模式使能	0x0000_XXXX
P1_OFFD	GP_BA+0x044	R/W	P1 Bit OFF数字使能	0x0000_0000
P1_DOUT	GP_BA+0x048	R/W	P1数据输出值	0x0000_00FF
P1_DMASK	GP_BA+0x04C	R/W	P1 数据输出写屏蔽	0x0000_0000
P1_PIN	GP_BA+0x050	R	P1管脚数值	0x0000_00XX
P1_DBEN	GP_BA+0x054	R/W	P1防反弹使能	0x0000_0000
P1_IMD	GP_BA+0x058	R/W	P1 中断模式控制	0x0000_0000
P1_IEN	GP_BA+0x05C	R/W	P1 中断使能	0x0000_0000
P1_ISRC	GP_BA+0x060	R/WC	P1 中断源标志	0x0000_0000
P2_PMD	GP_BA+0x080	R/W	P2 Bit 模式使能	0x0000_XXXX
P2_OFFD	GP_BA+0x084	R/W	P2 Bit OFF数字使能	0x0000_0000
P2_DOUT	GP_BA+0x088	R/W	P2数据输出值	0x0000_00FF
P2_DMASK	GP_BA+0x08C	R/W	P2数据输出写屏蔽	0x0000_0000
P2_PIN	GP_BA+0x090	R	P2 管脚数值	0x0000_00XX
P2_DBEN	GP_BA+0x094	R/W	P2防反弹使能	0x0000_0000
P2_IMD	GP_BA+0x098	R/W	P2中断模式控制	0x0000_0000
P2_IEN	GP_BA+0x09C	R/W	P2中断使能	0x0000_0000

<b>P2_ISRC</b>	GP_BA+0x0A0	R/WC	P2中断源标志	0x0000_0000
<b>P3_PMD</b>	GP_BA+0x0C0	R/W	P3 Bit模式使能	0x0000_XXXX
<b>P3_OFFD</b>	GP_BA+0x0C4	R/W	P3 Bit OFF数字使能	0x0000_0000
<b>P3_DOUT</b>	GP_BA+0x0C8	R/W	P3数据输出值	0x0000_00FF
<b>P3_DMASK</b>	GP_BA+0x0CC	R/W	P3 数据输出写屏蔽	0x0000_0000
<b>P3_PIN</b>	GP_BA+0xD0	R	P3管脚数值	0x0000_00XX
<b>P3_DBEN</b>	GP_BA+0xD4	R/W	P3防反弹使能	0x0000_0000
<b>P3_IMD</b>	GP_BA+0xD8	R/W	P3中断模式控制	0x0000_0000
<b>P3_IEN</b>	GP_BA+0xDC	R/W	P3中断使能	0x0000_0000
<b>P3_ISRC</b>	GP_BA+0xE0	R/WC	P3中断源标志	0x0000_0000
<b>P4_PMD</b>	GP_BA+0x100	R/W	P4 Bit模式使能	0x0000_XXXX
<b>P4_OFFD</b>	GP_BA+0x104	R/W	P4 Bit OFF数字使能	0x0000_0000
<b>P4_DOUT</b>	GP_BA+0x108	R/W	P4数据输出值	0x0000_00FF
<b>P4_DMASK</b>	GP_BA+0x10C	R/W	P4 数据输出写屏蔽	0x0000_0000
<b>P4_PIN</b>	GP_BA+0x110	R	P4 E管脚数值	0x0000_00XX
<b>P4_DBEN</b>	GP_BA+0x114	R/W	P4 防反弹使能	0x0000_0000
<b>P4_IMD</b>	GP_BA+0x118	R/W	P4 中断模式控制	0x0000_0000
<b>P4_IEN</b>	GP_BA+0x11C	R/W	P4 中断使能	0x0000_0000
<b>P4_ISRC</b>	GP_BA+0x120	R/WC	P4 中断源标志	0x0000_0000
<b>DBNCECON</b>	GP_BA+0x180	R/W	去抖动周期控制	0x0000_0020
<b>P00_DOUT</b>	GP_BA+0x200	R/W	P0.0数据输出值	0x0000_000 X
<b>P01_DOUT</b>	GP_BA+0x204	R/W	P0.1数据输出值	0x0000_000 X
<b>P02_DOUT</b>	GP_BA+0x208	R/W	P0.2数据输出值	0x0000_000 X
<b>P03_DOUT</b>	GP_BA+0x20C	R/W	P0.3数据输出值	0x0000_000 X
<b>P04_DOUT</b>	GP_BA+0x210	R/W	P0.4数据输出值	0x0000_000 X
<b>P05_DOUT</b>	GP_BA+0x214	R/W	P0.5数据输出值	0x0000_000 X
<b>P06_DOUT</b>	GP_BA+0x218	R/W	P0.6数据输出值	0x0000_000 X
<b>P07_DOUT</b>	GP_BA+0x21C	R/W	P0.7数据输出值	0x0000_000 X
<b>P10_DOUT</b>	GP_BA+0x220	R/W	P1.0数据输出值	0x0000_000 X
<b>P11_DOUT</b>	GP_BA+0x224	R/W	P1.1数据输出值	0x0000_000 X

<b>P12_DOUT</b>	GP_BA+0x228	R/W	P1.2数据输出值	0x0000_000 X
<b>P13_DOUT</b>	GP_BA+0x22C	R/W	P1.3数据输出值	0x0000_000 X
<b>P14_DOUT</b>	GP_BA+0x230	R/W	P1.4数据输出值	0x0000_000 X
<b>P15_DOUT</b>	GP_BA+0x234	R/W	P1.5数据输出值	0x0000_000 X
<b>P16_DOUT</b>	GP_BA+0x238	R/W	P1.6数据输出值	0x0000_000 X
<b>P17_DOUT</b>	GP_BA+0x23C	R/W	P1.7数据输出值	0x0000_000 X
<b>P20_DOUT</b>	GP_BA+0x240	R/W	P2.0数据输出值	0x0000_000 X
<b>P21_DOUT</b>	GP_BA+0x244	R/W	P2.1数据输出值	0x0000_000 X
<b>P22_DOUT</b>	GP_BA+0x248	R/W	P2.2数据输出值	0x0000_000 X
<b>P23_DOUT</b>	GP_BA+0x24C	R/W	P2.3 数据输出值	0x0000_000 X
<b>P24_DOUT</b>	GP_BA+0x250	R/W	P2.4 数据输出值	0x0000_000 X
<b>P25_DOUT</b>	GP_BA+0x254	R/W	P2.5 数据输出值	0x0000_000 X
<b>P26_DOUT</b>	GP_BA+0x258	R/W	P2.6 数据输出值	0x0000_000 X
<b>P27_DOUT</b>	GP_BA+0x25C	R/W	P2.7 数据输出值	0x0000_000 X
<b>P30_DOUT</b>	GP_BA+0x260	R/W	P3.0 数据输出值	0x0000_000 X
<b>P31_DOUT</b>	GP_BA+0x264	R/W	P3.1 数据输出值	0x0000_000 X
<b>P32_DOUT</b>	GP_BA+0x268	R/W	P3.2 数据输出值	0x0000_000 X
<b>P33_DOUT</b>	GP_BA+0x26C	R/W	P3.3 数据输出值	0x0000_000 X
<b>P34_DOUT</b>	GP_BA+0x270	R/W	P3.4 数据输出值	0x0000_000 X
<b>P35_DOUT</b>	GP_BA+0x274	R/W	P3.5 数据输出值	0x0000_000 X
<b>P36_DOUT</b>	GP_BA+0x278	R/W	P3.6 数据输出值	0x0000_000 X
<b>P37_DOUT</b>	GP_BA+0x27C	R/W	P3.7 数据输出值	0x0000_000 X
<b>P40_DOUT</b>	GP_BA+0x280	R/W	P4.0 数据输出值	0x0000_000 X
<b>P41_DOUT</b>	GP_BA+0x284	R/W	P4.1 数据输出值	0x0000_000 X
<b>P42_DOUT</b>	GP_BA+0x288	R/W	P4.2 数据输出值	0x0000_000 X
<b>P43_DOUT</b>	GP_BA+0x28C	R/W	P4.3 数据输出值	0x0000_000 X
<b>P44_DOUT</b>	GP_BA+0x290	R/W	P4.4 数据输出值	0x0000_000 X
<b>P45_DOUT</b>	GP_BA+0x294	R/W	P4.5 数据输出值	0x0000_000 X
<b>P46_DOUT</b>	GP_BA+0x298	R/W	P4.6 数据输出值	0x0000_000 X
<b>P47_DOUT</b>	GP_BA+0x29C	R/W	P4.7数据输出值	0x0000_000 X

### 6.8.7 寄存器描述

#### Port 0-4 I/O 模式控制(Px\_PMD)

寄存器	偏移量	R/W	描述	复位后的值
P0_PMD	GP_BA+0x000	R/W	P0 Pin I/O 模式控制	0x0000_XXXX
P1_PMD	GP_BA+0x040	R/W	P1 Pin I/O 模式控制	0x0000_XXXX
P2_PMD	GP_BA+0x080	R/W	P2 Pin I/O 模式控制	0x0000_XXXX
P3_PMD	GP_BA+0x0C0	R/W	P3 Pin I/O 模式控制	0x0000_XXXX
P4_PMD	GP_BA+0x100	R/W	P4 Pin I/O 模式控制	0x0000_XXXX

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
PMD7		PMD6		PMD5		PMD4	
7	6	5	4	3	2	1	0
PMD3		PMD2		PMD1		PMD0	

Bits	描述	
[31:16]	保留	保留
[2n+1 :2n]	PMDn	<p><b>P0 ~ P4 I/O Pin[n] 模式控制</b>            决定Px.n每个引脚的I/O模式            00 = Px.n为 输入模式.            01 = Px.n 为输出模式.            10 = Px.n 为开漏模式.            11 = Px.n 为准双端模式            注1: x=0~4, n = 0~7            注2: M05xxBN所有引脚的默认值为0x0000_FFFF, 就是说所有的引脚缺省为准双向模式            注3: M05xxDN/DE该域的默认值由CIOINI(CONFIG[10])决定            如果CIOINI等于1, 所有引脚的默认值为0x0000_FFFF, 就是说所有的引脚缺省为准双向模式            如果CIOINI等于0, 所有引脚的默认值为0x0000_0000, 就是说所有的引脚缺省为输入三态模式</p>

**Port 0-4数字输入通路关闭控制 (Px\_OFFD)**

寄存器	偏移量	R/W	描述	复位后的值
P0_OFFD	GP_BA+0x004	R/W	P0 数字输入通道关闭控制	0x0000_0000
P1_OFFD	GP_BA+0x044	R/W	P1 数字输入通道关闭控制	0x0000_0000
P2_OFFD	GP_BA+0x084	R/W	P2 数字输入通道关闭控制	0x0000_0000
P3_OFFD	GP_BA+0x0C4	R/W	P3 数字输入通道关闭控制	0x0000_0000
P4_OFFD	GP_BA+0x104	R/W	P4 数字输入通道关闭控制	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:16]	OFFD	<p><b>Px Pin[n]数字输入通道关闭控制</b></p> <p>该域每个位用来控制是否关闭相应Px.n引脚数字输入通路。如果某个引脚输入的是模拟信号，用户应该关闭相应引脚的数字通路避免漏电</p> <p>1 = 关闭Px.n数字输入通道（数字输入连接到低电平） 0 = 使能Px.n 数字输入通道 注: x=0~4, n = 0~7</p>
[15:0]	保留	保留

**Port 0-4数据输出值(Px\_DOUT)**

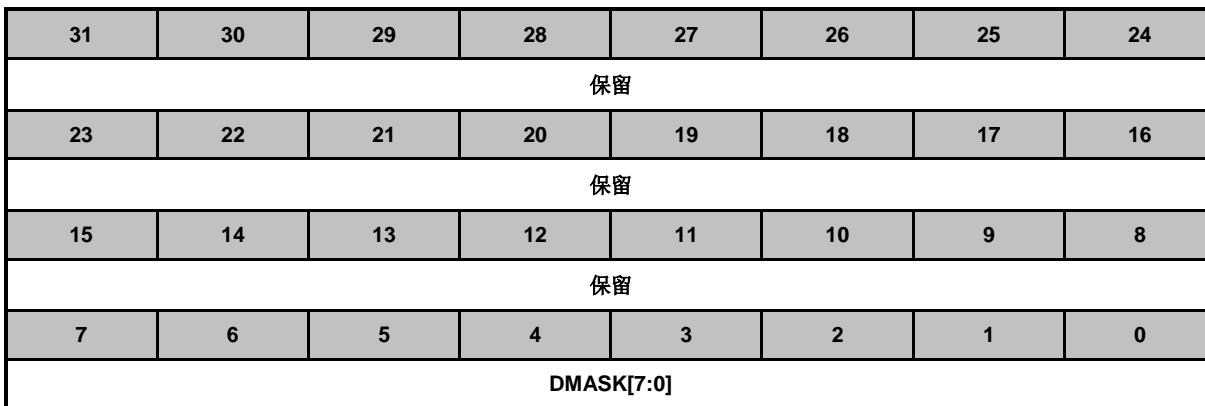
寄存器	偏移量	R/W	描述	复位后的值
P0_DOUT	GP_BA+0x008	R/W	P0数据输出值	0x0000_00FF
P1_DOUT	GP_BA+0x048	R/W	P1数据输出值	0x0000_00FF
P2_DOUT	GP_BA+0x088	R/W	P2数据输出值	0x0000_00FF
P3_DOUT	GP_BA+0xC8	R/W	P3数据输出值	0x0000_00FF
P4_DOUT	GP_BA+0x108	R/W	P4数据输出值	0x0000_00FF



Bits	描述	
[31:8]	保留	保留
[n]	DOUT[n]	<b>Px Pin[n] 输出值</b> Px.n配置成输出, 开漏, 和准双向模式时, 这些位控制Px.n引脚状态. 1 = Px.n 引脚状态驱动为高电平. 0 = Px.n 引脚状态驱动为低电平. 注: x=0~4, n = 0~7

**Port0-4数据输出写屏蔽(Px\_DMASK)**

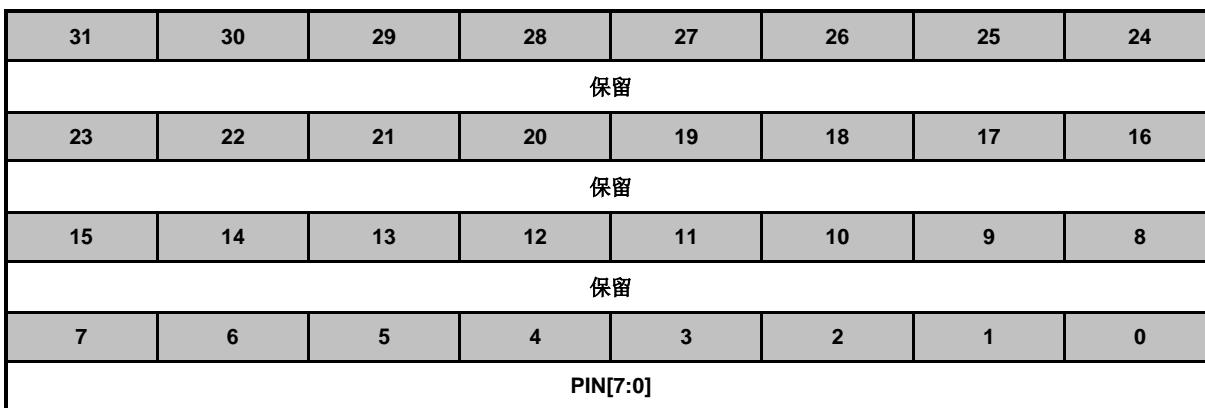
寄存器	偏移量	R/W	描述	复位后的值
P0_DMASK	GP_BA+0x00C	R/W	P0数据输出写屏蔽	0x0000_0000
P1_DMASK	GP_BA+0x04C	R/W	P1数据输出写屏蔽	0x0000_0000
P2_DMASK	GP_BA+0x08C	R/W	P2数据输出写屏蔽	0x0000_0000
P3_DMASK	GP_BA+0x0CC	R/W	P3数据输出写屏蔽	0x0000_0000
P4_DMASK	GP_BA+0x10C	R/W	P4数据输出写屏蔽	0x0000_0000



Bits	描述	
[31:8]	保留	保留
[n]	DMASK[n]	<p><b>Px 数据输出写屏蔽 (写保护)</b></p> <p>用于保护寄存器Px_DOUT 的相应位. 当设置DMASK[n] 为 “1”时， Px_DOUT 相应位 被保护，写Px_DOUT[n]相应位不起作用</p> <p>0 = 相应的Px_DOUT [n] 位未被屏蔽 1 = 相应的Px_DOUT [n] 位被屏蔽</p> <p><b>注1:</b> x=0~4, n = 0~7 <b>注2:</b> 该功能只是保护相应的Px_DOUT[n]位，但是不保护Pxx_DOUT寄存器中的位</p>

**Port 0-4管脚状态(Px\_PIN)**

寄存器	偏移量	R/W	描述	复位后的值
P0_PIN	GP_BA+0x010	R	P0管脚状态	0x0000_00XX
P1_PIN	GP_BA+0x050	R	P1管脚状态	0x0000_00XX
P2_PIN	GP_BA+0x090	R	P2管脚状态	0x0000_00XX
P3_PIN	GP_BA+0x0D0	R	P3管脚状态	0x0000_00XX
P4_PIN	GP_BA+0x110	R	P4管脚状态	0x0000_00XX



Bits	描述	
[31:8]	保留	保留
[n]	PIN[n]	<p><b>Px 管脚状态</b></p> <p>这些位的值反映各个Px.n 引脚的实际状态为高还是低，如果某个位的值为1，则相应引脚的状态为高电平，否则就是低电平</p> <p>注：x=0~4, n = 0~7</p>

**Port 0-4去抖动使能(Px\_DBEN)**

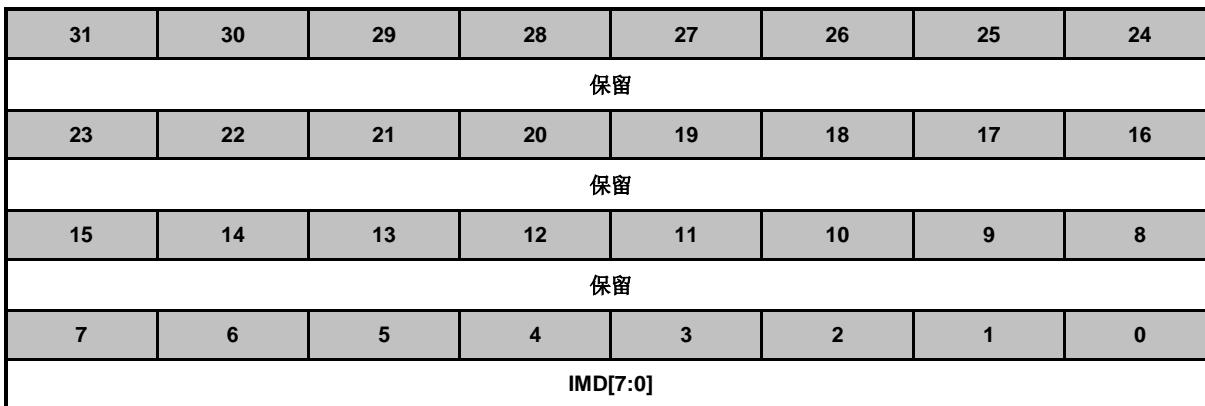
寄存器	偏移量	R/W	描述	复位后的值
P0_DBEN	GP_BA+0x014	R/W	P0去抖动使能控制	0x0000_0000
P1_DBEN	GP_BA+0x054	R/W	P1去抖动使能控制	0x0000_0000
P2_DBEN	GP_BA+0x094	R/W	P2去抖动使能控制	0x0000_0000
P3_DBEN	GP_BA+0xD4	R/W	P3去抖动使能控制	0x0000_0000
P4_DBEN	GP_BA+0x114	R/W	P4去抖动使能控制	0x0000_0000



Bits	描述	
[31:8]	保留	保留
[n]	DBEN[n]	<p><b>Px输入信号去抖动使能控制</b></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，则输入信号被视为信号抖动，从而不触发中断。去抖动时钟源由DBNCECON[4]控制，去抖动采样周期由DBNCECON[3:0]控制</p> <p>0 = 禁用 Px.n去抖动功能 1 = 使能 Px.n去抖动功能</p> <p>去抖动功能仅限于边沿触发 中断，不用于电平触发中断</p> <p><b>注1:</b> x=0~4, n = 0~7</p> <p><b>注2:</b> Px.n引脚如果作为唤醒源，进入睡眠模式之前，为了避免发生两次中断，用户应该关闭 Px.n引脚的去抖动功能</p>

**Port 0-4中断模式控制(Px\_IMD)**

寄存器	偏移量	R/W	描述	复位后的值
P0_IMD	GP_BA+0x018	R/W	P0中断模式控制	0x0000_0000
P1_IMD	GP_BA+0x058	R/W	P1中断模式控制	0x0000_0000
P2_IMD	GP_BA+0x098	R/W	P2中断模式控制	0x0000_0000
P3_IMD	GP_BA+0x0D8	R/W	P3中断模式控制	0x0000_0000
P4_IMD	GP_BA+0x118	R/W	P4中断模式控制	0x0000_0000



Bits	描述	
[31:8]	保留	保留
[n]	IMD[n]	<p><b>Port 0-4引脚边沿或者电平中断模式控制</b></p> <p>IMD[n] 用于控制中断是电平触发还是边沿触发。若中断是边沿触发，触发源可以由去抖动控制，如果中断是电平触发，输入信号源由一个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>如果引脚设置为电平触发，寄存器Px_IEN中只有一种电平(高电平/低电平)可以被设定。若设置两种电平都触发中断，设置将被忽略，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：x=0~4, n = 0~7</p>

**Port 0-4中断使能控制(Px\_IEN)**

寄存器	偏移量	R/W	描述	复位后的值
P0_IEN	GP_BA+0x01C	R/W	P0中断使能控制	0x0000_0000
P1_IEN	GP_BA+0x05C	R/W	P1中断使能控制	0x0000_0000
P2_IEN	GP_BA+0x09C	R/W	P2中断使能控制	0x0000_0000
P3_IEN	GP_BA+0x0DC	R/W	P3中断使能控制	0x0000_0000
P4_IEN	GP_BA+0x11C	R/W	P4中断使能控制	0x0000_0000

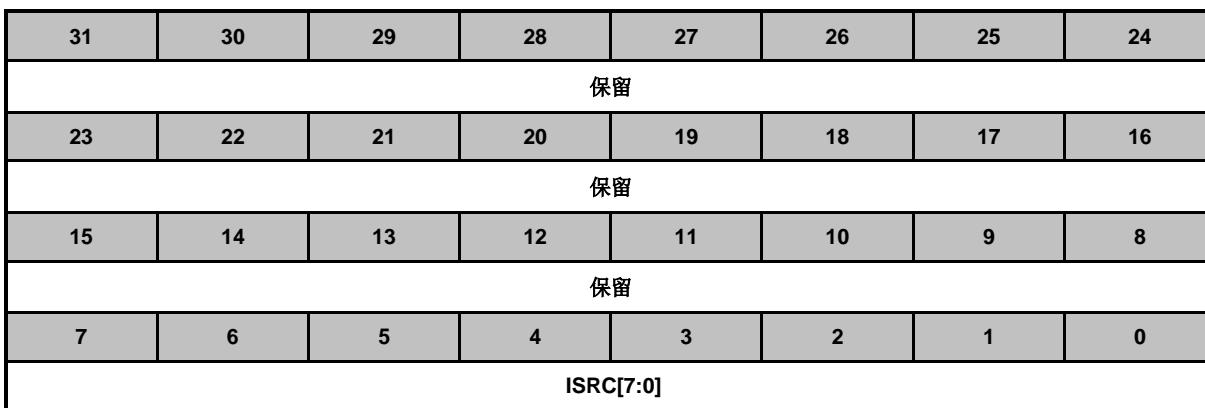
31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
IR_EN[7:0]							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
IF_EN[7:0]							

Bits	描述	
[31:24]	保留	保留
[23:16]	IR_EN[n]	<p><b>Port 0-4 输入上升沿或输入高电平中断使能</b></p> <p>IR_EN[n] 用于使能相应Px.n引脚的中断。置“1”也同时使能引脚唤醒功能</p> <p>设置 IR_EN[n] 位为“1”时：</p> <p>如果中断是电平触发模式(IMD[n] = 1)，输入Px.n引脚的状态为高电平时，产生中断。</p> <p>如果中断是边沿触发模式(IMD[n] = 0)，输入Px.n引脚的状态由低电平到高电平变化时，产生中断</p> <p>1 = 使能Px.n 高电平或由低电平到高电平变化的中断</p> <p>0 = 禁用Px.n 高电平或由低电平到高电平变化的中断.</p> <p>注：x=0~4, n = 0~7</p>
[15:8]	保留	保留

[7:0]	<b>IF_EN[n]</b>	<p><b>Port 0-4输入下降沿或输入低电平的中断使能</b></p> <p>IF_EN[n] 用于使能相应Px.n输入引脚的中断。置“1”也同时使能引脚唤醒功能</p> <p>设置 IF_EN[n] 位为“1”时：</p> <p>如果中断是电平触发模式(IMD[n] = 1)，输入Px.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(IMD[n] = 0)，输入Px.n的状态由高电平到低电平变化时，产生中断。</p> <p>1 = 使能Px.n低电平或由高电平到低电平变化的中断</p> <p>0 = 禁用Px.n低电平或由高电平到低电平变化的中断</p> <p>注：x=0~4, n = 0~7</p>
-------	-----------------	---

**Port 0-4 中断源标志(Px\_ISRC)**

寄存器	偏移量	R/W	描述	复位后的值
P0_ISRC	GP_BA+0x020	R/W	P0中断源标志	0x0000_0000
P1_ISRC	GP_BA+0x060	R/W	P1中断源标志	0x0000_0000
P2_ISRC	GP_BA+0x0A0	R/W	P2中断源标志	0x0000_0000
P3_ISRC	GP_BA+0x0E0	R/W	P3中断源标志	0x0000_0000
P4_ISRC	GP_BA+0x120	R/W	P4中断源标志	0x0000_0000



Bits	描述	
[31:8]	保留	保留
[7:0]	ISRC[n]	<p><b>Port 0-4 中断触发源标志</b></p> <p>读：</p> <p>1 = Px.n产生中断 0 = Px.n没有中断</p> <p>写：</p> <p>1= 清相应的中断标志 0= 无动作</p> <p>注： x=0~4, n = 0~7</p>

中断去抖动周期控制(DBNCECON)

寄存器	偏移量	R/W	描述	复位后的值
DBNCECON	GP_BA+0x180	R/W	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		<b>ICLK_ON</b>	<b>DBCLKSRC</b>	<b>DBCLKSEL</b>				

Bits	描述	
[5]	<b>ICLK_ON</b>	<p><b>中断时钟打开模式</b>            0 = 只有当Px_IEN 的的相应位等于1时，边沿触发电路才被激活            1 = 总是使能所有引脚边沿检测电路的时钟  <b>注：</b>如果没有特别考量，建议关闭该位以节省功耗         </p>
[4]	<b>DBCLKSRC</b>	<p><b>去抖动计数器时钟源选择</b>            1 = 去抖动计数器时钟源为内部10KHz 时钟            0 = 去抖动计数器时钟源为 HCLK         </p>

[3:0]	<b>DBCLKSEL</b>	去抖动采样周期选择 0000 = 每 1 个时钟周期采样中断输入一次. 0001 = 每 2 个时钟周期采样中断输入一次. 0010 = 每 4 个时钟周期采样中断输入一次. 0011 = 每 8 个时钟周期采样中断输入一次. 0100 = 每16 个时钟周期采样中断输入一次. 0101 = 每32 个时钟周期采样中断输入一次. 0110 = 每64 个时钟周期采样中断输入一次. 0111 = 每128 个时钟周期采样中断输入一次. 1000 = 每256 个时钟周期采样中断输入一次. 1001 = 每2*256 个时钟周期采样中断输入一次. 1010 = 每4*256 个时钟周期采样中断输入一次. 1011 = 每8*256 个时钟周期采样中断输入一次. 1100 = 每16*256 个时钟周期采样中断输入一次. 1101 = 每32*256 个时钟周期采样中断输入一次. 1110 = 每64*256 个时钟周期采样中断输入一次. 1111 = 每128*256 个时钟周期采样中断输入一次.
-------	-----------------	---

GPIO 端口 [P0/P1/P2/P3/P4] I/O 位输出控制 (Pxx\_DOUT)

寄存器	偏移量	R/W	描述	复位后的值
P0x_DOUT	GP_BA+0x200 - GP_BA+0x21C	R/W	P0.n 引脚数据输入/输出控制	0x0000_000 X
P1x_DOUT	GP_BA+0x220 - GP_BA+0x23C	R/W	P1.n 引脚数据输入/输出控制	0x0000_000 X
P2x_DOUT	GP_BA+0x240 - GP_BA+0x25C	R/W	P2.n 引脚数据输入/输出控制	0x0000_000 X
P3x_DOUT	GP_BA+0x260 - GP_BA+0x27C	R/W	P3.n 引脚数据输入/输出控制	0x0000_000 X
P4x_DOUT	GP_BA+0x280 - GP_BA+0x29C	R/W	P4.n 引脚数据输入/输出控制	0x0000_000 X

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							Pxx_DOUT

Bits	描述	
[0]	Pxx_DOUT	<p><b>Px.n</b> 引脚输入/输出控制</p> <p>写该位可以控制一个GPIO管脚的输出值</p> <p>1 = 设置相应的GPIO位为高</p> <p>0 = 设置相应的GPIO位为低</p> <p>读这个寄存器可以得到IO引脚的状态</p> <p>例如：写 P00_DOUT 将反应写的值到寄存器 P0_DOUT[0]，读 P00_DOUT 将得到 P0_PIN[0]的值</p> <p><b>注1:</b> x = 0~4, n = 0~7.</p> <p><b>注2:</b> 该寄存器的写操作不受寄存器 Px_DMASK[n]的影响.</p>

## 6.9 硬件除法器(HDIV) (只有M05xxDN/DE)

### 6.9.1 概述

硬件除法器 (HDIV) 用来提高应用程序的效率。硬件除法器是一个有符号，整数除法器，提供商和余数输出。硬件除法器只有M05xxDN/DE才支持。

### 6.9.2 特性

- 有符号 (2的补码) 整数计算
- 32-bit 被除数, 16位除数计算能力
- 32-bit 商和 32-bit 余数输出 (16-bit 余数带符号扩展到32位)
- 除0警告标志
- 每次计算花6个HCLK 时钟周期
- 写除数触发计算
- 当读商和余数的时候自动等待计算完成

### 6.9.3 基本配置

在使用硬件除法器之前，硬件除法器的时钟必须先使能。通过AHBCLK[4] 将HDIV\_EN设为1使能硬件除法器。

#### 6.9.4 功能描述

为了使用硬件除法器，需要先填被除数，然后再填除数，写除数之后硬件除法器将自动开始计算。计算结果包括商和余数可以从 DIVQUO 和 DIVREM 寄存器读到。如果在硬件除法器计算完成之前，CPU 读 DIVQUO 或者 DIVREM，CPU 将等待直到硬件除法器完成计算。因而，无需软件延迟，CPU 除法硬件除法器之后总是可以得到有效的结果。

如果除数是0，DIVSTS 寄存器的 DIV0 标志将被置位。

被除数是32-bit 有符号整数，被除数是16-bit 有符号数。商是32-bit 有符号整数，余数是16位有符号整数。

下图显示了硬件除法器的操作流程。为了计算  $X / Y$ ，CPU 需要写 X 到 被除数寄存器 DIVIDEND，然后写 Y 到除数寄存器 DIVISOR。写被除数寄存器之后，CPU 读 DIVQUO 和 DIVREM 寄存器可以得到计算结果。

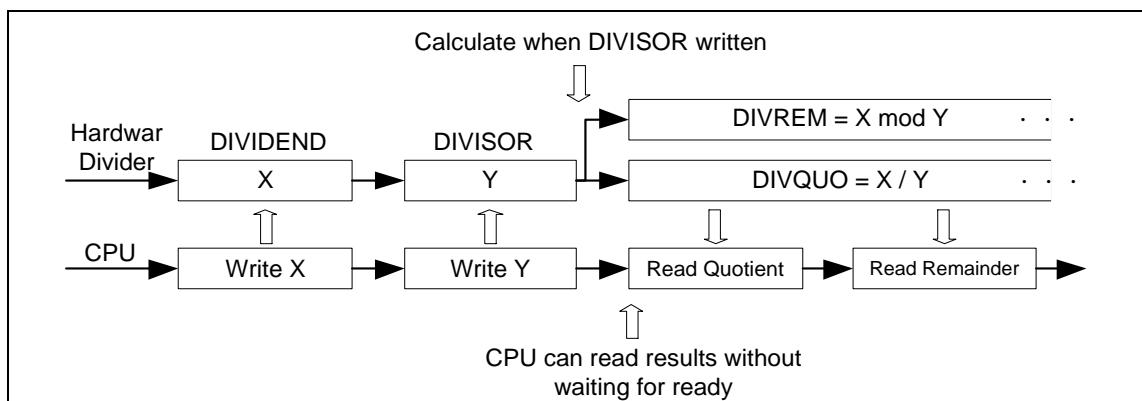


图 6-41 硬件除法器操作流程

### 6.9.5 寄存器映射

R: 只读, W: 只写, R/W: 可读写

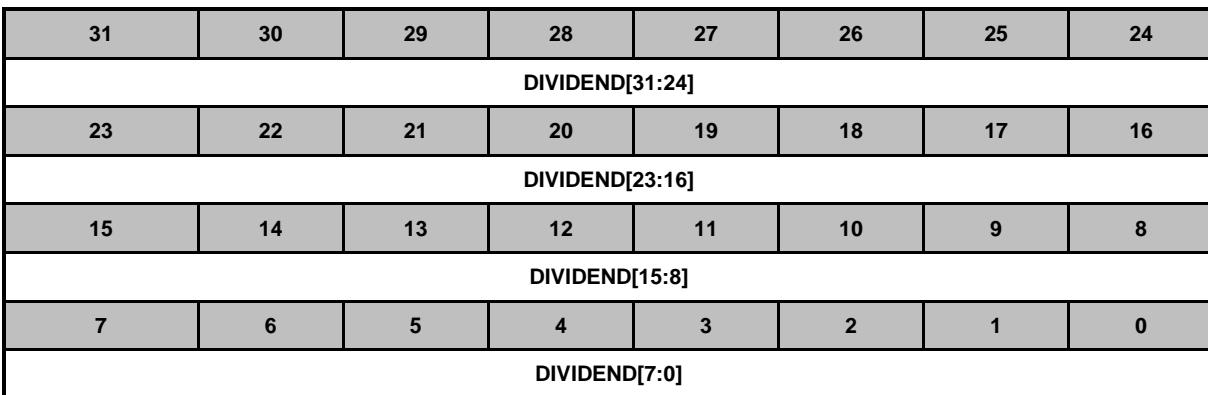
寄存器	偏移量	R/W	描述	复位后的值
<b>HDIV 基地址:</b>				
<b>HDIV_BA = 0x5001_4000</b>				
DIVIDEND	HDIV_BA+0x00	R/W	被除数源寄存器	0x0000_0000
DIVISOR	HDIV_BA+0x04	R/W	除数源寄存器	0x0000_FFFF
DIVQUO	HDIV_BA+0x08	R/W	商结果寄存器	0x0000_0000
DIVREM	HDIV_BA+0x0C	R/W	余数结果寄存器	0x0000_0000
DIVSTS	HDIV_BA+0x10	R	除数状态寄存器	0x0000_0001



### 6.9.6 寄存器描述

#### 被除数源寄存器(DIVIDEND)

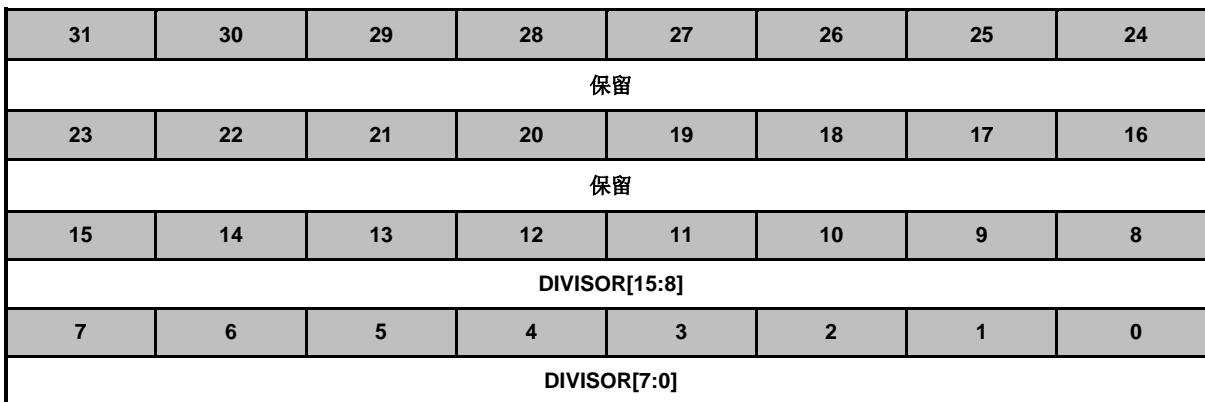
寄存器	偏移量	R/W	描述	复位后的值
DIVIDEND	HDIV_BA+0x00	R/W	被除数源寄存器	0x0000_0000



Bits	描述	
[31:0]	DIVIDEND	被除数 该寄存器用于存放被除数

**除数源寄存器 (DIVISOR)**

寄存器	偏移量	R/W	描述	复位后的值
DIVISOR	HDIV_BA+0x04	R/W	除数源寄存器	0x0000_FFFF

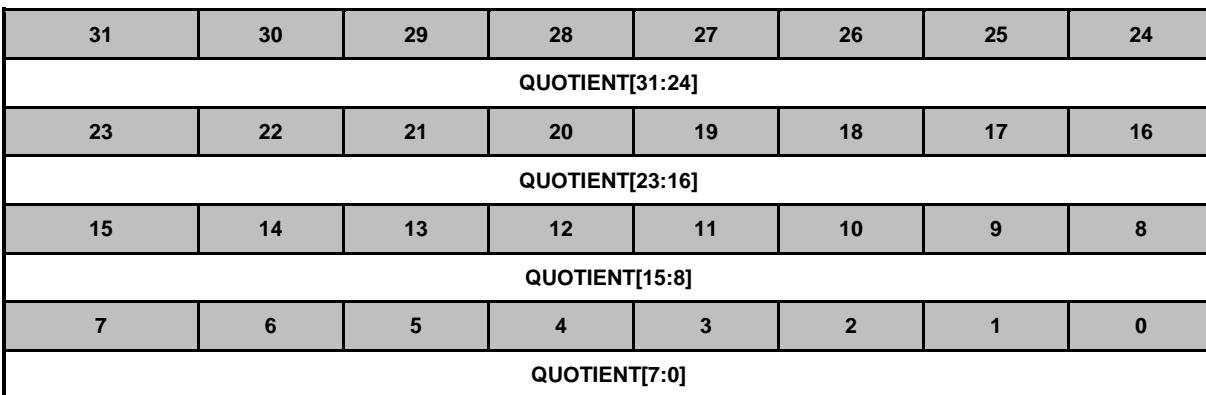


Bits	描述	
[31:16]	保留	保留.
[15:0]	DIVISOR	被除数 该寄存器用于存放被除数. 注: 该寄存器一旦被写, 硬件除法器马上开始计算.



### 商结果寄存器 (DIVQUO)

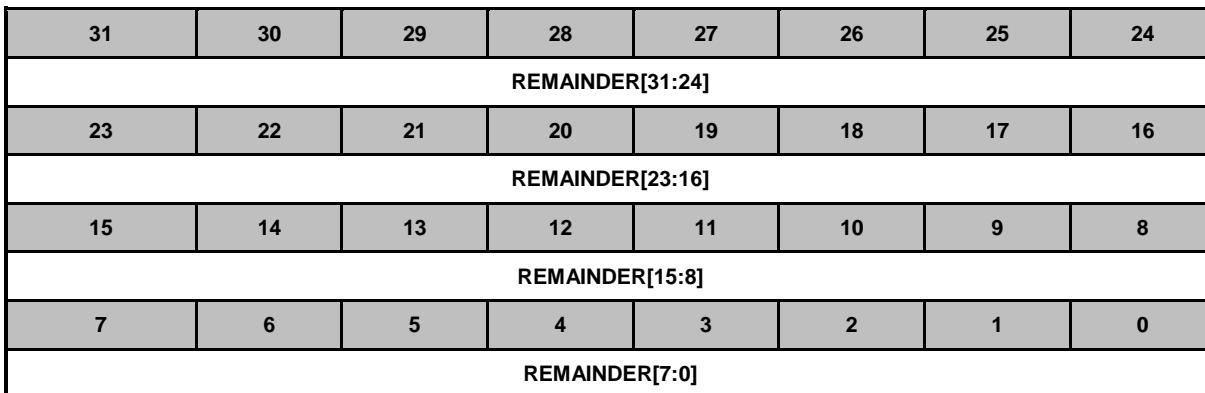
寄存器	偏移量	R/W	描述	复位后的值
DIVQUO	HDIV_BA+0x08	R/W	商结果寄存器	0x0000_0000



Bits	描述							
[31:0]	<b>QUOTIENT</b>	商结果 计算结束之后，该寄存器用于存放商.						

余数结果寄存器(DIVREM)

寄存器	偏移量	R/W	描述	复位后的值
DIVREM	HDIV_BA+0x0C	R/W	余数结果寄存器	0x0000_0000



Bits	描述	
[31:16]	REMAINDER[31:16]	REMAINDER[15:0]符号扩展 硬件除法器的余数是16位有符号数 (REMAINDER[15:0])，符号扩张 (REMAINDER[31:16]) 到32-bit整数.
[15:0]	REMAINDER[15:0]	余数 计算结束之后，该寄存器用于存放余数.

**除数状态寄存器 (DIVSTS)**

寄存器	偏移量	R/W	描述	复位后的值
DIVSTS	HDIV_BA+0x10	R	除数状态寄存器	0x0000_0001

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留						DIV0	保留

Bits	描述	
[31:2]	保留	保留.
[1]	DIV0	<b>除0警告</b> 0 = 除数不是0. 1 = 除数是0. <b>注:</b> DIV0 标志用来指示除0状态, 当DIVISOR寄存器被写该位马上被更新。该寄存器是只读的。
[0]	保留	保留.

## 6.10 I<sup>2</sup>C 总线控制器(主机/从机)

### 6.10.1 概述

I<sup>2</sup>C为2线，双向串行总线，为设备之间的数据通讯提供了简单有效的方法。I<sup>2</sup>C标准是多主机总线，包括冲突检测和仲裁机制以防止在两个或多个主机试图同时控制总线时发生数据冲突。

M05xxDN/DE系列有2组I<sup>2</sup>C提供睡眠唤醒功能。

### 6.10.2 特性

I<sup>2</sup>C总线通过两根线(SDA和SCL)在连接在总线上的设备间传输数据，总线的主要特征包括：

- 最多支持2组I<sup>2</sup>C端口
- 支持主机和从机模式
- 主从机之间双向数据传输
- 多主机总线(无中心主机)
- 多主机同时发送数据仲裁，总线上串行数据不会被损坏
- 串行时钟同步使得不同比特率的器件可以通过一条串行总线传输数据
- 串行时钟同步可用作握手方式来暂停和恢复串行传输
- 内建一个14位超时计数器，当I<sup>2</sup>C总线挂起并且计数器溢出时，该计数器将请求I<sup>2</sup>C中断
- 可编程的时钟适用于不同速率控制
- 支持7位寻址模式
- I<sup>2</sup>C总线控制器支持多地址识别(4组从机地址带屏蔽选项)
- 支持睡眠唤醒功能(M05xxDN/DE)

### 6.10.3 基本配置

$I^2C_0$ 配置如下:

- $I^2C_0$  引脚功能通过寄存器P3\_MFP [13:12] 配置.
- 通过寄存器 APBCLK [8] 使能 $I^2C_0$ 时钟( $I^2C_0\_EN$ ).
- 通过IPRSTC2 [8]寄存器( $I^2C_0\_RST$ )复位  $I^2C_0$  控制器.

$I^2C_1$ 配置如下:

- $I^2C_1$ 引脚功能通过寄存器P2\_MFP [5:4] 或者 P4\_MFP[5:4] 配置.
- 通过寄存器APBCLK [9] 使能 $I^2C_1$ 时钟( $I^2C_1\_EN$ ).
- 通过IPRSTC2 [9]寄存器( $I^2C_1\_RST$ )复位  $I^2C_1$  控制器.

### 6.10.4 功能描述

依靠SCL时钟同步，数据在主机与从机间在SDA数据线上一字节一字节的传输，每个字节为8位长度，一个SCL时钟脉冲传输一个数据位，MSB优先传输，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必须保持稳定。当SCL为高时，SDA线上的跳变视为一个命令（START 或 STOP），更多详细的I2C总线时序请参考下图。

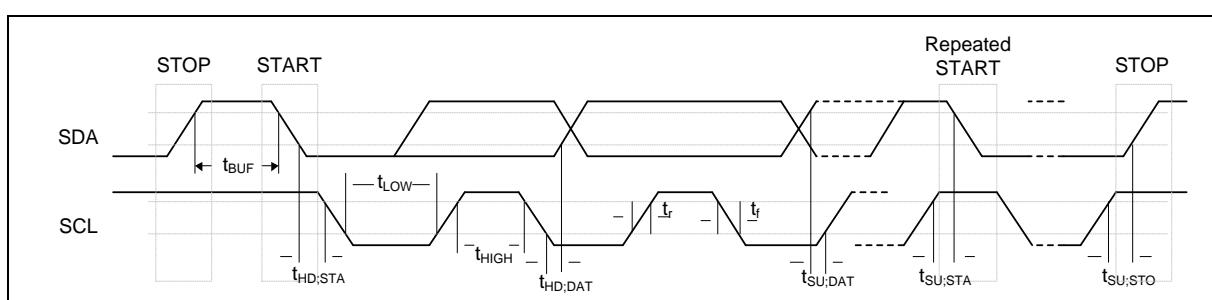


图 6-42 I2C 总线时序

该设备的片上I2C提供符合I2C总线标准模式规范的串行接口， $I^2C$ 端口自动处理字节传输。将I2CON的ENS1位设置为1，可以使能该端口。 $I^2C$  硬件接口通过两个引脚连接到 $I^2C$ 总线：SDA (串行数据线) 与 SCL (串行时钟线)。在作为 $I^2C$ 端口使用时，用户必须先将这两个引脚设置为 $I^2C$ 功能。

注：当SDA 与 SCL这两个引脚被配置为开漏模式时，引脚上需要加上拉电阻。

### 6.10.5 I<sup>2</sup>C协议

下图显示了典型的I<sup>2</sup>C协议，通常标准I<sup>2</sup>C传输包含四个部分：

- 1) 起始信号或重复起始信号的产生
- 2) 从机地址和R/W位传输
- 3) 数据传输
- 4) 停止信号的产生

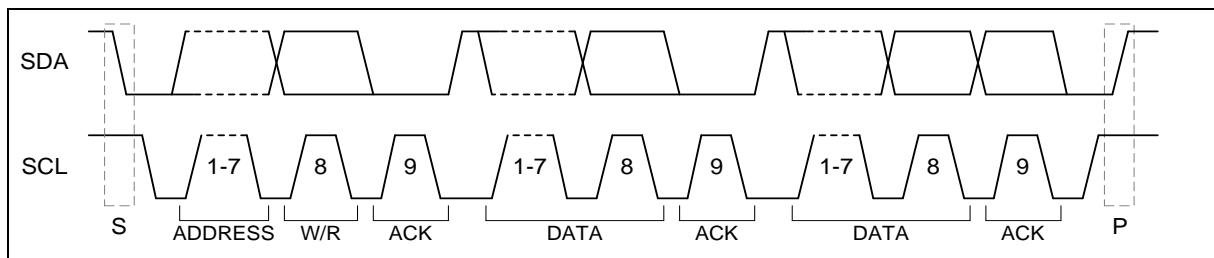


图 6-43 I<sup>2</sup>C 协议

#### 6.10.5.1 起始或重复起始信号

当总线处于空闲状态下，即没有任何主机设备占有总线(SCL 和SDA线同时为高)时，主机可以通过发送起始信号发起一次数据传输。起始信号，通常表示为**S-bit**，定义为当SCL线为高电平时，SDA线上产生一个高电平到低电平的跳变。起始信号表示新的数据传输的开始。

重复起始信号 (**Sr**) 是指在两个START信号间不存在STOP信号。主机用这种方式来和另外一个从机或同一个从机但是不同的传输方向的从机通讯，不用释放总线(例如：从写向一个设备到从该设备读取)。

#### 6.10.5.2 停止信号

主机可以通过产生一个停止信号来结束通讯。停止信号，通常用**P-bit**表示， 定义为当SCL线为高电平时，SDA线上产生一个低电平到高电平的跳变。

下图显示了起始信号、重复起始信号和停止信号

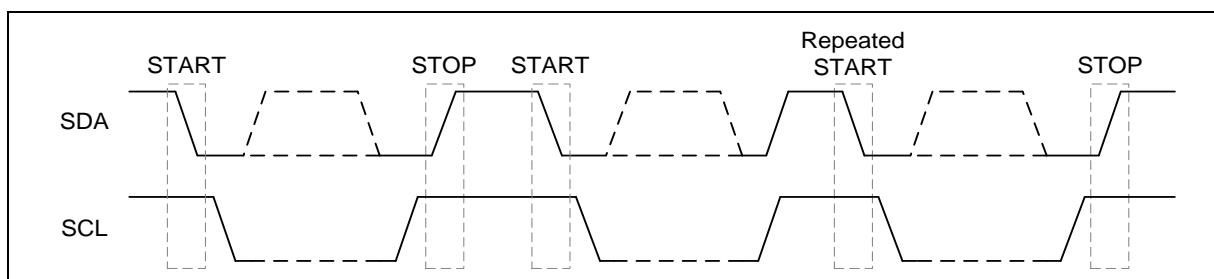


图 6-44 START 和 STOP 条件

### 6.10.5.3 从机地址传输

起始信号后传输的第一个字节就是从机地址(SLA)，从机地址的头7位是呼叫地址，紧跟7位地址后的是读/写(R/W)位。R/W位通知从机数据传输方向。系统当中不可以有两台从机有相同的地址。只有地址匹配的从机才会在SCL的第9个时钟周期拉低SDA作为应答信号来响应主机。

### 6.10.5.4 数据传输

当从机寻址成功完成，就可以根据主机发送的R/W位所决定的方向，开始一字节一字节的传输数据，每一个传输的字节会在第九个SCL时钟周期跟随一个应答位，如果从机上产生无应答信号(NACK)，主机可以产生一个停止信号来中止本次数据传输，或者产生重复起始信号开始新一轮的数据传输。

如果主机作为接收设备，没有应答(NACK)从机，则从机释放SDA线，以便于主机产生一个停止或重复起始信号。

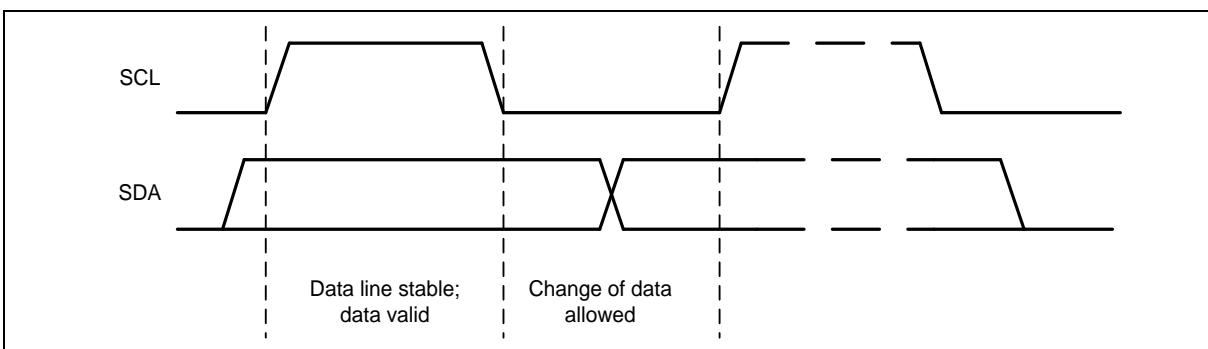


图 6-45 I<sup>2</sup>C 总线上的位传输

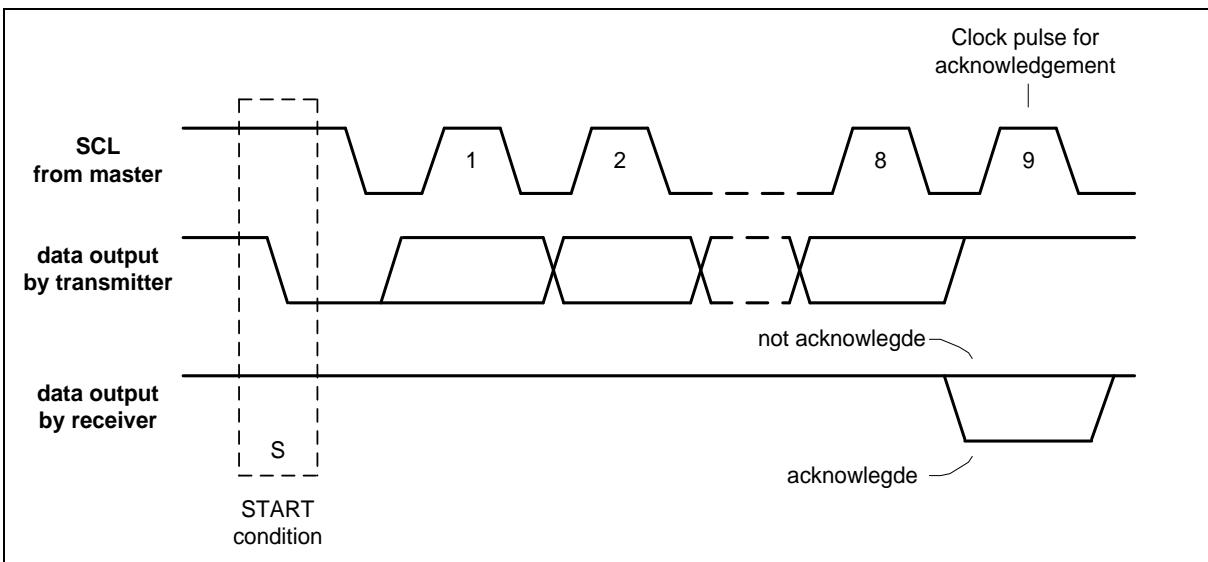


图 6-46 I<sup>2</sup>C 总线上的应答信号

### 6.10.5.5 I<sup>2</sup>C 总线上的数据传输

下图显示主机发送数据到从机。主机发送7位地址+1位写标志表示主机想发送数据到从机。

传输方向未改变

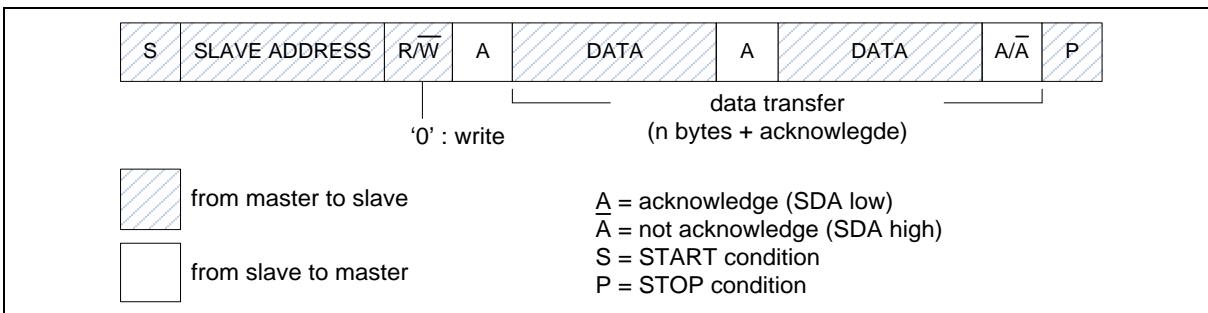


图 6-47 主机向从机传输数据

下图显示主机读取从机的数据。主机发送7位地址+1位读标志表示主机想从从机读数据。

传输方向改变

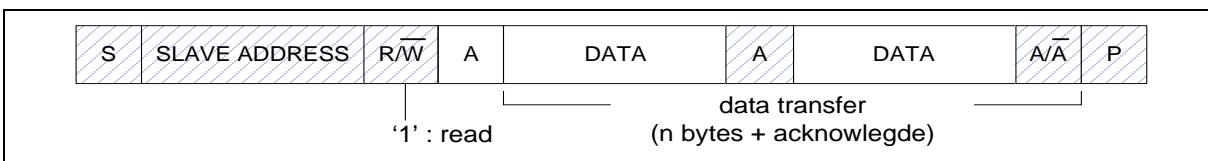


图 6-48 主机读取从机的数据

### 6.10.6 I<sup>2</sup>C 协议寄存器

CPU通过如下15个特殊功能寄存器控制I<sup>2</sup>C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 (地址屏蔽寄存器, n=0~3) I2CLK (时钟速率寄存器) , I2CTOC (超时计数器寄存器), I2CWKUPCON(唤醒控制寄存器) 和 I2CWKUPSTS(唤醒状态寄存器)。

#### 6.10.6.1 地址寄存器(I2CADDR)

I<sup>2</sup>C端口带有4个从机地址寄存器I2CADDRn (n=0~3)。当I<sup>2</sup>C处于主机模式时，这四个寄存器的值是无关的。在从机模式下，位域I2CADDRn [7:1]必须装入MCU自身从机地址，当I2CADDRn地址与接收的从机地址符合时，I<sup>2</sup>C硬件将应答。

I<sup>2</sup>C端口支持广播呼叫功能。当GC位(I2CADDRn [0])被置位，I<sup>2</sup>C端口硬件会响应广播呼叫地址(00H). 清GC位可禁止广播呼叫功能。

当GC位被置位，且I<sup>2</sup>C处于从机模式时，在主机发送广播呼叫地址到I<sup>2</sup>C总线上之后，I<sup>2</sup>C可以接收广播呼叫地址00H，然后它将跟随GC模式的状态。

#### 6.10.6.2 从机地址掩码寄存器(I2CADM)

I<sup>2</sup>C总线控制器支持多地址识别，带有4组地址屏蔽寄存器I2CADMn (n=0~3)。当地址屏蔽寄存器某一位置1，表示接收到的地址的相应位将被忽略。如果该位置0，表示接收到的地址的相应位应当与地址寄存器中相应位的值完全一致。

#### 6.10.6.3 数据寄存器(I2CDAT)

该寄存器存储的内容是准备发送的或刚接收的一个字节的数据。只要不在移位处理的过程中，CPU可以直接读写这8位I2CDAT [7:0]。当I<sup>2</sup>C处于定义的状态下，且串行中断标志(SI)被置位，只要SI位一直处于置位状态，I2CDAT[7:0]中的数据保持稳定。在数据被移出的过程中，总线上的数据同时被移入，I2CDAT[7:0]总是包含出现在总线上的最后一个字节数据。

应答位由I<sup>2</sup>C的硬件控制，CPU不能访问。串行数据在SCL线上串行时钟脉冲的上升沿被移入I2CDAT[7:0]。当一个字节被移入到I2CDAT [7:0]后，I2CDAT [7:0]中的串行数据是可以使用的，应答位(ACK或NACK)由控制逻辑在第9个时钟返回。为了在发送数据的时候监控总线状态，当发送I2CDATA[7:0]中数据到总线的时候，总线上的数据将同时被移入I2CDATA[7:0]中。串行数据在SCL时钟脉冲的下降沿从I2CDAT[7:0]被移出，在SCL时钟脉冲的上升沿被移入I2CDAT[7:0]。

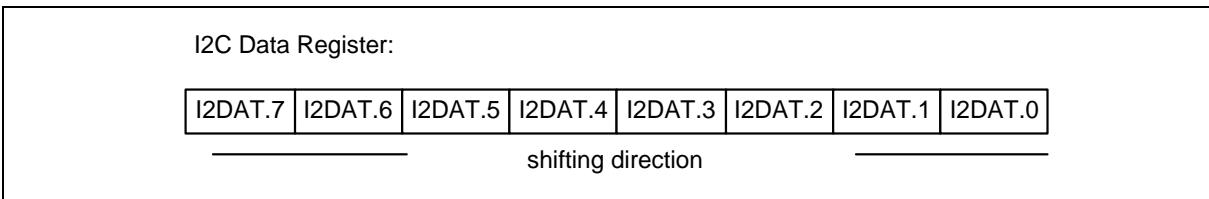


图 6-49 I<sup>2</sup>C 数据移动方向

#### 6.10.6.4 控制寄存器(I2CON)

CPU可以直接读写寄存器I2CON[7:0]。当I<sup>2</sup>C端口通过ENS1 (I2CON [6])等于1使能后，内部状态由I2CON 和 I<sup>2</sup>C逻辑硬件控制

I2CON有2位会受到硬件的影响：SI位在I<sup>2</sup>C硬件请求一个串行中断时被置位，STO位在总线上出现停止条件或者ENSI=0时被清除。

当有新的状态码产生并存储在I2CSTATUS寄存器中时，I<sup>2</sup>C中断标志位SI (I2CON [3])由硬件置位，如果使能中断位EI (I2CON [7])置位，则产生I<sup>2</sup>C中断请求。I2CSTATUS[7:0]中保存内部状态码，在SI被软件清除前，其内容保持不变

#### 6.10.6.5 状态寄存器(I2CSTATUS)

I2CSTATUS [7:0] 是一个8-位只读寄存器。低3位一直为0。位域I2CSTATUS[7:3]包含状态码。有26个可能的状态码。当I2CSTATUS [7:0]的内容是F8H时，没有串行中断请求。所有其它的I2CSTATUS [7:3]的值对应于定义的I<sup>2</sup>C状态。所有的状态在下表列出。如果I2CSTATUS [7:0]等于F8H，表示没有串行中断产生。所有其它的I2CSTATUS [7:0]值都对应一个已定义的I<sup>2</sup>C状态。当进入这些状态中的任一个，就会产生状态中断请求(SI = 1)。在SI被硬件置位1个机器周期后，有效状态码出现在I2CSTATUS [7:3]中，并保持稳定直至SI被软件清除。

另外，00H状态表示总线错误。总线错误在起始或停止信号出现在帧结构非法的位置时发生。总线错误可能在串行传输地址字节，数据字节或应答位期间发生。为了将I<sup>2</sup>C从总线错误中恢复，需要置位STO，清除SI从而进入没有寻址从机模式，然后清除STO释放总线并等待新的通信。I<sup>2</sup>C总线在总线错误时不能识别停止信号。

主模式		从模式	
状态	描述	状态	描述

0x08	起始信号发送成功	0xA0	GC或者从机接收/发送模式下，收到重复起始或者停止信号
0x10	重复起始信号发送成功	0xA8	从收到SLA+R返回ACK
0x18	主发送地址+W收到 ACK	0xB0	主模式时发送SLA+R/W仲裁失败时收到SLA+R返回ACK
0x20	主发送地址+W收到NACK	0xB8	从数据已发送收到 ACK
0x28	主数据已发送收到ACK	0xC0	从数据已发送收到NACK
0x30	主数据已发送收到 NACK	0xC8	最后一个字节已发送但是收到ACK
0x38	主仲裁失败	0x60	SLA+W已收到ACK返回
0x40	主发送地址+R收到 ACK	0x68	SLA+W已收到NACK返回
0x48	主发送地址+W收到 NACK	0x80	从收到数据ACK返回
0x50	主收到数据返回ACK	0x88	从收到数据NACK返回
0x58	主收到数据返回NACK	0x70	收到GC地址和1个或者多个数据字节，返回ACK
0x00	总线错误	0x78	主模式下SLA+R/W仲裁失败，收到GC地址，转到从模式，返回ACK
		0x90	GC接收模式下，收到数据返回 ACK
		0x98	GC接收模式下，收到数据返回NACK
0xF8	总线空闲  注：状态“0xF8”在主/从模式下都存在，不会导致中断。		

表6-12 I<sup>2</sup>C 状态码描述表

#### 6.10.6.6 I<sup>2</sup>C时钟波特率位 (I2CLK)

当I<sup>2</sup>C在主机模式下，I<sup>2</sup>C数据的波特率由I2CLK[7:0]寄存器决定。在从机模式下时I2CLK[7:0]的值不重要。在从机模式下，I<sup>2</sup>C将自动与主机I<sup>2</sup>C设备时钟频率同步。

I<sup>2</sup>C数据波特率设定公式：I<sup>2</sup>C的数据波特率 = (system clock) / (4x(I2CLK[7:0]+1))，如果system clock =16MHz, I2CLK[7:0]= 40(28H), I<sup>2</sup>C的数据波特率 = 16MHz /(4X (40 +1)) = 97.5K比特/秒。

#### 6.10.6.7 I<sup>2</sup>C超时计数寄存器 (I2CTOC)

有一个14位的超时计数器可以用于处理I<sup>2</sup>C总线挂起。当计数功能使能后，计数器开始计数直至发生超时，此时TIF置1，并产生I<sup>2</sup>C中断。可以通过将ENTI清除为0关闭计数功能。当超时计数器使能后，设定SI标志为高会复位计数器，清零SI之后计数器会重新开始计数。如果I<sup>2</sup>C总线挂起，会导致I2STATUS及SI标志在一段时间内不再更新。该14位超时计数器可能溢出并发出I<sup>2</sup>C中断请求。关于14位超时计数器参考下图，用户可通过对TIF位写1清0该标志。

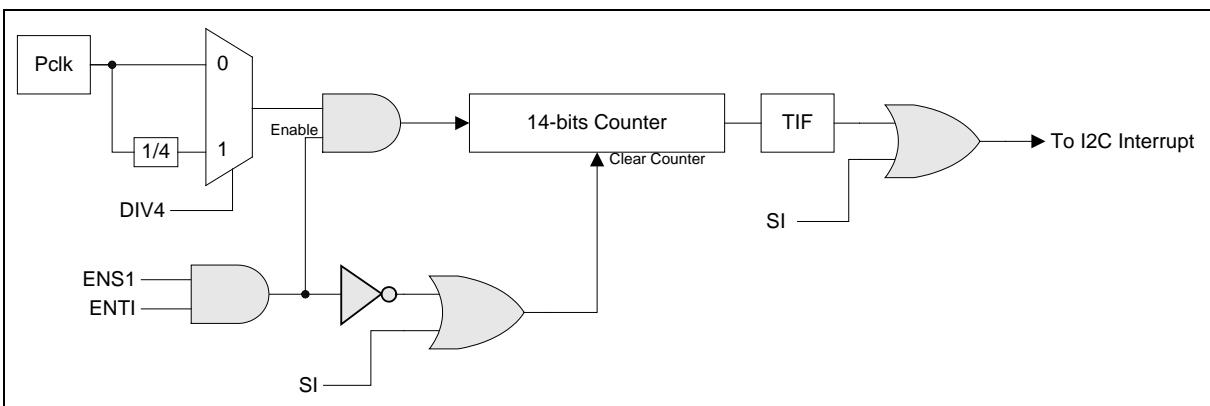


图 6-50 I<sup>2</sup>C 超时计数器框图

#### 6.10.6.8 I<sup>2</sup>C唤醒控制寄存器 (I2CWKUPCON)

当芯片进入睡眠模式时，其它的I<sup>2</sup>C主机可以通过寻址我们的I<sup>2</sup>C从机唤醒我们的芯片。进入睡眠模式之前，用户必须配置相应的寄存器。当芯片从地址匹配唤醒时(4个从机地址)，后面跟的数据将都被丢弃。需要再次收到START信号之后，后面的数据才会收下来。

#### 6.10.6.9 I<sup>2</sup>C唤醒状态寄存器(I2CWKUPSTS)

其它的I<sup>2</sup>C主机唤醒系统时，WKUPIF将被置1指示该事件。用户需要写‘1’清除该位。

### 6.10.7 操作模式

片上I<sup>2</sup>C端口支持3种操作模式：主机，从机和广播呼叫模式。

在实际应用中，I<sup>2</sup>C端口可以作为主机或从机。在从机模式，I<sup>2</sup>C端口寻找自身从机地址和广播呼叫地址，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。当微控制器希望成为总线主机时，在进入主机模式之前，硬件等待总线空闲以使可能的从机动作不会被打断。在主机模式下，如果总线仲裁失败，I<sup>2</sup>C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。

为了控制每个模式下I<sup>2</sup>C 总线传输，用户需要根据I2CSTATUS寄存器中的当前状态码设置I2CON和I2CDAT寄存器。换句话说，对于每个I<sup>2</sup>C总线动作，用户都需要检查I2CSTATUS寄存器中的当前状态，然后设置I2CON和 I2CDAT寄存器，进行下一次总线动作。

SI标志被清除以后，I2CON寄存器中的比特STA,STO 和AA将决定I2C硬件的下一个状态。新的动作完成时，一个新的状态码将被更新到I2CSTATUS寄存器中，SI将被置。如果I<sup>2</sup>C 中断控制位EI(I2CON [7])被置，状态码的处理以及下一个I<sup>2</sup>C硬件状态可以在中断处理函数中进行。

下图描述当前I<sup>2</sup>C状态码是0x08，然后设置I2CDATA=SLA+W并且(STA,STO,SI,AA) = (0,0,1,x)发送地址到I<sup>2</sup>C总线上。如果某个从机地址匹配，将响应ACK，然后I2CSTATUS寄存器的值将变为0x18

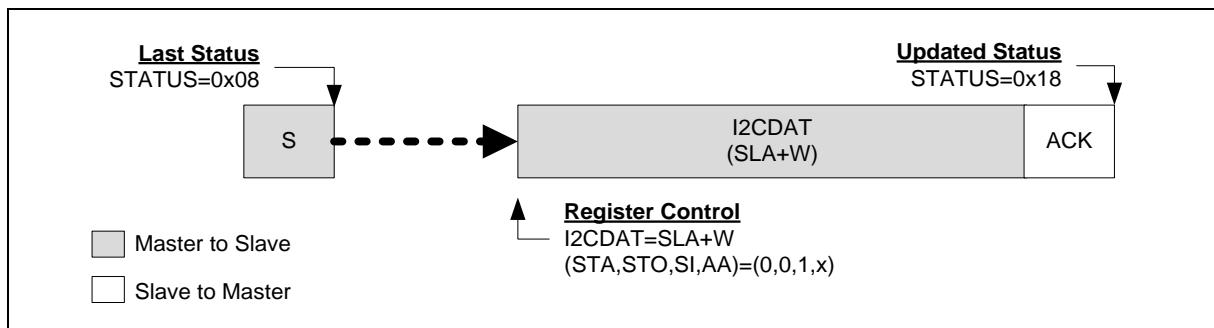


图6-51 根据当前I<sup>2</sup>C状态码控制I<sup>2</sup>C总线

### 6.10.7.1 主机模式

下图中所有可能的I<sup>2</sup>C主机协议都显示出来了，用户需要遵守合适的路径来实现I<sup>2</sup>C协议

用户发送START信号，当START信号被成功发送之后，芯片就进入主发送(图 6-52)或者主接收(图 6-53)模式，状态码就变成0x08。START信号之后，用户可以发送从机地址，读/写位，数据和重复START信号，STOP信号。

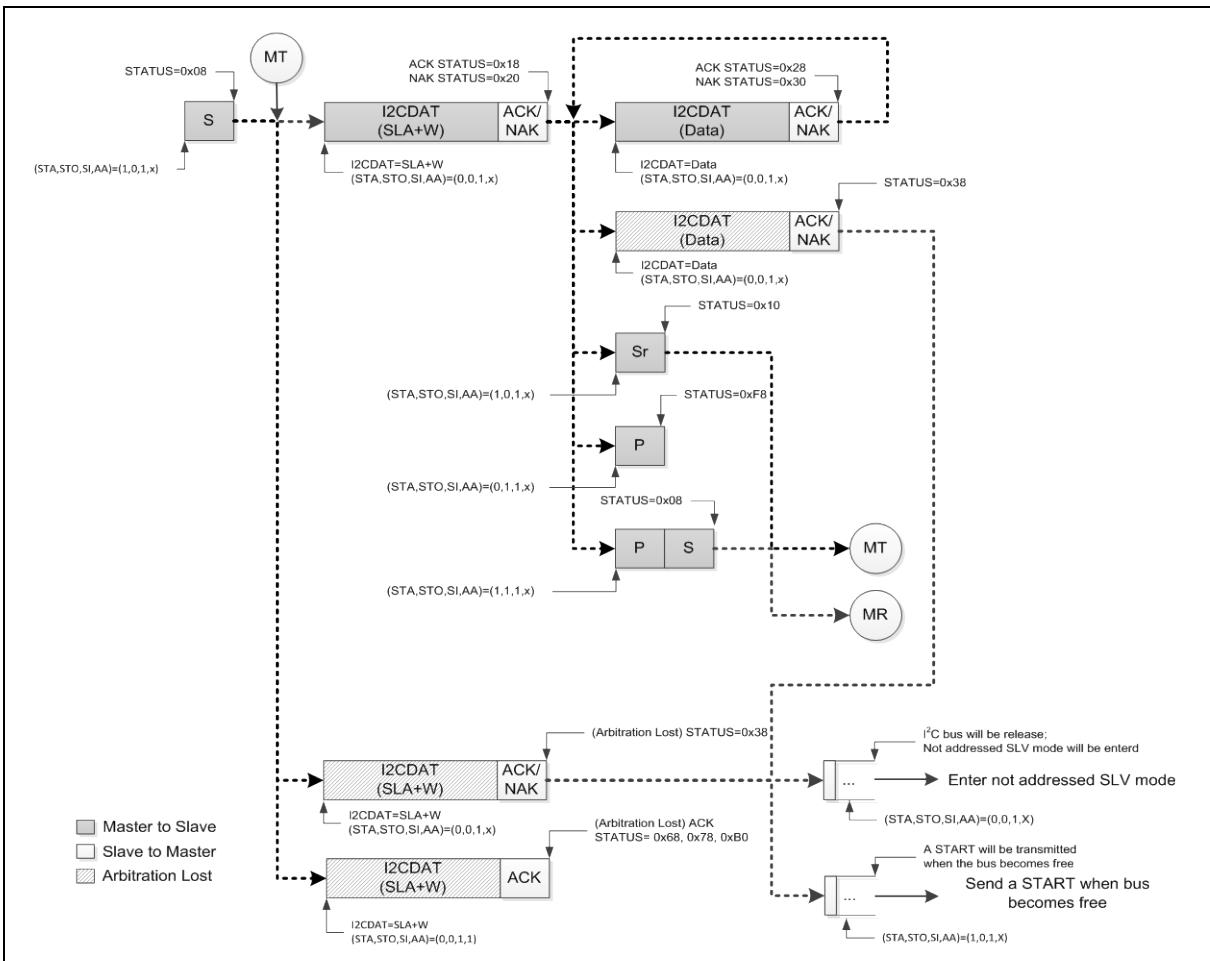


图 6-52 主机发送模式

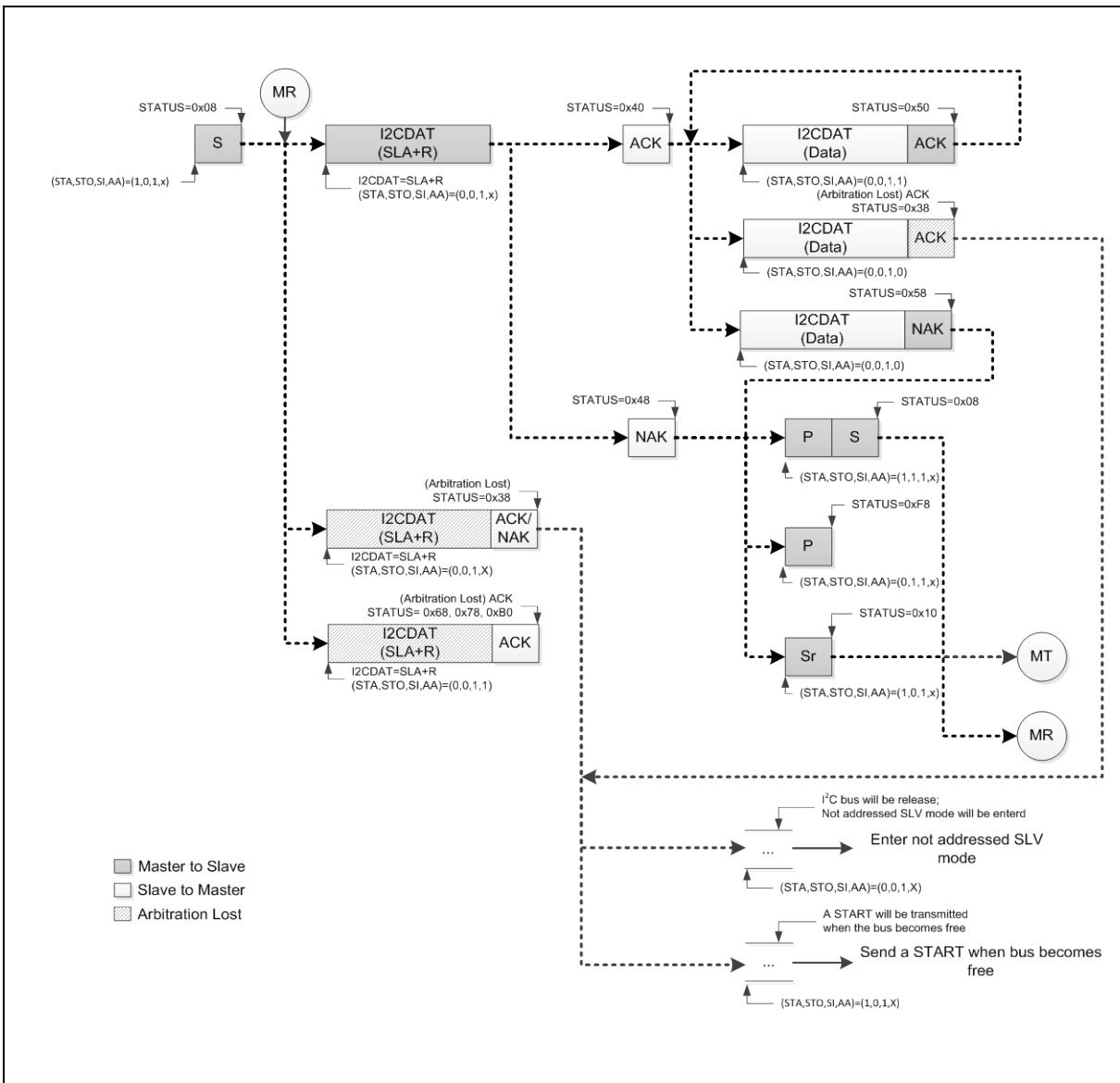


图 6-53 主机接收模式

如果I<sup>2</sup>C在主机模式，发生了仲裁失败，状态码为0x38。此时用户可以设置(STA, STO, SI, AA) = (1, 0, 1, X)来重新发送START信号，或者可以设置(STA, STO, SI, AA) = (0, 0, 1, X)释放I<sup>2</sup>C总线进入没有寻址从机模式

### 6.10.7.2 从机模式

系统复位之后默认I<sup>2</sup>C为未寻址的并且不会识别总线上的地址。用户需要写I2CADDRx寄存器设置从机地址并且设置(STA, STO, SI, AA) = (0, 0, 1, 1)让I<sup>2</sup>C识别主机发送的地址。图6-53显示了I<sup>2</sup>C作为从机时所有可能的流程，用户需要遵守合适的流程(图 6-53)来实现自己的I<sup>2</sup>C协议

主模式下如果总线仲裁失败，I<sup>2</sup>C端口将立即切换到从机模式，同时侦测本次串行传输总线上的从机地址。如果侦测到SLA+W(主机写)，状态码为0x68；如果侦测SLA+R(主机读)，状态码为0xB0。

**注：**I<sup>2</sup>C通讯期间，从模式下，当写1清除SI位时，SCL时钟才会被释放。

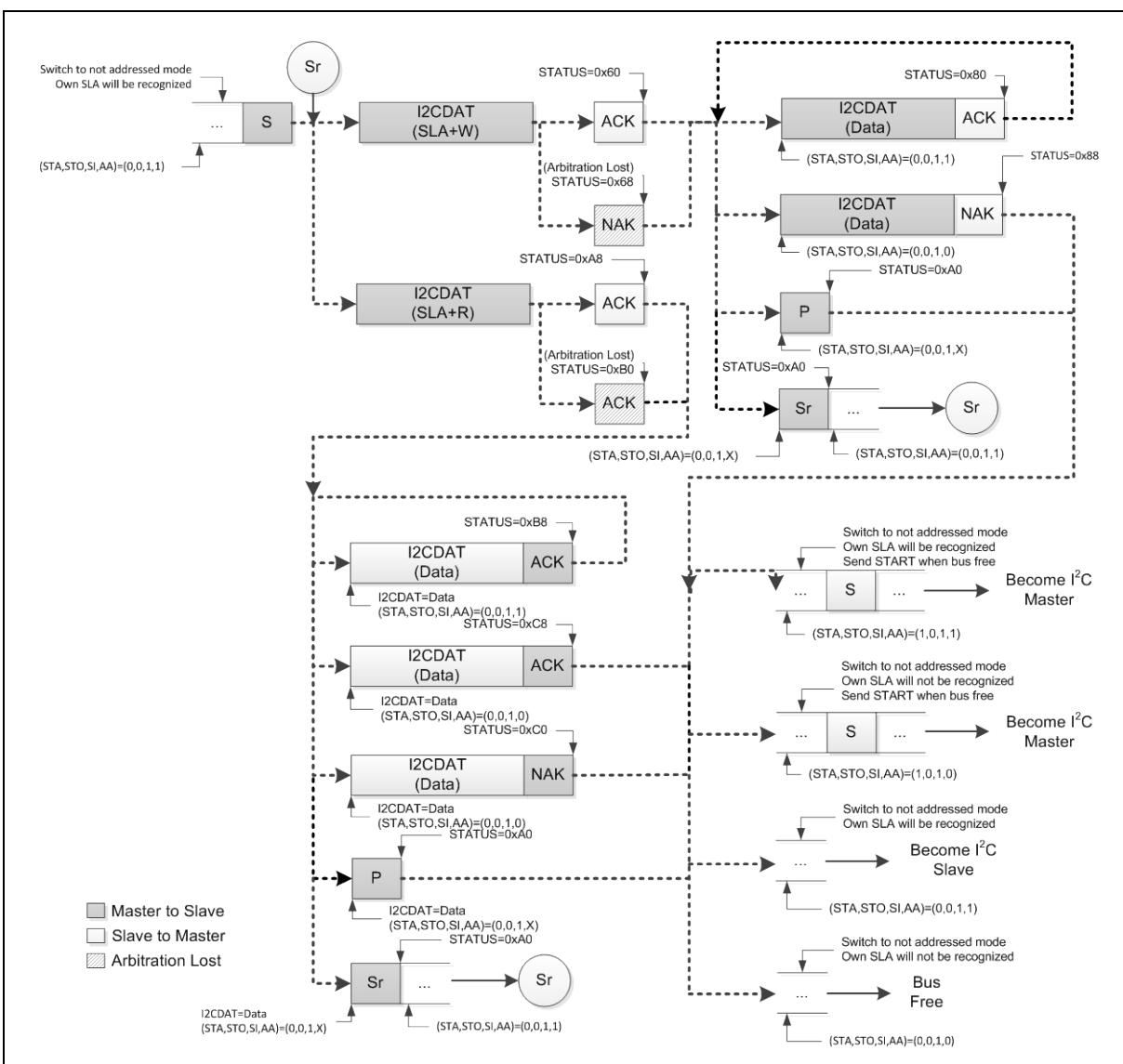


图 6-54 从机模式控制流程

如果I<sup>2</sup>C在已寻址从机模式下还在接收数据，但是收到了STOP或者重复START信号，状态码将变成0xA0，用户应该遵守上图0x88状态码的行为来处理。

如果I<sup>2</sup>C在已寻址从机模式下还在发送数据，但是收到了STOP或者重复START信号，状态码将变成0xA0，用户应该遵守上图0xC8状态码的行为来处理。

注：从机得到0x88、0xC8、0xC0和0xA0状态码之后，从机会切到未寻址模式，并且不识别自身地址。如果进入这个状态，从机不接收来自主机的任何I<sup>2</sup>C信号和地址。I<sup>2</sup>C控制器应该被复位以离开此状态。

#### 6.10.7.3 广播呼叫I (GC) 模式

如果 GC bit (I2CADDRn [0]) 被设，I<sup>2</sup>C 端口硬件将响应广播呼叫地址 (00H)。用户可以关闭GC位来禁止广播呼叫功能。当GC bit被设并且I<sup>2</sup>C在从模式时，当主机发送广播呼叫地址到I<sup>2</sup>C总线上时，可以接收广播呼叫地址00H，然后遵守广播呼叫模式的状态。

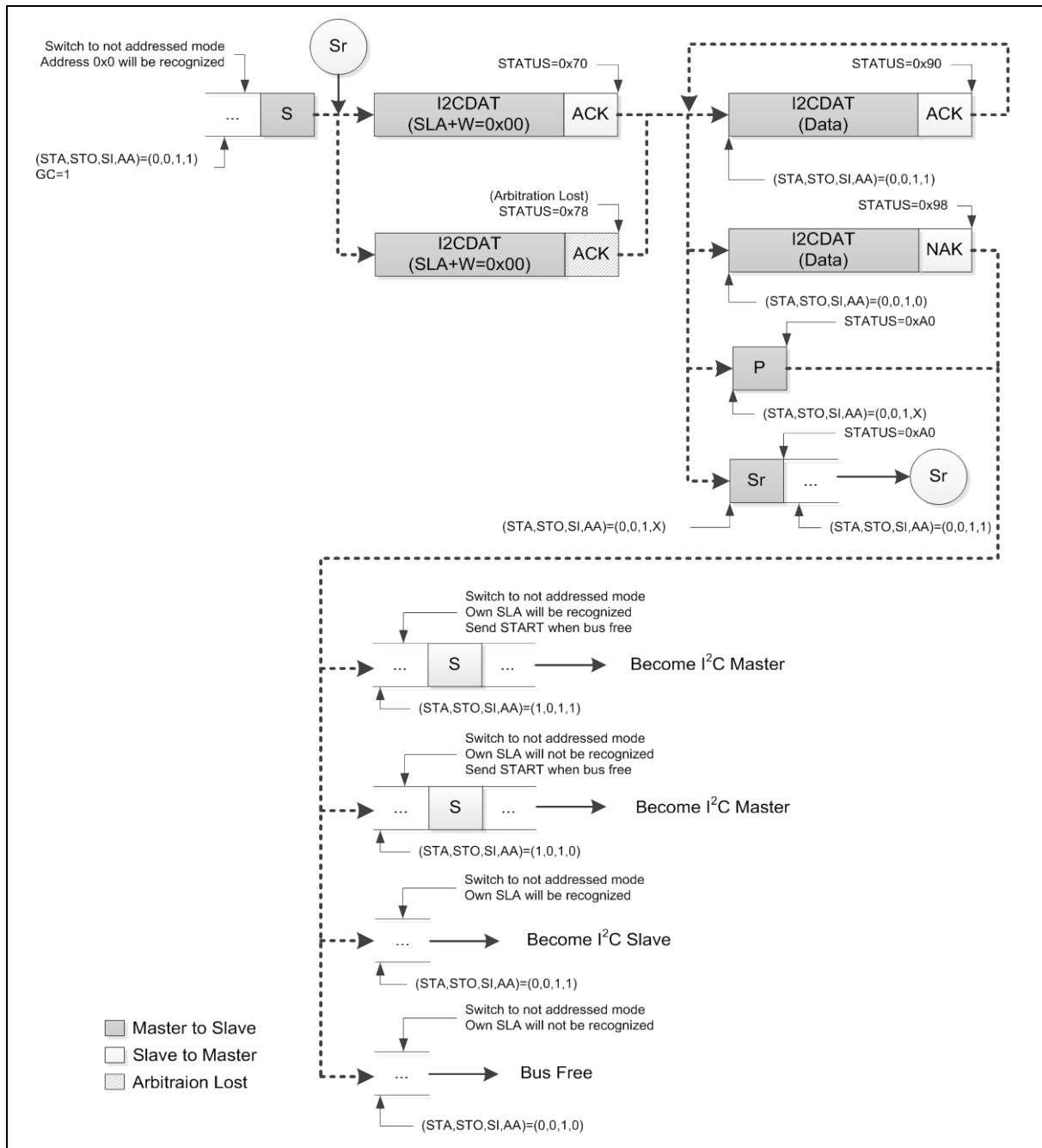


图6-55 广播呼叫模式

GC模式下，如果I<sup>2</sup>C还在接收数据但是收到了STOP或者重复START信号，状态码将变成0xA0。用户应该遵守上图状态码0x98来进行处理

**注：**从机得到0x98和0xA0状态码之后，从机会切到未寻址模式，并且不识别自身地址。如果进入这个状态，从机不接收来自主机的任何I<sup>2</sup>C信号和地址。I<sup>2</sup>C控制器应该被复位以离开此状态

#### 6.10.7.4 多主机

在某些应用中，I<sup>2</sup>C总线上同时有2个或者多个主机，主机可能同时发送数据。M05xxBN/DN/DE中的I<sup>2</sup>C支持多主机，包括冲突检测和仲裁来避免数据冲突。

- 当 I2CSTATUS = 0x38时，发生“仲裁失败”。仲裁失败事件可能在发送START信号、数据或者STOP信号时发生，用户应该设置 (STA, STO, SI, AA) = (1, 0, 1, X) 来再次发送 START 信号，或者设置(STA, STO, SI, AA) = (0, 0, 1, X) 来发送STOP 返回未寻址从机模式。
- 当I2CSTATUS = 0x00时，发生“总线错误”，为了将I<sup>2</sup>C从总线错误中恢复，STO应该被设并且SI 应该被清除，然后STO 被清除释放总线。
  - 设置 (STA, STO, SI, AA) = (0, 1, 1, X) 停止当前传输
  - 设置 (STA, STO, SI, AA) = (0, 0, 1, X) 释放总线

#### 6.10.7.5 随机读写EEPROM的例子

使用I<sup>2</sup>C 读EEPROM时，使用下列步骤配置I<sup>2</sup>C相关寄存器：

1. “P3\_MFP”和“ALT\_MFP”寄存器中设置多功能引脚当作SCL 和 SDA 引脚.
2. 使能 I<sup>2</sup>C APB 时钟：“APBCLK”寄存器中I2C\_EN=1.
3. 设置 I2C\_RST=1 复位 I<sup>2</sup>C 控制器，然后设置I2C\_RST=0使I<sup>2</sup>C 控制器到正常操作模式.
4. “I2CON”寄存器中设置ENS1=1 使能 I<sup>2</sup>C 控制器.
5. 设置“I2CLK”寄存器，设置合适的I<sup>2</sup>C 比特率.
6. “NVIC\_ISER”寄存器中设置 SETENA=0x00040000使能 I<sup>2</sup>C 中断.
7. “I2CON”寄存器中设置 EI=1 使能 I<sup>2</sup>C 中断.
8. 设置 I<sup>2</sup>C 地址寄存器：“I2CADDR0~I2CADDR3”.

随机读操作是EEPROM中的一个访问方法。该方法允许主机访问EEPROM 中的任何地址。下图显示了EEPROM 随机读操作的流程。

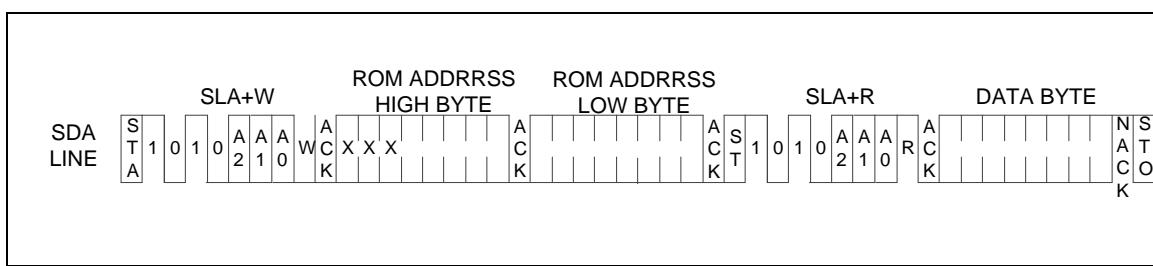


图 6-56 EEPROM 随机读操作

下图显示了怎样使用I<sup>2</sup>C控制器来实现 EEPROM 随机读协议.

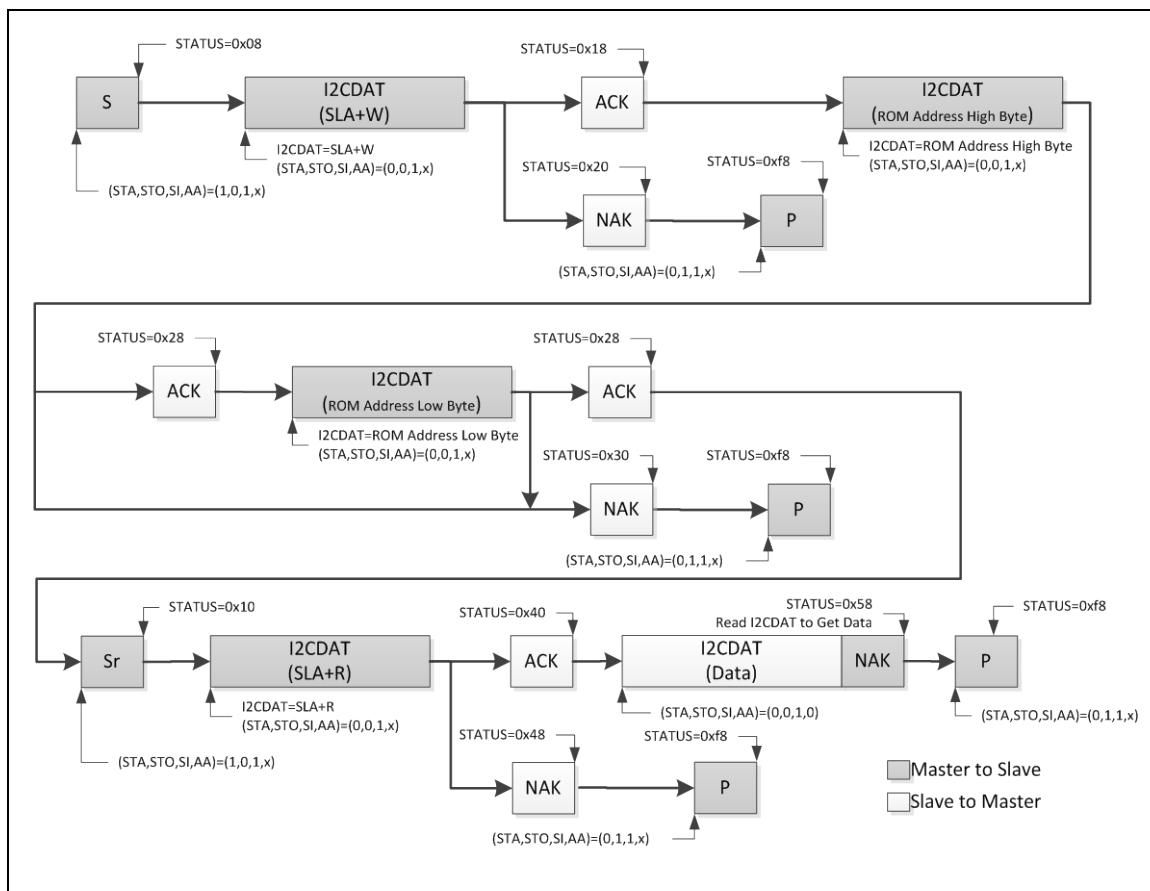


图 6-57 EEPROM 随机读协议

作为主机，I<sup>2</sup>C 控制器发送 START 信号到总线上，然后发送 SLA+W (从地址 + 写比特) 到 EEPROM，之后是2个字节的要读的EEPROM 数据的地址。最后，重复 START 信号和 SLA+R被发送，然后就可以读EEPROM的数据了。

### 6.10.8 寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>I<sup>2</sup>C基地址:</b>				
<b>I2C0_BA = 0x4002_0000</b>				
<b>I2C1_BA = 0x4012_0000 (M05xxDN/DE)</b>				
<b>I2CON</b>	I2Cx_BA+0x00	R/W	I <sup>2</sup> C控制寄存器	0x0000_0000
<b>I2CADRR0</b>	I2Cx_BA+0x04	R/W	I <sup>2</sup> C从机地址寄存器0	0x0000_0000
<b>I2CDAT</b>	I2Cx_BA+0x08	R/W	I <sup>2</sup> C数据寄存器	0x0000_0000
<b>I2CSTATUS</b>	I2Cx_BA+0x0C	R	I <sup>2</sup> C状态寄存器	0x0000_00F8
<b>I2CLK</b>	I2Cx_BA+0x10	R/W	I <sup>2</sup> C时钟分频寄存器	0x0000_0000
<b>I2CTOC</b>	I2Cx_BA+0x14	R/W	I <sup>2</sup> C超时控制寄存器	0x0000_0000
<b>I2CADDR1</b>	I2Cx_BA+0x18	R/W	I <sup>2</sup> C从机地址寄存器1	0x0000_0000
<b>I2CADDR2</b>	I2Cx_BA+0x1C	R/W	I <sup>2</sup> C从机地址寄存器2	0x0000_0000
<b>I2CADDR3</b>	I2Cx_BA+0x20	R/W	I <sup>2</sup> C从机地址寄存器3	0x0000_0000
<b>I2CADM0</b>	I2Cx_BA+0x24	R/W	I <sup>2</sup> C从机地址掩码寄存器0	0x0000_0000
<b>I2CADM1</b>	I2Cx_BA+0x28	R/W	I <sup>2</sup> C从机地址掩码寄存器1	0x0000_0000
<b>I2CADM2</b>	I2Cx_BA+0x2C	R/W	I <sup>2</sup> C从机地址掩码寄存器2	0x0000_0000
<b>I2CADM3</b>	I2Cx_BA+0x30	R/W	I <sup>2</sup> C从机地址掩码寄存器3	0x0000_0000
<b>I2CWKUPCON</b>	I2Cx_BA+0x3C	R/W	I <sup>2</sup> C唤醒控制寄存器(M05xxDN/DE)	0x0000_0000
<b>I2CWKUPSTS</b>	I2Cx_BA+0x40	R/W	I <sup>2</sup> C唤醒状态寄存器(M05xxDN/DE)	0x0000_0000

注: x=0,1

### 6.10.9 寄存器描述

#### I<sup>2</sup>C控制寄存器(I2CON)

寄存器	偏移量	R/W	描述	复位后的值
I2CON	I2Cx_BA+0x00	R/W	I <sup>2</sup> C控制寄存器	0x0000_0000

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

Bits	描述	
[31:8]	保留	保留
[7]	EI	<b>I<sup>2</sup>C中断使能控制</b> 1 = 使能I <sup>2</sup> C中断功能 0 = 禁用I <sup>2</sup> C中断功能
[6]	ENSI	<b>I<sup>2</sup>C控制器使能控制</b> 1 = 使能I <sup>2</sup> C控制器 0 = 禁用I <sup>2</sup> C控制器 当ENSI=1时, I <sup>2</sup> C串行功能使能, SDA和SCL管脚必须先设置为I <sup>2</sup> C功能
[5]	STA	<b>I<sup>2</sup>C起始控制位</b> STA置1, 进入主机模式, 如果总线处于空闲状态, I <sup>2</sup> C硬件会送出起始信号或重复起始信号。
[4]	STO	<b>I<sup>2</sup>C停止控制位</b> 在主机模式下, 置位STO将向总线传输停止条件, 然后I <sup>2</sup> C硬件会检查总线状态。一旦检测到停止条件, 该位将被硬件自动清零。在从机模式下, 置位STO会将I <sup>2</sup> C硬件复位至没有寻址从机模式。这意味着该设备不再处于从机接收模式, 不能从主发送设备接收数据。
[3]	SI	<b>I<sup>2</sup>C中断标志位</b> I2CSTATUS 寄存器哦那个有新的I <sup>2</sup> C状态时, 硬件置位SI标志。如果EI (I2CON [7])已经置位, 将发生I <sup>2</sup> C中断。SI 必须由软件清零。向该位写1清除为零。
[2]	AA	<b>接收应答控制位</b> 若在地址或数据接收之前, AA=1, 则在下列情况下: 1.从机应答主机发送的地址信息 2.接收设备应答发送设备发送的数据 时将在SCL时钟的应答时钟脉冲返回应答信号 (SDA为低); 若在地址或数据接收之前, AA=0, 则在SCL的应答时钟脉冲不会返回应答信号 (SDA为高)。从发送模式下, 该位为1表示还有数据等待发送; 该位为0表示已经是最后一个数据



		了，这时候从机应该收到NACK。
[1:0]	保留	保留

**I<sup>2</sup>C数据寄存器 (I2CDAT)**

寄存器	偏移量	R/W	描述	复位后的值
I2CDAT	I2Cx_BA+0x08	R/W	I <sup>2</sup> C数据寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
I2CDAT[7:0]							

Bits	描述	
[31:8]	保留	保留
[7:0]	I2CDAT	I <sup>2</sup> C 数据寄存器 Bit[7:0] 为8位I <sup>2</sup> C串行端口的传输数据.

**I<sup>2</sup>C状态寄存器 (I2CSTATUS)**

寄存器	偏移量	R/W	描述	复位后的值
I2CSTATUS	I2Cx_BA+0x0C	R/W	I <sup>2</sup> C状态寄存器	0x0000_00F8

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
I2CSTATUS[7:0]							

Bits	描述	
[31:8]	保留	保留
[7:0]	I2CSTATUS	<p><b>I<sup>2</sup>C状态寄存器</b></p> <p>低三位始终是0；高5位包含状态码。总共有26种状态码；当I2CSTATUS的值是F8H，不会产生中断请求；其它的所有的I2CSTATUS值都对应定义的I<sup>2</sup>C的状态。当进入这些状态时会产生一个状态中断请求(SI=1)。一个有效的状态码在SI被硬件设为'1'后一个周期内会出现在I2CSTATUS中，并保持稳定至SI被软件清零的下一个周期。另外，状态码是00H时表示总线错。当‘起始’或‘结束’信号出现在不正确的帧位置时会产生总线错误。比如在串行传输地址字节、数据字节或者应答位时都可能出现总线错误。</p>

**I<sup>2</sup>C时钟分频寄存器(I2CLK)**

寄存器	偏移量	R/W	描述	复位后的值
I2CLK	I2Cx_BA+0x10	R/W	I <sup>2</sup> C时钟分频寄存器	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
I2CLK[7:0]							

Bits	描述	
[31:8]	保留	保留
[7:0]	I2CLK	I <sup>2</sup> C时钟分频寄存器 I <sup>2</sup> C时钟比特率 = PCLK /(4x(I2CLK+1)). 注：I2CLK的最小值是4

I<sup>2</sup>C超时控制寄存器 (I2CTOC)

寄存器	偏移量	R/W	描述	复位后的值
I2CTOC	I2Cx_BA+0x14	R/W	I <sup>2</sup> C超时计数寄存器	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					ENTI	DIV4	TIF

Bits	描述	
[31:3]	保留	保留
[2]	ENTI	<p><b>超时计数使能控制</b>            1 = 使能超时计数器            0 = 禁用超时计数器</p> <p><b>注:</b> 当14位超时计数器被使能时, SI被清0后, 14位超时计数器开始计数。硬件将SI置1会使计数器复位, 在SI被软件清零后计数器又重新开始计数</p>
[1]	DIV4	<p><b>超时计数输入时钟除4</b>            1 = 使能超时计数输入时钟除4            0 = 禁用超时计数输入时钟除4</p> <p><b>注:</b> 使能后, 超时时间延长4倍。</p>
[0]	TIF	<p><b>超时标志</b>            当I<sup>2</sup>C超时发生时, 这个位由硬件置位, 如果I<sup>2</sup>C中断同时使能(EI=1), CPU将发生中断</p> <p><b>注:</b> 软件写"1"清零.</p>

**I<sup>2</sup>C从机地址寄存器(I2CADDRx)**

寄存器	偏移量	R/W	描述	复位后的值
I2CADDR0	I2Cx_BA+0x04	R/W	I <sup>2</sup> C从机地址寄存器0	0x0000_0000
I2CADDR1	I2Cx_BA+0x18	R/W	I <sup>2</sup> C从机地址寄存器1	0x0000_0000
I2CADDR2	I2Cx_BA+0x1C	R/W	I <sup>2</sup> C从机地址寄存器2	0x0000_0000
I2CADDR3	I2Cx_BA+0x20	R/W	I <sup>2</sup> C从机地址寄存器3	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
I2CADDR[7:1]							GC

Bits	描述	
[31:8]	保留	保留
[7:1]	I2CADDR	<b>I<sup>2</sup>C地址寄存器</b> 主机模式下，该寄存器的值无效。从机模式下，高七位为MCU自身地址，如果地址匹配，I <sup>2</sup> C硬件会响应。
[0]	GC	<b>广播呼叫功能使能控制</b> 0 = 禁用广播呼叫功能。 1 = 使能广播呼叫功能

**I<sup>2</sup>C从机地址掩码寄存器(I2CADMx)**

寄存器	偏移量	R/W	描述	复位后的值
I2CADM0	I2Cx_BA+0x24	R/W	I <sup>2</sup> C从机地址掩码寄存器0	0x0000_0000
I2CADM1	I2Cx_BA+0x28	R/W	I <sup>2</sup> C从机地址掩码寄存器1	0x0000_0000
I2CADM2	I2Cx_BA+0x2C	R/W	I <sup>2</sup> C从机地址掩码寄存器2	0x0000_0000
I2CADM3	I2Cx_BA+0x30	R/W	I <sup>2</sup> C从机地址掩码寄存器3	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
I2CADMx[7:1]							保留

Bits	描述	
[31:8]	保留	保留
[7:1]	I2CADMx	<b>I<sup>2</sup>C地址掩码</b> 1 = 掩码使能(不关心收到的相应地址位) 0 = 掩码禁止(接收到的地址相应位必须完全符合正确的地址内容)
[0]	保留	保留

I<sup>2</sup>C 唤醒控制寄存器 (I2CWKUPCON) (M05xxDN/DE)

寄存器	偏移量	R/W	描述	复位后的值
I2CWKUPCON	I2Cx_BA+0x3C	R/W	I <sup>2</sup> C 唤醒控制寄存器 (M05xxDN/DE)	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							WKUPEN

Bits	描述	
[31:1]	保留	保留.
[0]	WKUPEN	I <sup>2</sup> C 唤醒功能使能控制 1 = 使能 I <sup>2</sup> C 唤醒功能. 0 = 禁止 I <sup>2</sup> C 唤醒功能

I<sup>2</sup>C 唤醒状态寄存器 (I2CWKUPSTS) (M05xxDN/DE)

寄存器	偏移量	R/W	描述	复位后的值
I2CWKUPSTS	I2Cx_BA+0x40	R/W	I <sup>2</sup> C 唤醒状态寄存器 (M05xxDN/DE )	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							WKUPIF

Bits	描述	
[31:1]	保留	保留.
[0]	WKUPIF	I <sup>2</sup> C 唤醒中断标志 当芯片从睡眠模式由I <sup>2</sup> C唤醒时，该位被置为1.软件可以写1清除该位

## 6.11 PWM发生器和捕捉定时器(PWM)

### 6.11.1 概述

NuMicro™ M051 系列有 2 个 PWM 组，共有 4 组 PWM 发生器，可配置成 8 个独立的 PWM 输出，PWM0~PWM7，或者 4 个互补的 PWM 对，(PWM0, PWM1), (PWM2, PWM3), (PWM4, PWM5) 和 (PWM6, PWM7)，带 4 个可编程的死区发生器。

每组 PWM 发生器带有 8 位预分频器，一个时钟分频器提供 5 种分频(1, 1/2, 1/4, 1/8, 1/16)，两个 PWM 定时器包括 2 个时钟选择器，两个 16 位 PWM 计数器用于 PWM 周期控制，两个 16 位比较器用于 PWM 占空比控制以及一个死区发生器。4 组 PWM 发生器提供 8 个独立的 PWM 中断标志，当相应的 PWM 周期向下计数器达到零时这些中断标志由硬件置位。每个 PWM 中断源和它相应的中断使能位可以导致 PWM 发生中断。PWM 发生器可以配置为单触发模式产生仅仅一个 PWM 周期或自动重载模式连续输出 PWM 波形。

当 DZEN01(PC[4].)置位，PWM0 与 PWM1 实现互补的 PWM 对功能，这一对 PWM 的周期，占空比和死区时间由 PWM0 定时器和死区发生器 0 决定。同样，PWM 互补对(PWM2, PWM3), (PWM4, PWM5) 与 (PWM6, PWM7) 分别由 PWM2, PWM4 与 PWM6 的定时器和死区发生器 2, 4, 6 控制，PWM 定时器架构请参考下图。

为防止 PWM 输出不稳定波形，16 位向下计数器和 16 位比较器采用双缓存。当用户向计数器/比较器寄存器写入值的时候，只有当向下计数器的值达到 0 时，被更新的值才会被装载到 16 位计数器/比较器。该双缓冲特性避免 PWM 输出波形上产生毛刺。

当 16 位向下计数器达到 0 时，中断请求产生。如果 PWM 定时器被配置为自动重装载模式，当向下计数器达到 0 时，会自动重新装载 PWM 计数器寄存器(CNRx)的值，并开始递减计数，如此连续重复。如果定时器设为单触发模式，当向下计数器达到 0 时，向下计数器停止计数，并产生一个中断请求。

PWM 计数器比较器的值(CMRx)用于高电平脉冲宽度调制，当向下计数器的值与比较寄存器的值相同时，计数器控制逻辑反转输出为高电平。

PWM 定时器可复用为数字输入捕捉功能。如果捕捉功能使能，PWM 的输出引脚将被切换至捕捉输入模式。捕捉器 0 和 PWM0 使用同一个定时器，捕捉器 1 和 PWM1 使用另一组定时器，以此类推。因此在使用捕捉功能之前，用户必须预先配置 PWM 定时器。捕捉功能使能后，捕捉器在输入通道有上升沿跳变时，将 PWM 计数器的值锁存至捕捉上升沿锁存寄存器(CRLR)，在输入通道有下降沿跳变时将 PWM 计数器值锁存至捕捉下降沿锁存寄存器(CFLR)。捕捉通道 0 的中断是可编程的，通过设定.CRL\_IE0 (CCR0[1]) (上升沿锁存中断使能) 和 CFL\_IE0(CCR0[2]) (下降沿锁存中断使能) 来决定中断发生的条件。通过设置.CRL\_IE1(CCR0 [17]) 和.CRL\_IE1(CCR0[18])，捕捉通道 1 有同样的特性。通过设置 CCR2 中的相应的控制位，每组的通道 2 到通道 3 有同样的特性。对于每一组，不管捕捉何时产生中断 0/1/2/3，PWM 计数器 0/1/2/3 都将在该时刻重载。

最大的捕捉频率受捕捉中断延迟限制。捕捉中断发生时，软件至少要执行三个步骤：读 PIIRx 以得到中断源，读 CRLRx/CFLRx(x=0~3) 以得到捕捉值，最后写 1 清 PIIRx 为 0。如果中断延迟要花时间 T0 完成，在这段时间内(T0)，捕捉信号一定不能翻转。在这种情况下，最大的捕捉频率将是 1/T0。例如：

HCLK = 50 MHz, PWM\_CLK = 25 MHz, 中断延迟时间 900 ns

因此最大的捕捉频率将是  $1/900\text{ns} \approx 1000 \text{ kHz}$

### 6.11.2 特性

#### 6.11.2.1 PWM 功能特性:

每组 PWM 有两个 PWM 发生器。每个 PWM 发生器支持一个 8 位的预分频器，一个时钟分频器，两个 PWM 定时器（向下计数），一个死区发生器和两路 PWM 输出。

- 2个PWM组 (PWMA/PWMB), 支持8个PWM通道或者4个互补的PWM通道
- 每组PWM有2个PWM发生器，每个PWM发生器支持一个8位的预分频器，一个时钟分频器，两个PWM定时器（向下计数），一个死区发生器和两路PWM输出
- 最高16位分辨率
- 单触发模式或自动重载模式
- 支持边沿对齐或者中心对齐
- PWM触发ADC启动转换

#### 6.11.2.2 捕捉功能模块特性:

- 与PWM发生器共享时序控制逻辑
- 8路捕捉输入通道与8个PWM输出通道复用
- 每个通道支持一个上升沿锁存寄存器(CRLRx)，一个下降沿锁存寄存器(CFLRx)和捕捉中断标志(CAPIFx)

### 6.11.3 PWM 框图

下图按对说明PWM架构(例如：PWM定时器0/1为一对， 定时器2/3为另外一对， 等等)。

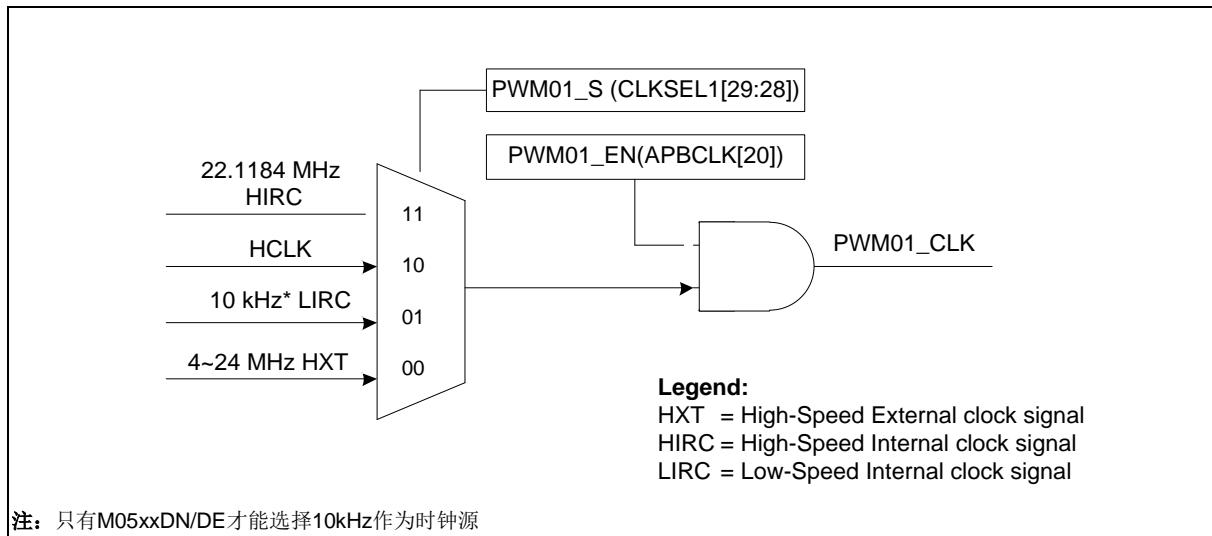


图 6-58 PWM 发生器 0 时钟源控制

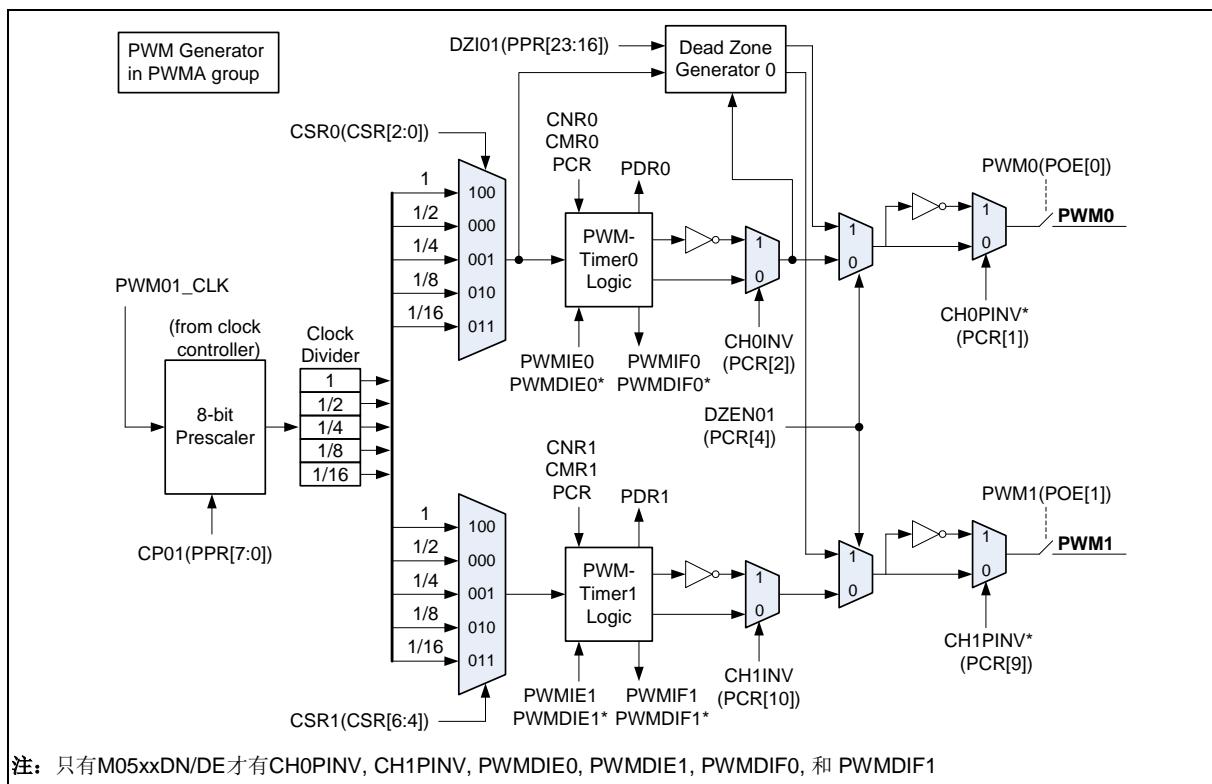


图 6-59 PWM 发生器 0 结构框图

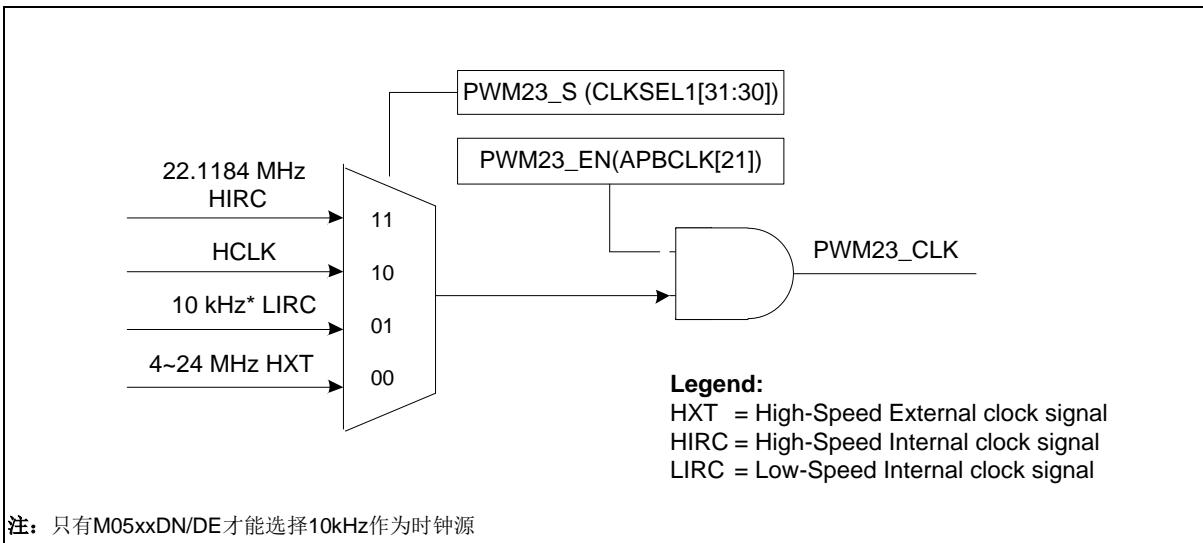


图 6-60 PWM 发生器 2 时钟源控制

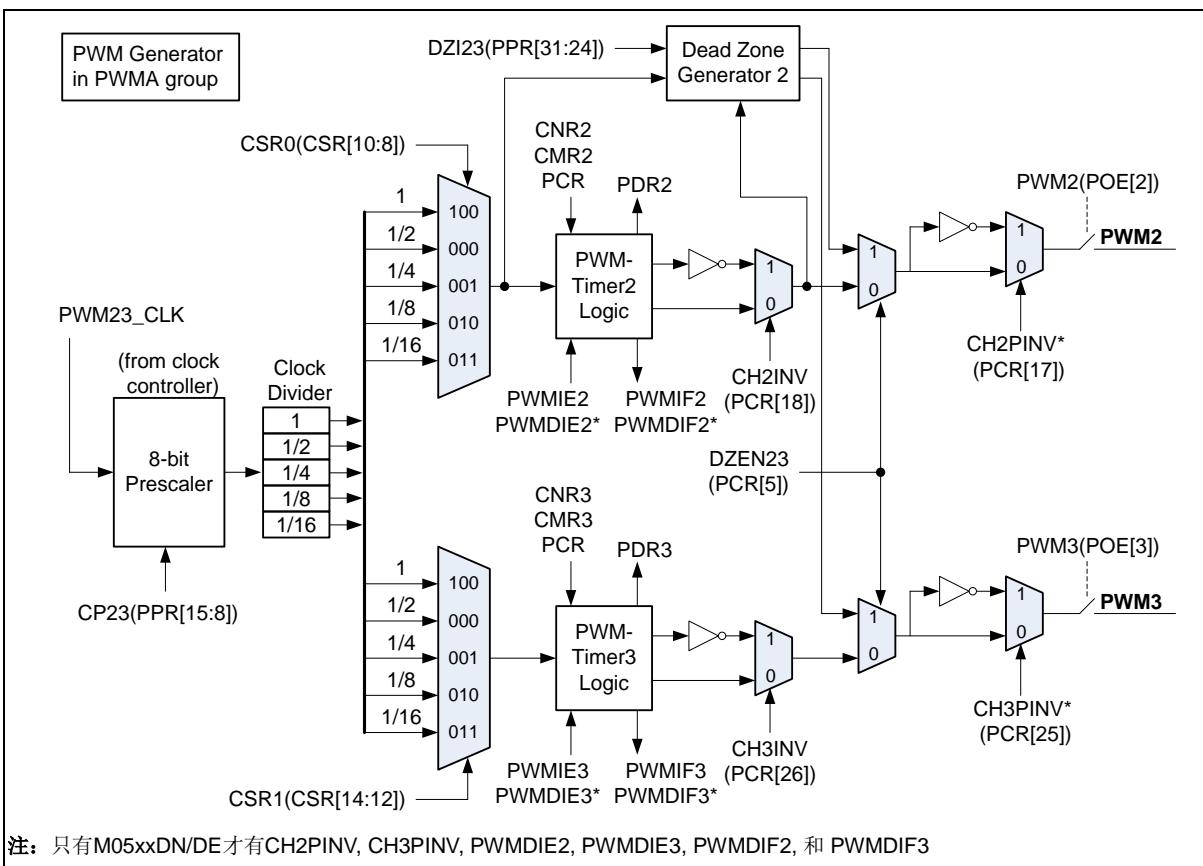


图 6-61 PWM 发生器 2 结构框图

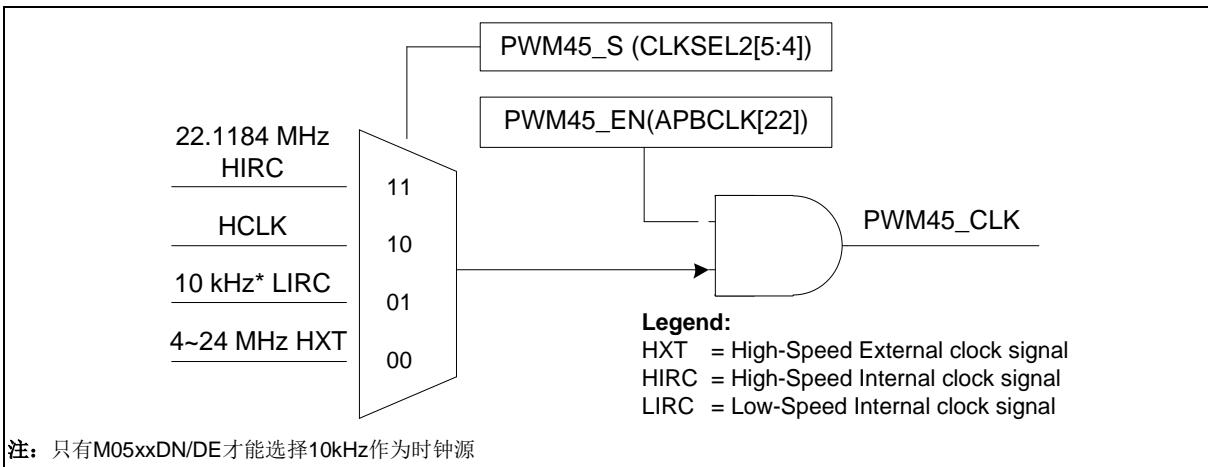


图 6-62 PWM 发生器 4 时钟源控制

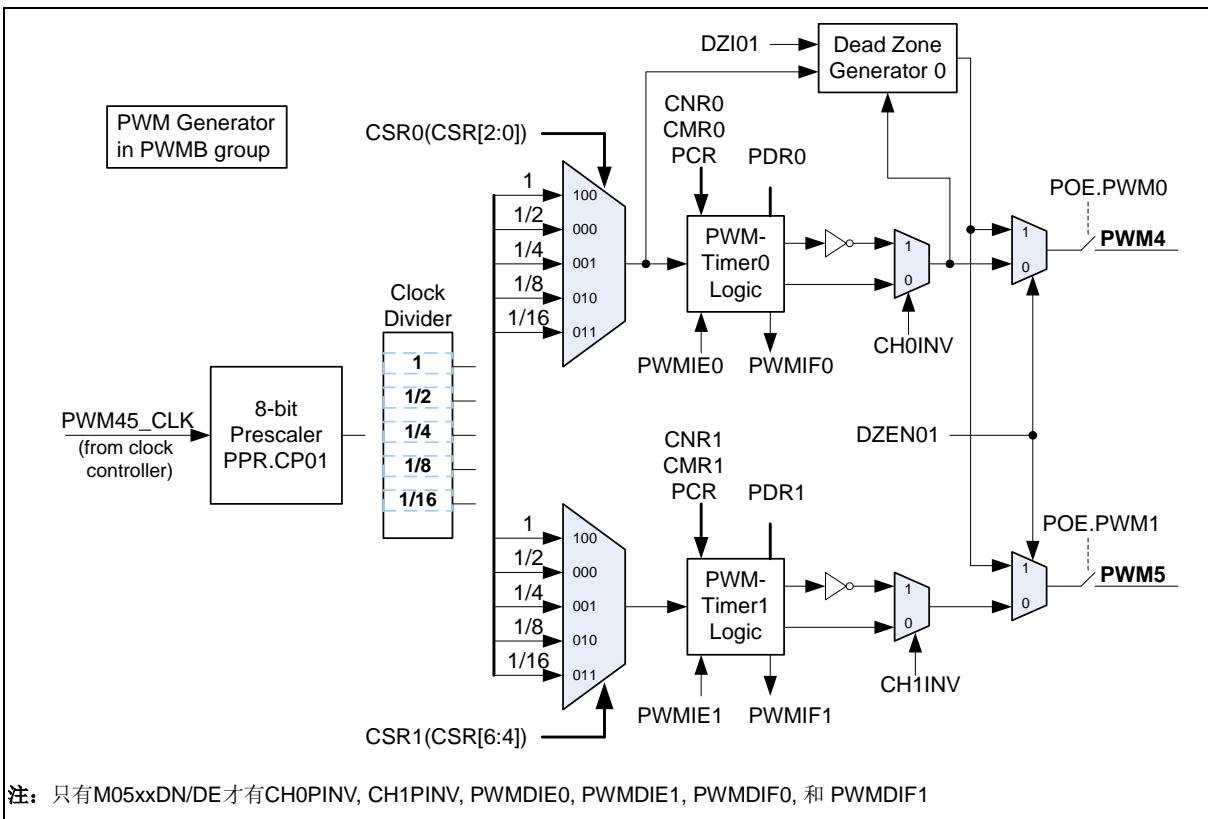


图 6-63 PWM 发生器 4 结构框图

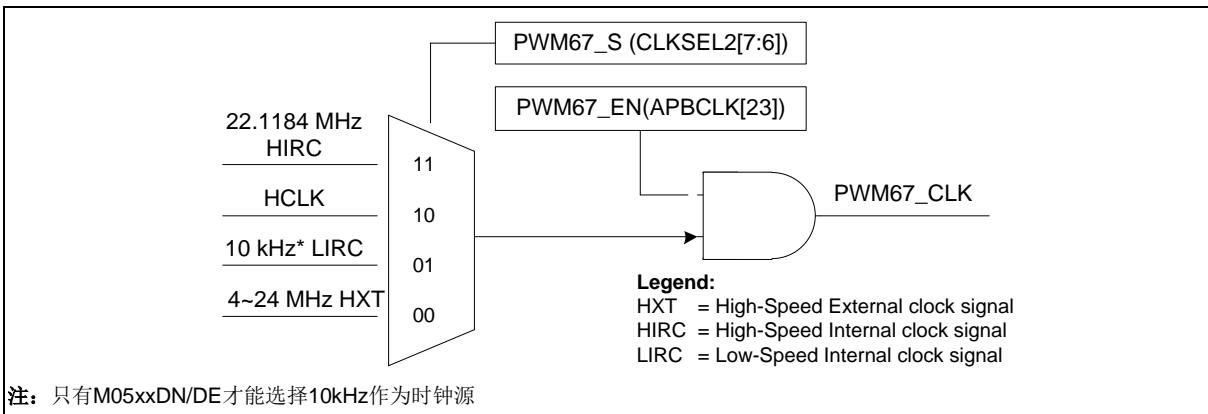


图 6-64 PWM 发生器 6 时钟源控制

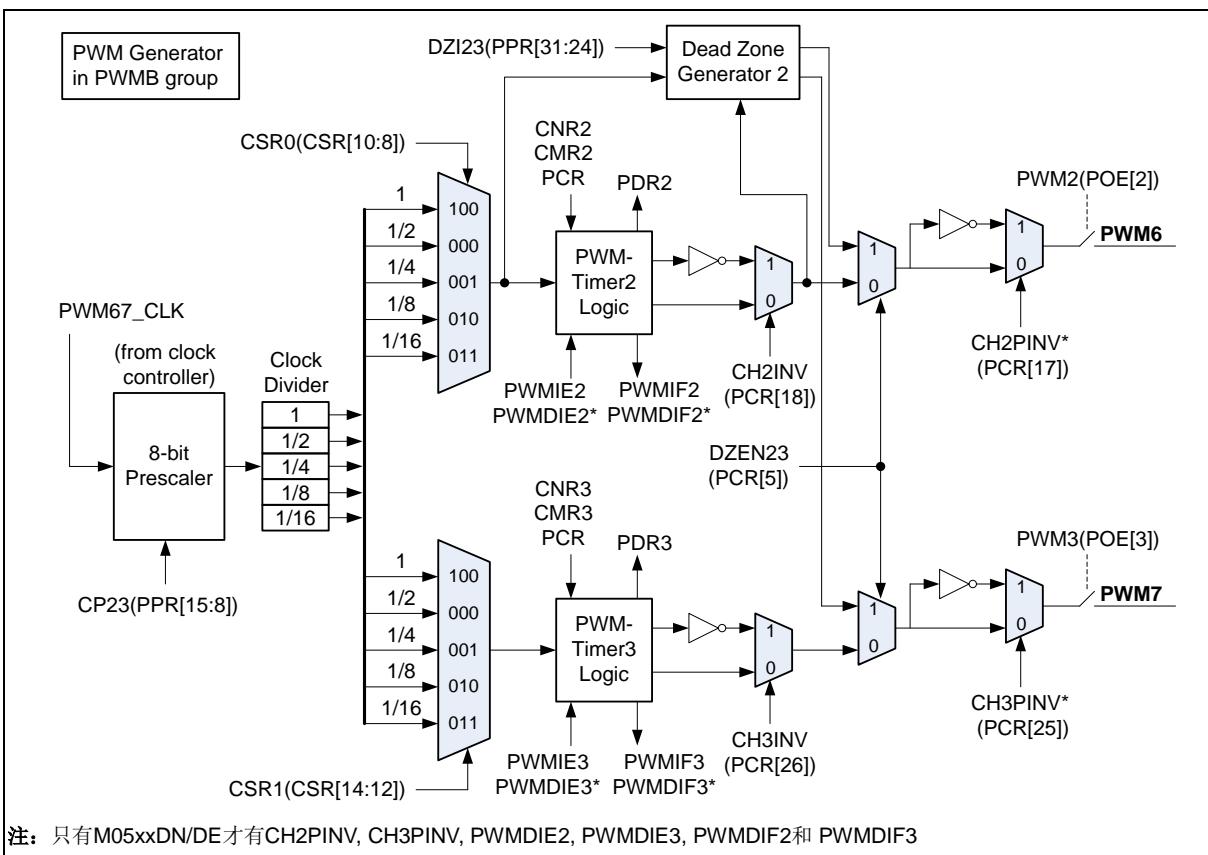


图 6-65 PWM 发生器 6 结构框图

#### 6.11.4 基本配置

PWM 引脚在P2\_MFP and P4\_MFP 寄存器中配置.

PWM 时钟在APBCLK[23:20]中使能。 PWM 时钟源在CLKSEL1[31:28] 和 CLKSEL2[7:4]中选择.

#### 6.11.5 PWM 功能描述

##### 6.11.5.1 PWM-定时器操作

M05xxBN中， PWM控制器只支持边沿对齐。 M05xxDN/DE中PWM控制器支持边沿对齐和中心对齐  
边沿对齐 PWM (下数计数器)

边沿对齐PWM输出模式下，16位PWM计数器从CNRx开始向下数，直到计数器的值等于CMRx，然后PWMx发生器反转输出为高电平。计数器继续向下数直到0，此时PWMx发生器反转输出为高电平，如果CHxMODE=1，CMRx(新的)和CNRx(新的)被更新，此时如果PWM中断使能(PIERx=1)，PWM中断将发生。

PWM 周期和占空比控制由PWM向下计数器寄存器(CNR)以及PWM比较寄存器(CMR)配置。 PWM定时器工作时序如下图所示。. 脉宽调制的公式如下， PWM定时器比较器的说明如图6-67所示。注意：相应的GPIO管脚必须配置成PWM功能(使能POE 和禁用CAPENR)。

- PWM 频率 =  $\text{PWM}_{xy\_CLK}/((\text{prescale}+1)*(\text{clock divider})*(\text{CNR}+1))$ ; xy 代表 01, 23, 45 或 67, 取决于所选择的 PWM 通道.
- 占空比 =  $(\text{CMR}+1)/(\text{CNR}+1)$
- $\text{CMR} \geq \text{CNR}$ : PWM 输出总为高
- $\text{CMR} < \text{CNR}$ : PWM低脉冲宽度=  $(\text{CNR}-\text{CMR})$ 单位<sup>[1]</sup>; PWM高脉冲宽度 =  $(\text{CMR}+1)$  单位
- $\text{CMR} = 0$ : PWM 低脉冲宽度 =  $(\text{CNR})$  单位; PWM高脉冲宽度= 1 单位

注: [1]. 单位 = 一个PWM时钟周期

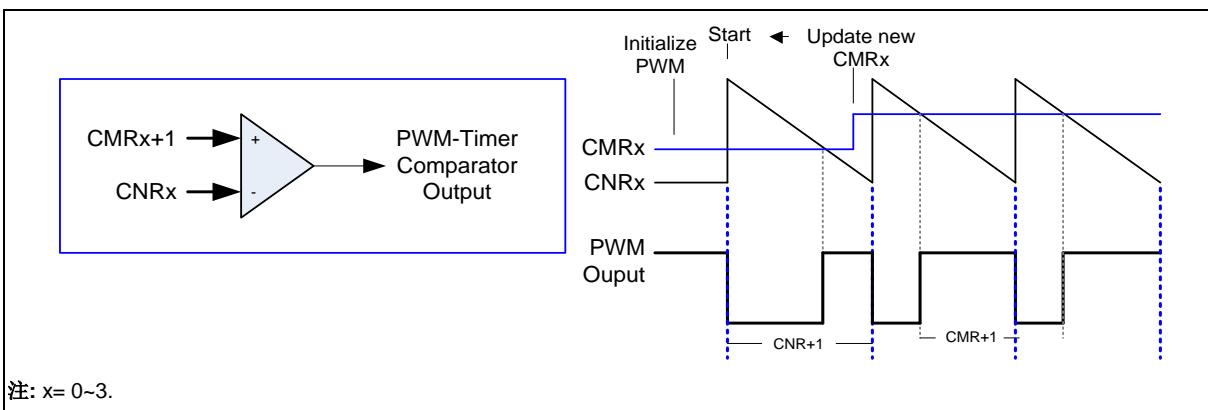


图 6-66 PWM 定时器内部比较器输出

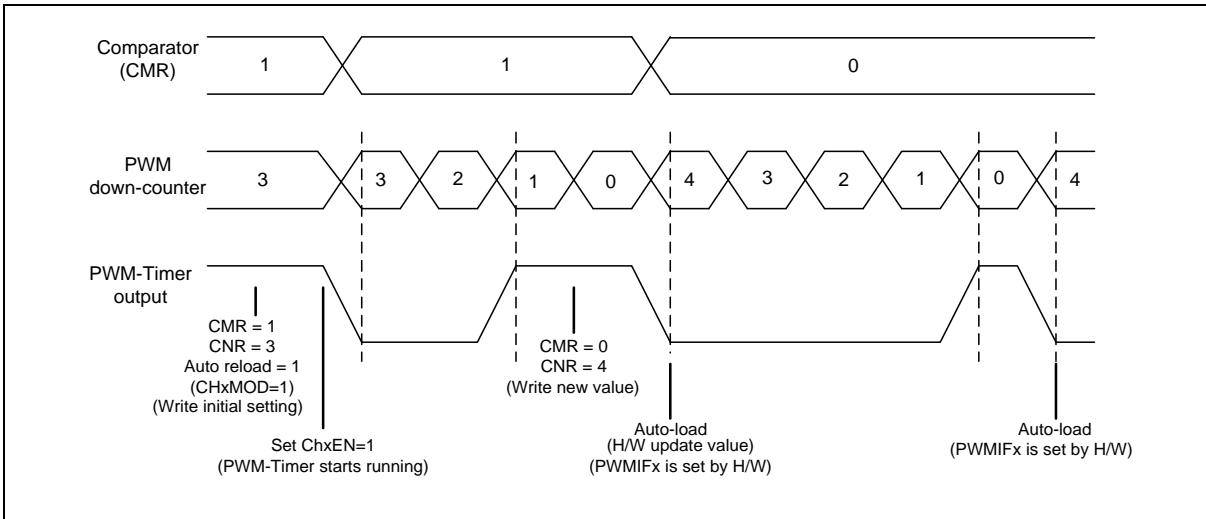


图 6-67 PWM 定时器操作时序

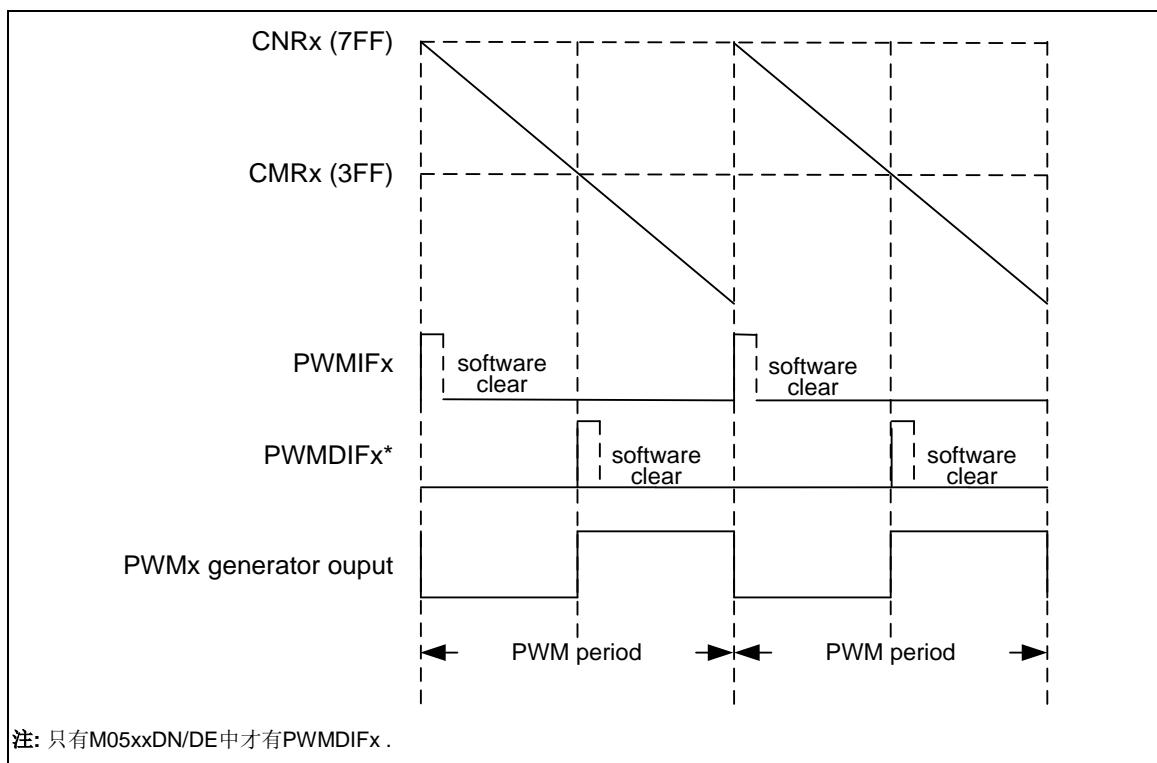


图 6-68 PWM 周期中断发生时序图

### 中心对齐 PWM (上数/下数计数器) (M05xxDN/DE)

当PWM配置为中心对齐模式时，其计数器为上数/下数模式。PWM计数器从0开始上数直到等于CMRx的值，然后PWMx发生器反转输出为低电平。计数器继续上数直到等于CNRx的值，然后计数器自动开始下数直到再次等于CMRx的值，然后PWM发生器反转输出为高电平。计数器继续下数一直到0，如果CHxMODE = 1，CNRx和CMRx的值将被重新加载，再次开始上述的循环。

- PWM 频率 =  $\text{PWMxy\_CLK}/[(\text{prescale}+1) * (\text{clock divider}) * (\text{CNR}+1)]$ ; 根据选择的 PWM 通道, xy 应该是 01, 23, 45 or 67.
  - 占空比 =  $[(2 \times \text{CMR}) + 1]/[2 \times (\text{CNR}+1)]$
  - CMR > CNR: PWM 输出总是为高
  - CMR <= CNR: PWM 低电平宽度 =  $2 \times (\text{CNR}-\text{CMR}) + 1$  单位<sup>[1]</sup>; PWM 高电平宽度 =  $(2 \times \text{CMR}) + 1$  单位
  - CMR = 0: PWM 低电平宽度 =  $2 \times \text{CNR} + 1$  单位; PWM 高电平宽度 = 1 单位
- 注 [1]: 单位 = 一个 PWM 时钟周期.

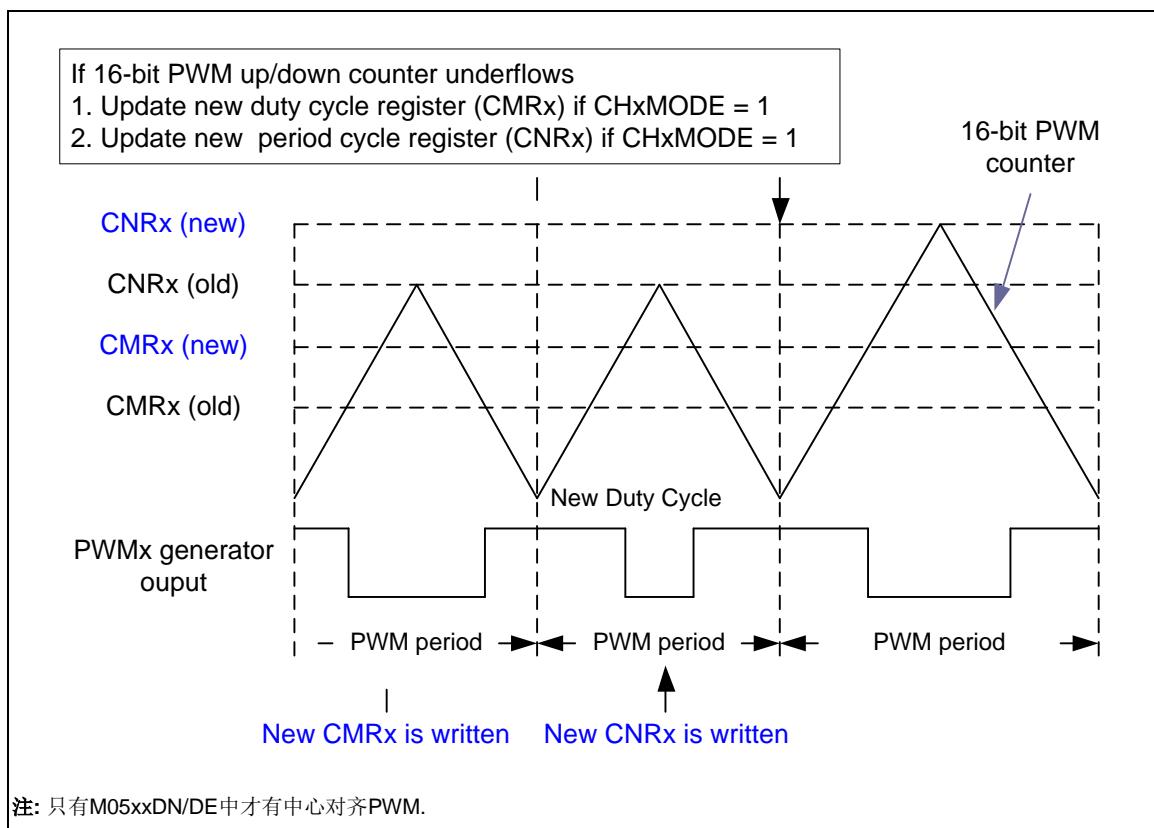


图 6-69 中心对齐输出波形

中心对齐模式下，系统可以产生2种中断：周期中断和工作中断，有4种时序：如果INTxxTYPE (PIER[17:16]) = 0，计数器下数到0时发生PWM 周期中断；INTxxTYPE (PIER[17:16]) = 1，计数器上数到等于CMRx的值时发生 PWM周期中断，就是说在 PWM周期的中点发生中断；如果INTxxDTYPE (PIER[25:24]) = 0，计数器下数等于CMRx 的值发生 PWM 工作中断；如果INTxxDTYPE (PIER[25:24]) = 1，计数器上数等于CMRx的值发生PWM工作中断。

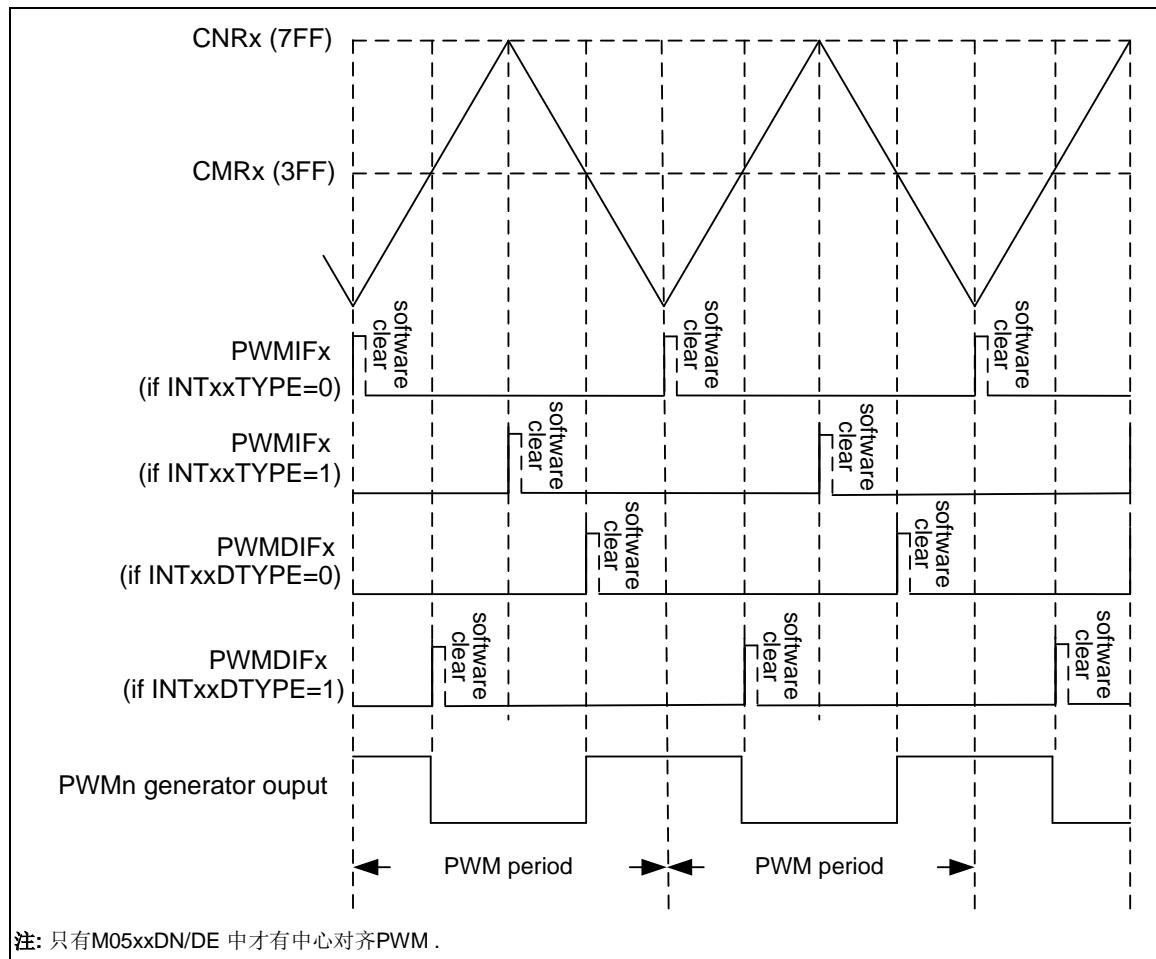


图 6-70 PWM 中心对齐中断发生时序图

### 6.11.5.2 PWM双缓存，自动重载以及单触发模式

PWM定时器具有双缓存功能。新的值将在下一个周期开始时重新加载，不会影响当前定时器的工作。新的PWM计数器的值可写入CNRx，当前PWM计数器的值可以从PDRx读取。

PWM0 控制寄存器(PCR) 的CH0MOD 位定义 PWM0 是自动重载模式还是单次触发模式。如果CH0MOD 被设置为1，PWM0 为自动重载模式；如果CH0MOD 被设置为0，PWM0 为单次触发模式。设置CH0EN 为1使能 PWM0 计数器开始计数之前，就应该先设好CH0MOD，因为改变CH0MOD 的值会把CNRO 和 CMR0 的内容清为0。如果 PWM0 工作在单次触发模式，填好CMR0 和 CNR0 之后，设置CH0EN 为1使能 PWM0 开始计数。在 PWM0 计数器从CNRO 下数到0 时，CNRO 和 CMR0 寄存器的值将被硬件清为0，PWM0 计数器将停止工作。软件需要填新的CMR0 和 CNR0 的值启动下一次单次触发模式。但重新开始下一次单次触发模式时，CMR0 应该先被写，因为CNRO 一旦被写，PWM0 计数器将自动重新开始计数。

数。如果PWM0工作在自动重载模式，填好CNR0和CMR0之后，设置CH0EN等于1启动PWM0开始计数。当计数器下数到0时，CNR0的值将被自动加载到PWM0计数器中。如果CNR0设为0，PWM0计数器将停止工作。PWM1 ~ PWM7工作方式和PWM0相同。

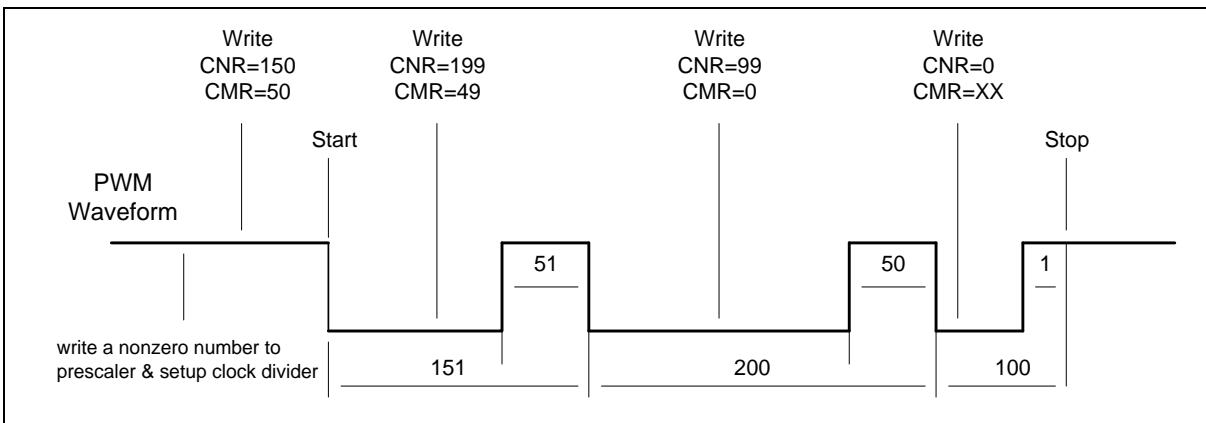


图 6-71 PWM 双缓存图解

#### 6.11.5.3 调制占空比

双缓存功能允许CMRx在当前周期的任意时刻被写入。写入的值将在下个周期生效。

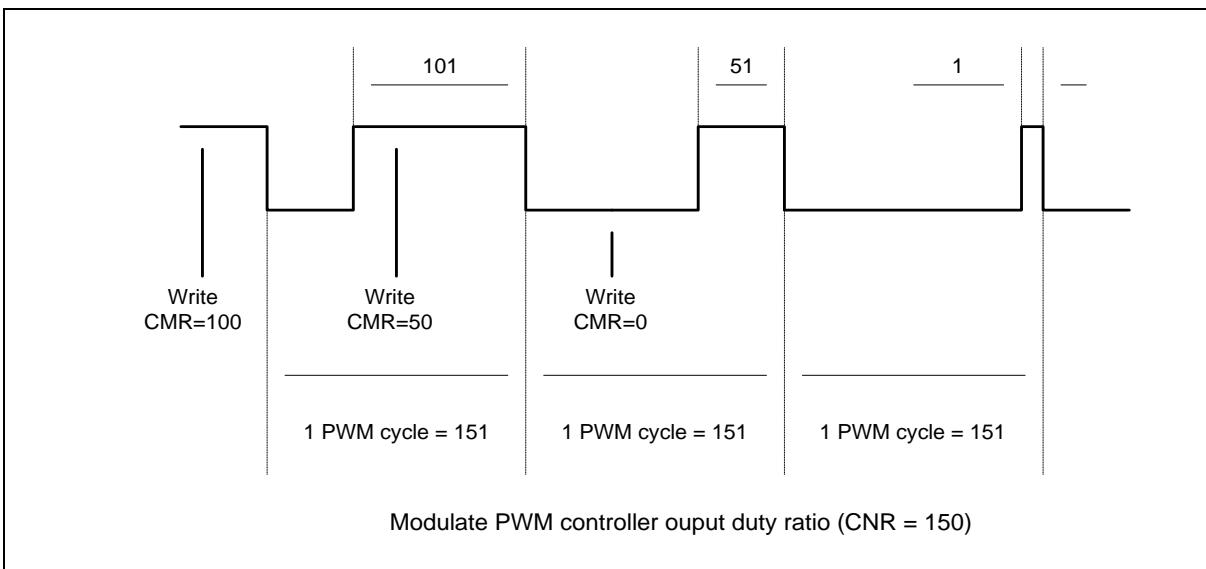


图 6-72 PWM 控制器输出占空比

#### 6.11.5.4 死区发生器

PWM控制器提供死区发生器，用于保护功率器件。该功能产生可编程的时隙来延迟PWM上升沿输出时间点，用户可通过编程PPRx.DZI确定死区间隔。

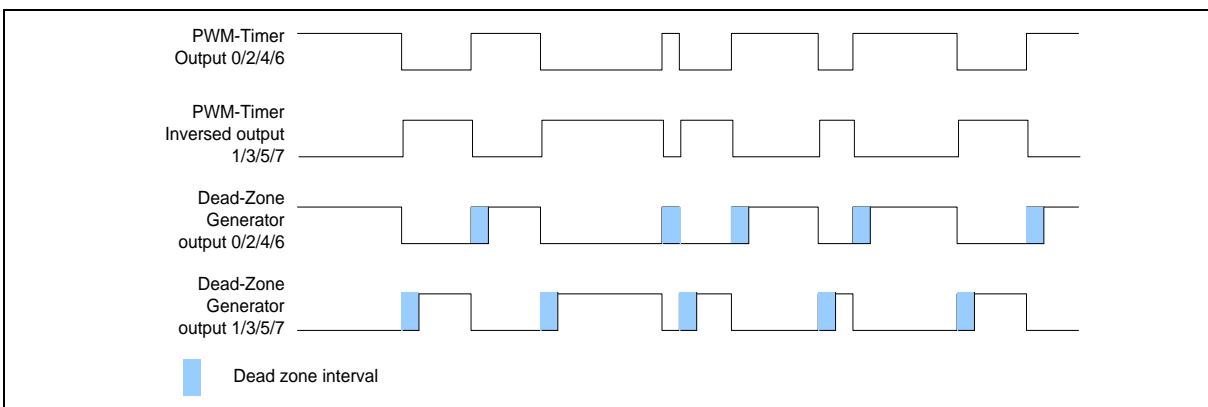


图 6-73 死区发生器操作

#### 6.11.5.5 PWM触发 ADC 开始转换(M05xxDN/DE)

PWM 可以触发ADC开始转换。当 PWM 工作在中心对齐模式时，有4种PWM触发ADC开始转换的时序；当 PWM 工作在边沿对齐模式时，有2种PWM触发ADC开始转换的时序。

对于中心对齐模式，通过设定INTxxDTYPE/ INTxxTYPE 和 PWMxDTEN/ PWMxTEN，PWM可以在不同的时序触发ADC开始转换。触发ADC时序和设定如下所示，共4种。

INTxxTEN	INTxxTYPE	描述
1	0	下数时 PWM 计数器等于0时触发
1	1	上数时 PWM 计数器等于 CNRx + 1 时触发

INTxxDTEN	INTxxDTYPE	描述
1	0	下数时 PWM 计数器等于CMRx时触发
1	1	上数时 PWM 计数器等于CMRx 时触发

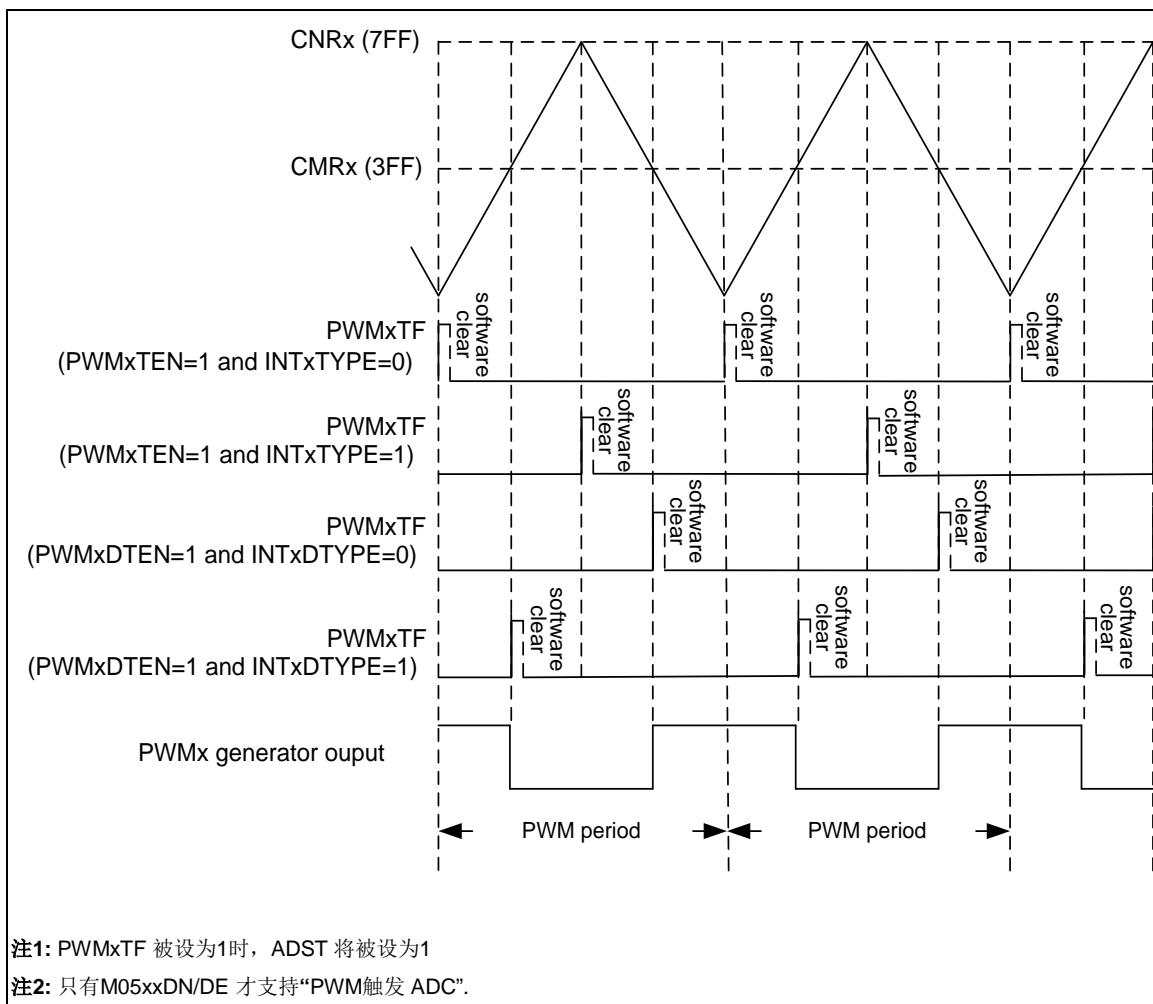


图 6-74 中心对齐模式下 PWM 触发 ADC 标志 (PWMxTF) 时序图

对于边沿对齐模式，通过设定 PWMxDTEN/ PWMxTEN，PWM可以在不同的时序触发 ADC 开始转换。触发ADC时序和设定如下所示，共2种。

INTxDTEN	描述
1	下数时PWM 计数器等于CMRx 触发
INTxTEN	描述
1	下数时PWM 计数器等于0触发

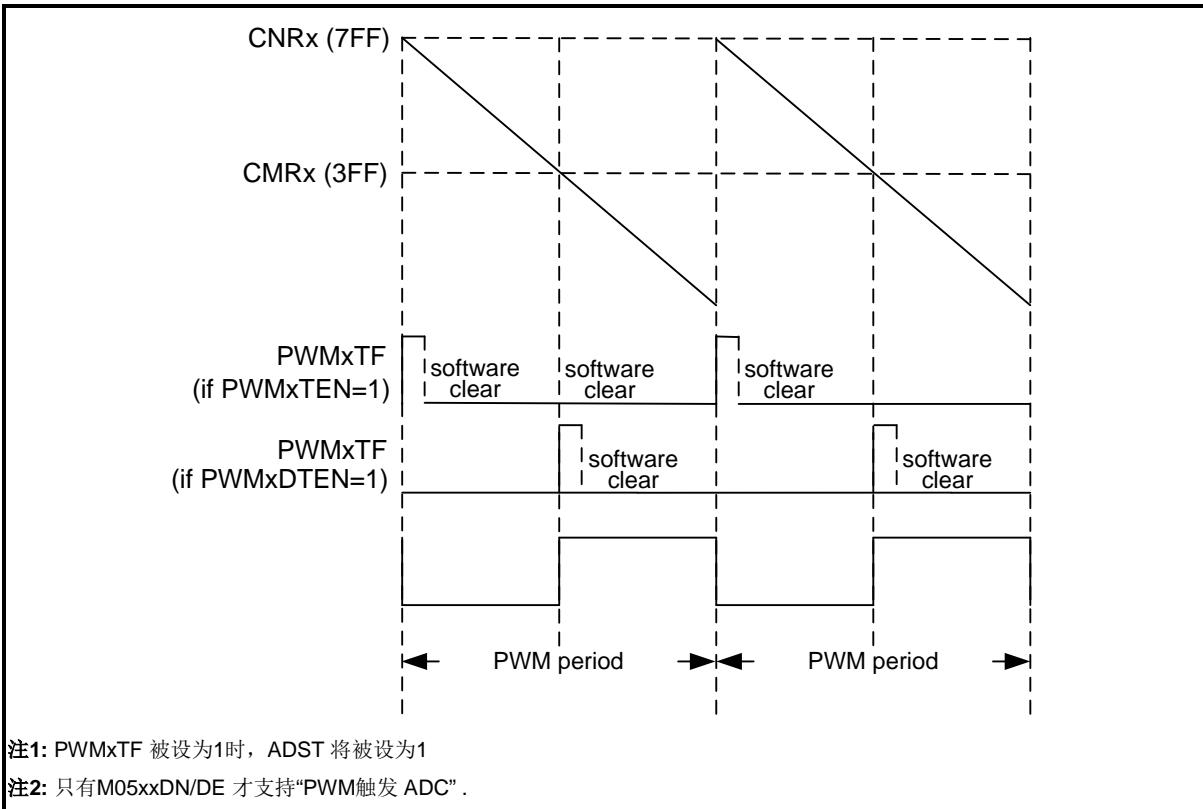


图 6-75 边沿对齐时 PWM 触发 ADC 标志 (PWMxTF) 时序图

## 6.11.5.6 PWM 触发 ADC 开始转换步骤 (M05xxDN/DE)

1. 配置预分频寄存器 (PPRx) 设置预分频值 (CPxx).
2. 配置时钟选择寄存器 (CSR<sub>x</sub>) 选择时钟源除频值(CSR<sub>x</sub>).
3. 配置 PWM 控制寄存器 (PCR<sub>x</sub>) 设置自动重载模式 (CH<sub>x</sub>MOD = 1), PWM 对齐类型 (PWM<sub>xx</sub>TYPE) 并关闭 PWM 计数器 (CH<sub>x</sub>EN = 0).
4. 配置 PWM 控制寄存器 (PCR<sub>x</sub>) 设置反转使能/关闭 (CH<sub>x</sub>INV), 极性反转使能/关闭 (CH<sub>x</sub>PINV) 和死区发生器使能/关闭 (DZEN<sub>xx</sub>). (可选)
5. 配置 PWM 中断使能寄存器 (PIER) 设置 PWM 周期中断类型 (INT<sub>xx</sub>TYPE) 和 PWM 工作中断类型(INT<sub>xx</sub>DTYPE).
6. 配置 PWM 触发控制寄存器(TCON) 使能/禁止PWM 周期触发 ADC (PWM<sub>x</sub>TEN) 和 PWM 工作触发 ADC (PWM<sub>x</sub>DTEN).
7. 配置比较寄存器(CMR<sub>x</sub>) 设置 PWM 高电平时间 (CMRx).
8. 配置 PWM 计数寄存器(CNR<sub>x</sub>) 设置 PWM 的周期(CNR<sub>x</sub>).
9. 配置 PWM 输出使能 寄存器(POE) 使能 PWM 输出通道 (PWM<sub>x</sub> = 1)
10. 配置ADC 触发延迟控制寄存器(ADTDCR) (在 ADC 控制器中)设置 PWM 触发延迟时间 (PTDT) (可选)
11. 配置ADC 通道使能寄存器(ADCHER) (在 ADC 控制器中)使能模拟输入通道 (CHEN<sub>x</sub> =1)

12. 配置ADC 控制 寄存器(ADCR) (在 ADC 控制器中) 设置硬件触发源来自 PWM (TRGS = 3), 使能外部触发(TRGEN = 1), A/D 转换器操作模式(ADM0 = 0,2,3) 和 A/D 转换器使能(ADEN = 1)
13. 配置PWM 控制 寄存器(PCR) 使能 PWM 开始工作 (CHxEN = 1)
14. 当 PWM触发标志(PWMxTF)被硬件设为1时, ADST (ADCR[11]) 位将被硬件设为 1
15. 软件可以轮询 A/D 转换结束标志 – ADF (ADSR[0]) 来检查转换是否结束.
16. 单次模式和单周期扫描模式下ADST (ADCR[11]) 将被自动清为0；循环扫描模式下, ADST 维持为1直到软件将其清0.

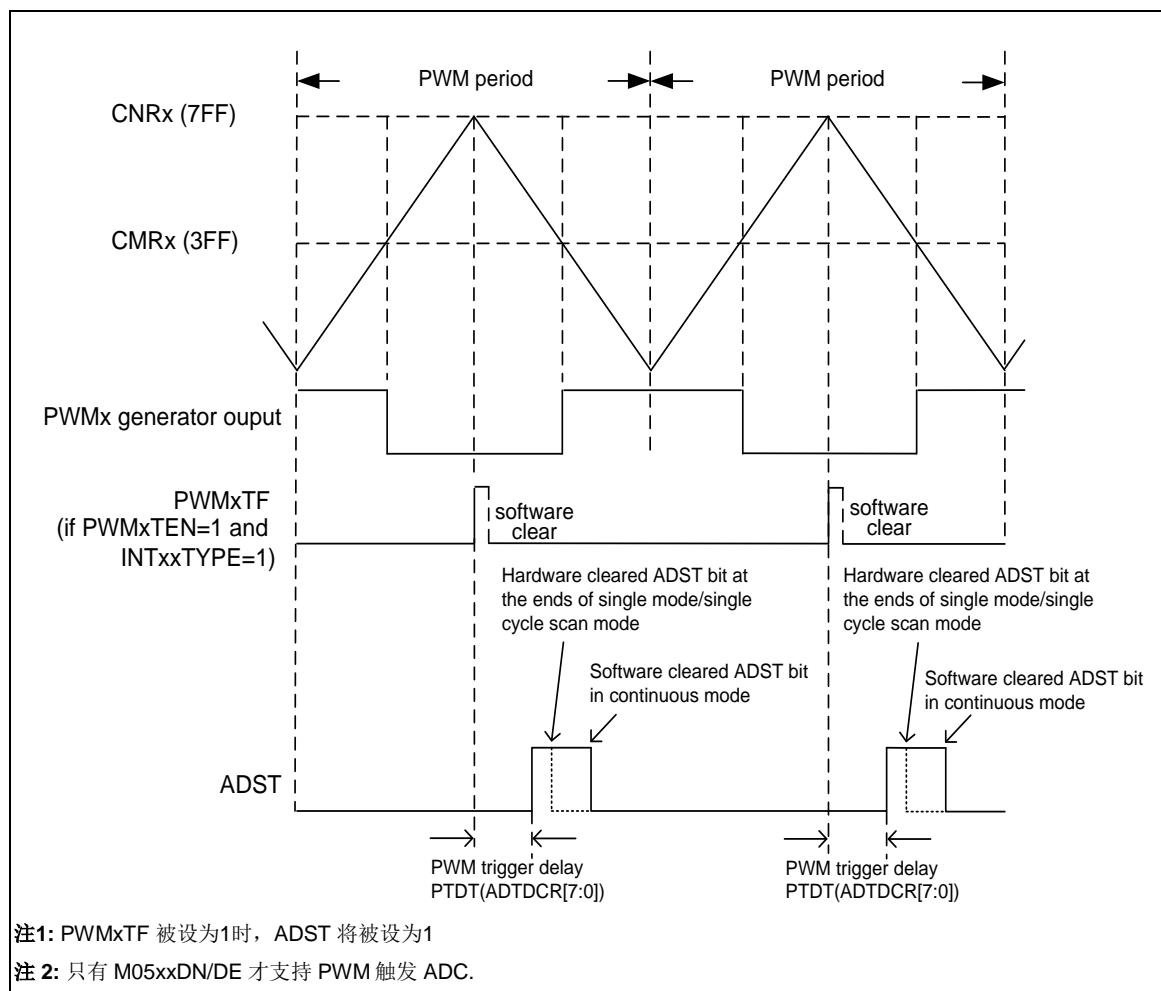


图 6-76 中心对齐模式下 PWM 触发 ADC 开始转换

### 6.11.5.7 捕获操作

捕获器0和PWM0共用同一个定时器，捕获器1和PWM1共用另一个定时器，以此类推。在输入通道有上升沿跳变时捕获器将PWM计数器的值锁存至CRLRx寄存器，在输入通道有下降沿跳变时将PWM计数器的值锁存至CFLRx寄存器。捕获通道0的中断是可编程的，通过设定CCR0.CRL\_IE0[1] (上升沿锁存中断使能) 和 CCR0.CFL\_IE0[2]] (下降沿锁存中断使能) 来决定中断发生的条件。通过设置CCR0.CRL\_IE1[17]和CCR0.CRL\_IE1[18]，捕获通道1有同样的特性。无论捕获模块何时触发一个捕获中断，相应的PWM计数器都将在此刻重载CNRx的值。注：相应的GPIO管脚必须配置成捕获功能(禁用POE和使能CAPENR)。

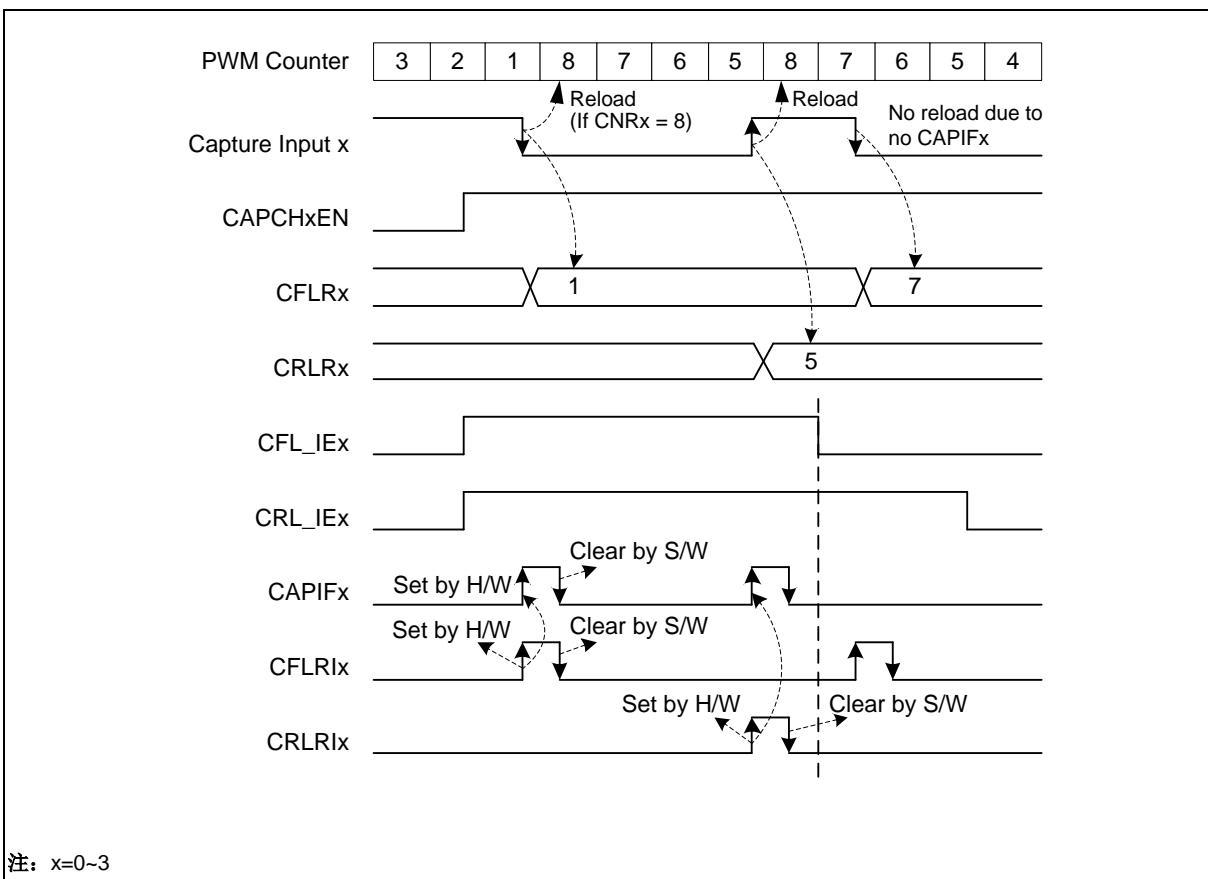


图 6-77 捕捉操作时序

在上述范例中，CNR 为8:

1. 捕捉中断标志(CAPIFx)置位时，PWM计数器将重装载CNRx的值.
2. 通道低脉冲宽度为(CNR + 1 - CRLR).
3. 通道高脉冲宽度为(CNR + 1 - CFLR).

### 6.11.5.8 PWM-定时器中断架构

有8个PWM中断，PWM0\_INT~PWM7\_INT，被分到2个中断向量PWMA\_INT和PWMB\_INT。PWM 0与捕获器0共用同一个中断。PWM1 与捕获器1 共用同一个中断，以此类推。因此，同一通道的PWM功能和捕获功能不能同时使用。下图说明了 PWM定时器中断结构。

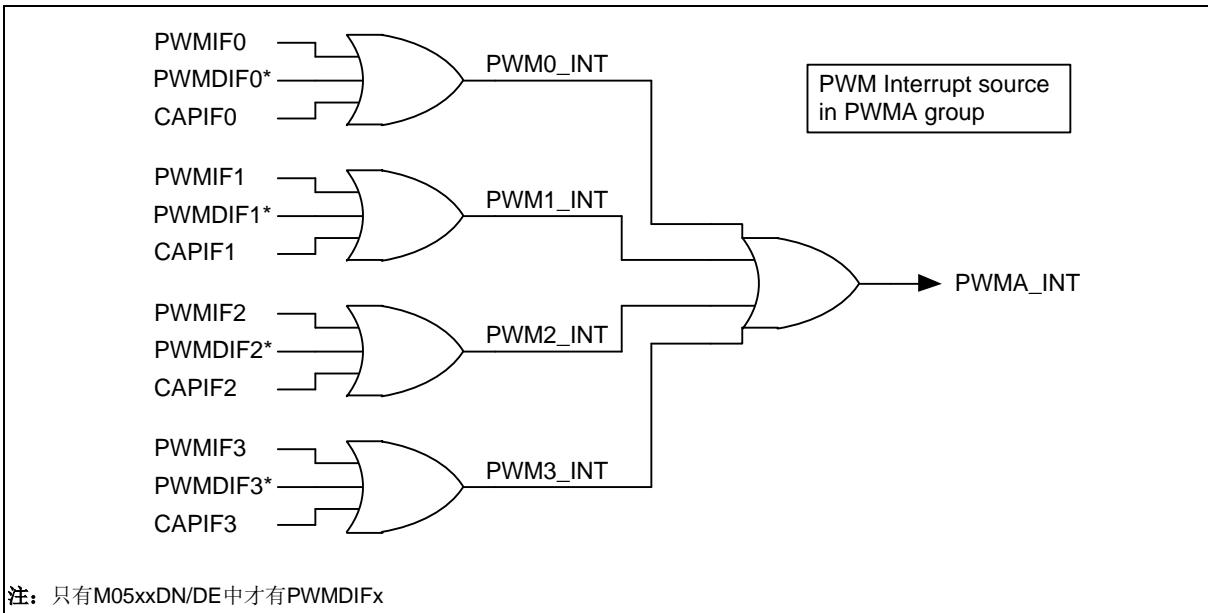


图 6-78 PWM A 组 PWM-定时器中断结构图

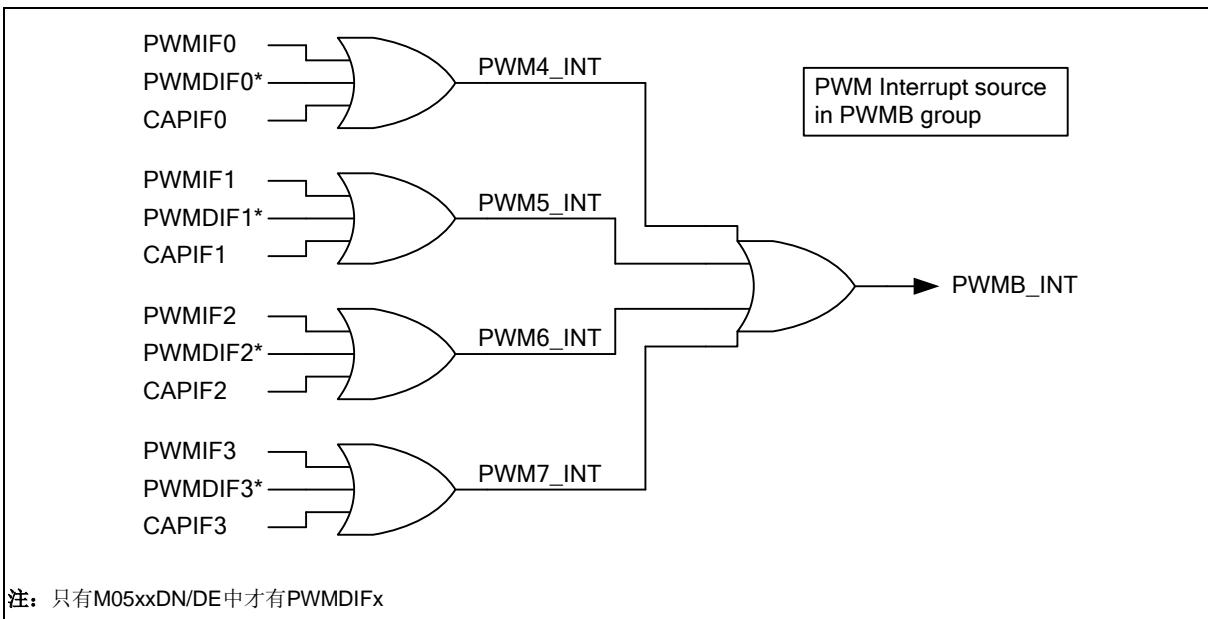


图 6-79 PWM B 组 PWM-定时器中断结构图

#### 6.11.5.9 PWM-定时器启动步骤

推荐使用如下步骤启动PWM驱动器。

1. 配置预分频寄存器(PPR), 设置预分频(CSR)
2. 配置时钟除数选择寄存器 (CSR), 选择时钟除数(CSRx)
3. 配置PWM控制寄存器(PCR), 设置自动重载模式(CHxMOD = 1)并关闭PWM计数器 (CHxEN = 0)
4. 配置PWM控制寄存器(PCR), 设置反向打开/关闭, 死区发生器打开/关闭(可选)
5. 配置比较器寄存器(CMRx), 设定PWM占空比
6. 配置PWM计数器寄存器 (CNRx) 设定PWM周期.
7. 配置PWM中断使能寄存器(PIER), 使能PWM周期中断(PWMIE).(可选)
8. 配置PWM输出使能寄存器(POE), 使能PWM输出通道(PWMx = 1)
9. 配置PWM控制寄存器(PCR), 使能PWM定时器开始运行(CHxEN = 1)

#### 6.11.5.10 单次触发模式下PWM定时器重新启动步骤

单次触发模式下, PWM 波形产生一次以后, PWM定时器将自动停止计数, CNRx 和 CMRx 将被硬件自动清0。软件必须重填 CMRx 和 CNRx 的值来再次启动PWM。单次触发模式下推荐下列步骤来重新启动PWM

1. 配置比较寄存器 (CMRx), 设置 PWM 高电平 (CMRx).
2. 配置 PWM 计数器寄存器 (CNRx) 来设置 PWM 的周期 (CNRx)。一旦写 CNRx 寄存器, PWM 将马上启动, 波形将产生.

#### 6.11.5.11 PWM-定时器停止步骤

##### 方式 1:

设定16位计数器(CNRx)为0 , 并监视数据寄存器(PDRx, 16位向下计数器的当前值)。当PDRx达到0 , 关闭PWM定时器 (PCR的CHxEN位)。 (推荐)

##### 方式 2:

设定16位计数器(CNR)为0, 当中断请求发生, 关闭PWM定时器(PCR的CHxEN位)。 (推荐)

##### 方式 3:

直接关闭PWM定时器(PCR的CHxEN位)。 (不推荐)

不推荐方式3的原因是: 禁用CHxEN将立即停止PWM输出信号, 引起PWM输出占空比的改变, 这可能导致电机控制电路损坏。

#### 6.11.5.12 PWM定时器同步启动步骤

PWMA组定时器1~3 、 PWMB组定时器0~3 可以和PWM A组定时器0同时启动。 PWM定时器同步

启动步骤如下：

1. 配置预分频 寄存器(PPR)，设置预分频值 (CPxx).
2. 配置时钟选择 寄存器(CSR) ，选择时钟源除频值 (CSR<sub>x</sub>).
3. 配置PWM 控制 寄存器(PCR)，设置反转自动加载模式(CH<sub>x</sub>MOD = 1), PWM 对齐类型 (PWM<sub>xx</sub>TYPE) 并关闭PWM定时器 (CH<sub>x</sub>EN = 0).
4. 配置PWM 控制 寄存器(PCR) ，设置反转打开/关闭 (CH<sub>x</sub>INV), 极性反转打开/关闭(CH<sub>x</sub>PINV) 和死区发生器 打开/关闭(DZEN<sub>n</sub>). (可选)
5. 配置比较 寄存器(CMRx)，设置 PWM 高电平 (CMRx).
6. 配置PWM 计数寄存器 (CNRx) ，设置PWM周期 (CNRx).
7. 配置PWM 中断使能 寄存器(PIER)，设置 PWM 周期中断类型 (INT<sub>xx</sub>TYPE), PWM 工作中断类型 (INT<sub>xx</sub>DTYPE), PWM 周期中断使能位 (PWMIEx) 和 PWM 工作中断使能位 (PWMDIEx). (可选)
8. 配置PWM 输出使能 寄存器(POE)，使能PWM输出通道 (PWM<sub>x</sub>)
9. 配置PWM 同步控制 寄存器(PSCR)使能 PWM 同步使能位 (PSSEN<sub>x</sub> = 1)
10. 配置 PWMA组 控制 寄存器(PCR) 使能 A组 PWM定时器0 开始计数 (A组PCR中CH0EN = 1). 所有相关的PWM定时器江河A组的定时器0同步启动

#### 6.11.5.13 捕获开始步骤

- 配置预分频寄存器(PPR)，设置时钟预分频(CPxx)
- 配置时钟选择寄存器(CSR)，选择时钟除频值
- 配置PWM控制寄存器(PCR)，选择自动加载模式(CH<sub>x</sub>MOD = 1) 并关闭PWM定时器 (CH<sub>x</sub>EN = 0)
- 配置 PWM捕获控制寄存器(CCRx)，设置上升沿中断使能(CRL\_IEx)和下降沿中断使能 (CFL\_IEx)以及输入信号反向打开/关闭(INVx)
- 配置PWM计数器寄存器(CNRx)，设置计数周期
- 配置PWM控制寄存器(PCR)，使能PWM计数器开始计数(CH<sub>x</sub>EN = 1).
- 使能捕获输入通道使能定时器(CAPENR)，使能相应的GPIO引脚作为捕获功能

### 6.11.6 寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>PWM基地址:</b>				
<ul style="list-style-type: none"> <li>● PWM A 组 PWMA_BA = 0x4004_0000</li> <li>● PWM B 组 PWMB_BA = 0x4014_0000</li> </ul>				
PPR	PWMA_BA+0x00	R/W	PWM预分频寄存器	0x0000_0000
	PWMB_BA+0x00			
CSR	PWMA_BA+0x04	R/W	PWM时钟除数选择寄存器	0x0000_0000
	PWMB_BA+0x04			
PCR	PWMA_BA+0x08	R/W	PWM控制寄存器	0x0000_0000
	PWMB_BA+0x08			
CNR0	PWMA_BA+0x0C	R/W	PWM计数寄存器0	0x0000_0000
	PWMB_BA+0x0C			
CMR0	PWMA_BA+0x10	R/W	PWM比较寄存器0	0x0000_0000
	PWMB_BA+0x10			
PDR0	PWMA_BA+0x14	R	PWM数据寄存器0	0x0000_0000
	PWMB_BA+0x14			
CNR1	PWMA_BA+0x18	R/W	PWM计数寄存器1	0x0000_0000
	PWMB_BA+0x18			
CMR1	PWMA_BA+0x1C	R/W	PWM比较寄存器1	0x0000_0000
	PWMB_BA+0x1C			
PDR1	PWMA_BA+0x20	R	PWM数据寄存器1	0x0000_0000
	PWMB_BA+0x20			
CNR2	PWMA_BA+0x24	R/W	PWM计数寄存器2	0x0000_0000
	PWMB_BA+0x24			
CMR2	PWMA_BA+0x28	R/W	PWM 比较寄存器2	0x0000_0000
	PWMB_BA+0x28			
PDR2	PWMA_BA+0x2C	R	PWM数据寄存器 2	0x0000_0000
	PWMB_BA+0x2C			

<b>CNR3</b>	PWMA_BA+0x30	R/W	PWM 计数寄存器 3	0x0000_0000
	PWMB_BA+0x30			
<b>CMR3</b>	PWMA_BA+0x34	R/W	PWM 比较寄存器3	0x0000_0000
	PWMB_BA+0x34			
<b>PDR3</b>	PWMA_BA+0x38	R	PWM 数据寄存器3	0x0000_0000
	PWMB_BA+0x38			
<b>PIER</b>	PWMA_BA+0x40	R/W	PWM 中断使能寄存器	0x0000_0000
	PWMB_BA+0x40			
<b>PIIR</b>	PWMA_BA+0x44	R/W	PWM 中断标志寄存器	0x0000_0000
	PWMB_BA+0x44			
<b>CCR0</b>	PWMA_BA+0x50	R/W	PWM 捕捉控制寄存器 0	0x0000_0000
	PWMB_BA+0x50			
<b>CCR2</b>	PWMA_BA+0x54	R/W	PWM 捕捉控制寄存器2	0x0000_0000
	PWMB_BA+0x54			
<b>CRLR0</b>	PWMA_BA+0x58	R	PWM 捕捉上升沿锁存寄存器(Channel 0)	0x0000_0000
	PWMB_BA+0x58			
<b>CFLR0</b>	PWMA_BA+0x5C	R	PWM 捕捉下降沿锁存寄存器(Channel 0)	0x0000_0000
	PWMB_BA+0x5C			
<b>CRLR1</b>	PWMA_BA+0x60	R	PWM 捕捉上升沿锁存寄存器(Channel 1)	0x0000_0000
	PWMB_BA+0x60			
<b>CFLR1</b>	PWMA_BA+0x64	R	PWM 捕捉下降沿锁存寄存器(Channel 1)	0x0000_0000
	PWMB_BA+0x64			
<b>CRLR2</b>	PWMA_BA+0x68	R	PWM 捕捉上升沿锁存寄存器(Channel 2)	0x0000_0000
	PWMB_BA+0x68			
<b>CFLR2</b>	PWMA_BA+0x6C	R	PWM 捕捉下降沿锁存寄存器(Channel 2)	0x0000_0000
	PWMB_BA+0x6C			
<b>CRLR3</b>	PWMA_BA+0x70	R	PWM 捕捉上升沿锁存寄存器(Channel 3)	0x0000_0000
	PWMB_BA+0x70			
<b>CFLR3</b>	PWMA_BA+0x74	R	PWM 捕捉下降沿锁存寄存器(Channel 3)	0x0000_0000
	PWMB_BA+0x74			

CAPENR	PWMA_BA+0x78	R/W	PWM捕捉输入 0~3 使能寄存器	0x0000_0000
	PWMB_BA+0x78			
POE	PWMA_BA+0x7C	R/W	PWM通道0~3输出使能	0x0000_0000
	PWMB_BA+0x7C			
TCON	PWMA_BA+0x80	R/W	PWM Trigger Control Register for Channel 0~3 (M05xxDN/DE Only)	0x0000_0000
	PWMB_BA+0x80			
TSTATUS	PWMA_BA+0x84	R/W	PWM Trigger Status Register (M05xxDN/DE Only)	0x0000_0000
	PWMB_BA+0x84			
PSCR	PWMA_BA+0x98	R/W	PWM Synchronous Control Register (M05xxDN/DE Only)	0x0000_0000
	PWMB_BA+0x98			

### 6.11.7 寄存器描述

#### PWM 预分频寄存器 (PPR)

寄存器	偏移量	R/W	描述	复位后的值
PPR	PWMA_BA+0x00	R/W	PWM 预分频寄存器	0x0000_0000
	PWMB_BA+0x00			

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

Bits	描述
[31:24]	DZI23 通道2和通道3死区间隔设定寄存器(PWMA组的PWM2与PWM3, PWMB组的PWM6与PWM7) 该8位寄存器决定死区长度。 死区时间单位 = [(prescale+1)*(clock source divider)]/ PWMxy_CLK (xy 可能是 23 或者 67).
[23:16]	DZI01 通道0和通道1死区间隔设定寄存器(PWMA组的PWM0与PWM1, PWMB组的PWM4与PWM5) 该8位寄存器决定死区长度。 死区时间单位 = [(prescale+1)*(clock source divider)]/ PWMxy_CLK (xy 可能是 01 或者 45).
[15:8]	CP23 时钟预分频器 2 (PWMA组的PWM 定时器 2 & 3, PWMB组的PWM 定时器 6 & 7) 时钟输入相应PWM计数器之前, 根据(CP23 + 1)分频 如果CP23=0, 预分频器2输出时钟停止, PWM 计数器2和3也停止.
[7:0]	CP01 时钟预分频器0(PWMA组的PWM counter 0 & 1, PWMB组的PWM counter 4 & 5) 时钟输入相应PWM计数器之前, 根据(CP01 + 1)分频 如果CP01=0, 预分频器0输出时钟停止。PWM计数器0和1也停止.



### PWM 时钟除频选择寄存器(CSR)

寄存器	偏移量	R/W	描述	复位后的值
CSR	PWMA_BA+0x04	R/W	PWM时钟除频选择寄存器	0x0000_0000
	PWMB_BA+0x04			

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	CSR3			保留	CSR2		
7	6	5	4	3	2	1	0
保留	CSR1			保留	CSR0		

Bits	描述													
[31:15]	保留	保留												
[14:12]	CSR3	PWM定时器 3 时钟源除频选择(PWM定时器3对应于A组， PWM定时器7对应于B组) 为PWM定时器3选择时钟除频. <table border="1" style="margin-left: 20px;"> <tr> <th>CSR3 [14:12]</th> <th>输入时钟分频</th> </tr> <tr> <td>100</td> <td>1</td> </tr> <tr> <td>011</td> <td>16</td> </tr> <tr> <td>010</td> <td>8</td> </tr> <tr> <td>001</td> <td>4</td> </tr> <tr> <td>000</td> <td>2</td> </tr> </table>	CSR3 [14:12]	输入时钟分频	100	1	011	16	010	8	001	4	000	2
CSR3 [14:12]	输入时钟分频													
100	1													
011	16													
010	8													
001	4													
000	2													
[11]	保留	保留												
[10:8]	CSR2	PWM定时器 2 时钟除频选择(PWM定时器2 对应于A组， PWM定时器6 对应于B组) 为PWM定时器选择时钟除频. (表格同CSR3)												
[7]	保留	保留												
[6:4]	CSR1	PWM定时器 1 时钟除频选择(PWM定时器1 对应于A组， PWM定时器5 对应于B组) 为PWM定时器选择时钟除频. (表格同CSR3)												
[3]	保留	保留												
[2:0]	CSR0	PWM定时器 0 时钟除频选择(PWM定时器0 对应于A组， PWM定时器4 对应于B组)												

		为PWM定时器选择时钟除频。 (表格同CSR3)
--	--	-----------------------------

**PWM 控制寄存器(PCR)**

寄存器	偏移量	R/W	描述	复位后的值
PCR	PWMA_BA+0x08	R/W	PWM 控制寄存器	0x0000_0000
	PWMB_BA+0x08			

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

Bits	描述
[31]	PWM23TYPE PWM23 对齐类型选择位(A组的PWM2 和 PWM3 为一对, B组的PWM6 和 PWM7 为一对) (M05xxDN/DE ) 0 = 边沿对齐类型. 1 = 中心对齐类型.
[30]	PWM01TYPE PWM01 对齐类型选择位(A组的PWM0 和 PWM1 为一对, B组的PWM4 和 PWM5 为一对) (M05xxDN/DE ) 0 = 边沿对齐类型. 1 = 中心对齐类型.
[29:28]	保留 保留
[27]	CH3MOD PWM-定时器3 自动重载/单触发模式选择(A组的PWM定时器3 和B组的PWM定时器7) 1 = 自动重载模式 0 = 单触发模式 注: 如果该位的值发生改变, 会导致CNR3 和CMR3 被清0
[26]	CH3INV PWM定时器3输出反转使能控制(A组的PWM定时器3 和B组的PWM定时器7) 1 = 反转打开 0 = 反转关闭
[25]	CH3PINV PWM-定时器3输出极性反转使能控制 (A组的 PWM 定时器3 和 B 组的 PWM 定时器7) (M05xxDN/DE ) 0 = 极性反转禁止. 1 = 极性反转使能.
[24]	CH3EN PWM定时器3 使能控制(A组的PWM定时器3 和B组的PWM定时器7) 1 = 相应PWM定时器开始运行

		0 = 相应PWM定时器停止运行
[23:20]	保留	保留
[19]	CH2MOD	<p><b>PWM-定时器2 自动重载/单触发模式选择 (A组的PWM定时器2 和B组的PWM定时器6)</b></p> <p>1 = 自动重载模式 0 = 单触发模式</p> <p>注: 如果该位的值发生改变, 会导致CNR2 和CMR2 被清0.</p>
[18]	CH2INV	<p><b>PWM-定时器2输出反转使能控制 (A组的PWM定时器2 和B组的PWM定时器6)</b></p> <p>1 = 反转打开 0 = 反转关闭</p>
[17]	CH2PINV	<p><b>PWM-定时器2输出极性反转使能控制 (A组的PWM 定时器 2和B组的 PWM 定时器 6) (M05xxDN/DE )</b></p> <p>0 = 极性反转禁止. 1 = 极性反转使能.</p>
[16]	CH2EN	<p><b>PWM定时器2 使能 (A组的PWM定时器2 和B组的PWM定时器6)</b></p> <p>1 = 相应PWM定时器开始运行 0 =相应PWM定时器停止运行</p>
[15:12]	保留	保留
[11]	CH1MOD	<p><b>PWM定时器1 自动重载/单触发模式选择(A组的PWM定时器1 和B组的PWM定时器5)</b></p> <p>1 = 自动重载模式 0 = 单触发模式</p> <p>注: 如果该位的值发生改变, 会导致CNR1 和CMR1 被清0.</p>
[10]	CH1INV	<p><b>PWM-定时器1输出反转使能 (A组的PWM定时器1 和B组的PWM定时器5)</b></p> <p>1 = 反转打开 0 = 反转关闭</p>
[9]	CH1PINV	<p><b>PWM-定时器1输出极性反转使能控制 (A组的PWM 定时器 1和B组的 PWM 定时器 5) (M05xxDN/DE )</b></p> <p>0 = 极性反转禁止. 1 = 极性反转使能.</p>
[8]	CH1EN	<p><b>PWM定时器1 使能控制 (A组的PWM 定时器 1和B组的 PWM 定时器 5)</b></p> <p>1 = 相应PWM定时器开始运行 0 =相应PWM定时器停止运行</p>
[7:6]	保留	保留
[5]	DZEN23	<p><b>死区发生器2使能控制(A组的PWM2 和 PWM3 为一对, B组的PWM6 和 PWM7 为一对)</b></p> <p>1 = 使能 0 = 禁用</p> <p>注: 当死区发生器使能时, PWM A组的PWM2与PWM3将成为互补对, PWM B组的PWM6与PWM7将成为互补对.</p>
[4]	DZEN01	<p><b>死区发生器0使能控制(A组的PWM0 和 PWM1 为一对, B组的PWM4 和 PWM5 为一对)</b></p> <p>1 = 使能 0 = 禁用</p> <p>注: 当死区发生器使能时, PWM A组的PWM0与PWM1将成为互补对, PWM B组的PWM4与</p>

		PWM5将成为互补对.
[3]	<b>CH0MOD</b>	<b>PWM-定时器 0 自动加载/单触发模式选择 (A组的PWM 定时器 0和B组的 PWM 定时器 4)</b> 1 = 自动重载模式 0 = 单触发模式 注: 如果该位的值发生改变, 会导致CNR0 和CMR0 被清0.
[2]	<b>CH0INV</b>	<b>PWM-定时器0输出反转使能控制 (A组的PWM 定时器 0和B组的 PWM 定时器 4)</b> 1 = 反转打开 0 = 反转关闭
[1]	<b>CH0PINV</b>	<b>PWM-定时器0输出极性反转使能控制 (A组的PWM 定时器 0和B组的 PWM 定时器 4) (M05xxDN/DE )</b> 0 = 极性反正禁止. 1 = 极性反正使能.
[0]	<b>CH0EN</b>	<b>PWM-定时器0 使能控制 (A组的PWM 定时器 0和B组的 PWM 定时器 4)</b> 1 = 相应PWM定时器开始运行 0 =相应PWM定时器停止运行

PWM 计数器寄存器3-0 (CNR3-0)

寄存器	偏移量	R/W	描述	复位后的值
CNR0	PWMA_BA+0x0C	R/W	PWM 计数器寄存器0	0x0000_0000
	PWMB_BA+0x0C			
CNR1	PWMA_BA+0x18	R/W	PWM 计数器寄存器1	0x0000_0000
	PWMB_BA+0x18			
CNR2	PWMA_BA+0x24	R/W	PWM 计数器寄存器2	0x0000_0000
	PWMB_BA+0x24			
CNR3	PWMA_BA+0x30	R/W	PWM 计数器寄存器3	0x0000_0000
	PWMB_BA+0x30			

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
CNRx [15:8]							
7	6	5	4	3	2	1	0
CNRx [7:0]							

Bits	描述
[31:16]	保留
[15:0]	<p><b>CNRx</b></p> <p><b>PWM 计数器载入值</b> CNR 决定 PWM 的周期. PWM 频率 = PWMxy_CLK / ((prescale+1) * (clock divider) * (CNR+1)); xy代表01, 23, 45或 67, 取决于所选择的PWM通道.</p> <p><b>对于边沿对齐类型:</b></p> <ul style="list-style-type: none"> <li>占空比 = (CMR+1)/(CNR+1).</li> <li>CMR &gt;= CNR: PWM输出总是高.</li> <li>CMR &lt; CNR: PWM低脉冲宽度= (CNR-CMR) 单位; PWM高脉冲宽度= (CMR+1) 单位.</li> <li>CMR = 0: PWM低脉冲宽度= (CNR) 单位; PWM高脉冲宽度= 1 单位</li> </ul> <p><b>对于中心对齐类型 (M05xxDN/DE):</b></p> <ul style="list-style-type: none"> <li>占空比= [(2 x CMR) + 1]/[2 x (CNR+1)].</li> <li>CMR &gt; CNR: PWM 输出总是高.</li> <li>CMR &lt;= CNR: PWM 低脉冲宽度 = 2 x (CNR-CMR) + 1单位; PWM 高脉冲宽度 = (2 x CMR) 单位</li> </ul>

		<p>x CMR) + 1单位.</p> <ul style="list-style-type: none"><li>● CMR = 0: PWM 低脉冲宽度= <math>2 \times \text{CNR} + 1</math>单位; PWM 高脉冲宽度= 1单位 (单位 = 一个 PWM 时钟周期)</li></ul> <p><b>注1:</b>写数据到CNR寄存器后将在下一个PWM周期生效.</p> <p><b>注2:</b>CNR的值被设为0时, PWM输出总是为高</p> <p><b>注3:</b>PWM工作在中心对齐模式时, CNR的值应该在0x0001 to 0xFFFF之间。如果CNR等于0x0000或者0xFFFF, PWM工作将不正常</p>
--	--	---

PWM 比较器寄存器 3-0 (CMR3-0)

寄存器	偏移量	R/W	描述	复位后的值
CMR0	PWMA_BA+0x10	R/W	PWM 比较器寄存器0	0x0000_0000
	PWMB_BA+0x10			
CMR1	PWMA_BA+0x1C	R/W	PWM 比较器寄存器1	0x0000_0000
	PWMB_BA+0x1C			
CMR2	PWMA_BA+0x28	R/W	PWM 比较器寄存器2	0x0000_0000
	PWMB_BA+0x28			
CMR3	PWMA_BA+0x34	R/W	PWM 比较器寄存器3	0x0000_0000
	PWMB_BA+0x34			

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
CMRx [15:8]							
7	6	5	4	3	2	1	0
CMRx [7:0]							

Bits	描述	
[31:16]	保留	保留
[15:0]	CMRx	<p><b>PWM 比较器寄存器</b></p> <p>CMR 决定 PWM 的占空比.</p> <p>PWM 频率 = PWMxy_CLK/(prescale+1)*(clock divider)/(CNR+1); xy代表 01, 23, 45 or 67, 取决于所选择的PWM通道.</p> <p>对于边沿对齐类型:</p> <ul style="list-style-type: none"> <li>占空比 = (CMR+1)/(CNR+1).</li> <li>CMR &gt;= CNR: PWM 输出总是高.</li> <li>CMR &lt; CNR: PWM低脉冲宽度= (CNR-CMR) 单位; PWM高脉冲宽度 = (CMR+1) 单位.</li> <li>CMR = 0: PWM低脉冲宽度= (CNR) 单位; PWM 高脉冲宽度 = 1 单位</li> </ul> <p>对于中心对齐类型(M05xxDN/DE)</p> <ul style="list-style-type: none"> <li>占空比= [(2 x CMR) + 1]/[2 x (CNR+1)].</li> <li>CMR &gt; CNR: PWM 输出总是高.</li> </ul>

- |  |  |
|--|--|
|  | <ul style="list-style-type: none"><li>● CMR &lt;= CNR: PWM 低脉冲宽度 = <math>2 \times (\text{CNR}-\text{CMR}) + 1</math>单位; PWM 高脉冲宽度 = <math>(2 \times \text{CMR}) + 1</math>单位.</li><li>● CMR = 0: PWM 低脉冲宽度 = <math>2 \times \text{CNR} + 1</math>单位; PWM 高脉冲宽度 = 1单位<br/>(单位 =一个PWM时钟周期)</li></ul> <p>注: CMR写入数据后将在下一个PWM周期生效.</p> |
|--|--|

**PWM 数据寄存器 3-0 (PDR 3-0)**

寄存器	偏移量	R/W	描述	复位后的值
PDR0	PWMA_BA0+0x14	R	PWM 数据寄存器0	0x0000_0000
	PWMB_BA0+0x14			
PDR1	PWMA_BA0+0x20	R	PWM数据寄存器1	0x0000_0000
	PWMB_BA0+0x20			
PDR2	PWMA_BA0+0x2C	R	PWM数据寄存器2	0x0000_0000
	PWMB_BA0+0x2C			
PDR3	PWMA_BA0+0x38	R	PWM数据寄存器3	0x0000_0000
	PWMB_BA0+0x38			

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
PDR[15:8]							
7	6	5	4	3	2	1	0
PDR[7:0]							

Bits	描述	
[31:16]	保留	保留
[15:0]	PDRx	<b>PWM 数据寄存器</b> 用户可以查询PDR得到16位计数器的当前值。



### PWM 中断使能寄存器 (PIER)

寄存器	偏移量	R/W	描述	复位后的值
PIER	PWMA_BA+0x40	R/W	PWM 中断使能寄存器	0x0000_0000
	PWMB_BA+0x40			

31	30	29	28	27	26	25	24
保留						INT23DTYPE	INT01DTYPE
23	17	16	20	19	18	17	16
保留						INT23TYPE	INT01TYPE
15	14	13	12	11	10	9	8
保留				PWMDIE3	PWMDIE2	PWMDIE1	PWMDIE0
7	6	5	4	3	2	1	0
保留				PWMIE3	PWMIE2	PWMIE1	PWMIE0

Bits	描述	
[31:26]	保留	保留
[25]	INT23DTYPE	<p><b>PWM23 工作中断类型选择 (A组的PWM2 和 PWM3 对和B组的 PWM6 和 PWM7 对) (M05xxDN/DE )</b></p> <p>0 = PWM计数器下数时如果其值等于CMRx寄存器的值, PWMDIFx 将被设。如果此时 PWM触发ADC功能使能(PWMxDTEN=1), 将触发ADC开始转换</p> <p>1 = PWM计数器上数时如果其值等于CMRx寄存器的值, PWMDIFx 将被设。如果此时 PWM触发ADC功能使能(PWMxDTEN=1), 将触发ADC开始转换</p> <p><b>注:</b>只有在中心对齐模式下才可以设置 INT23DTYPE = 1</p>
[24]	INT01DTYPE	<p><b>PWM01 工作中断类型选择 (A组的PWM0 和 PWM1 对和B组的 PWM4 和 PWM5 对) (M05xxDN/DE )</b></p> <p>0 = PWM计数器下数时如果其值等于CMRx寄存器的值, PWMDIFx 将被设。如果此时 PWM触发ADC功能使能(PWMxDTEN=1), 将触发ADC开始转换</p> <p>1 = PWM计数器上数时如果其值等于CMRx寄存器的值, PWMDIFx 将被设。如果此时 PWM触发ADC功能使能(PWMxDTEN=1), 将触发ADC开始转换</p> <p><b>注:</b>只有在中心对齐模式下才可以设置 INT01DTYPE = 1</p>
[23:18]	保留	保留
[17]	INT23TYPE	<p><b>PWM23 周期中断类型选择 (A组的PWM2 和 PWM3 对和B组的 PWM6 和 PWM7 对) (M05xxDN/DE )</b></p> <p>0 = PWM计数器的值等于0时, PWMIFx 将被置。如果此时 PWM触发ADC功能使能(PWMxTEN=1), ADC开始转换</p> <p>1 = PWM 计数器的值等于 CNRx 时, PWMIFx 将被置。如果此时 PWM 触发 ADC 功能使能(PWMxTEN=1), ADC 开始转换</p> <p><b>Note:</b> 只有PWM工作在中心对齐模式时, 才可以设置INT23TYPE = 1</p>
[16]	INT01TYPE	<b>PWM01 周期中断类型选择 (A组的PWM0 和 PWM1 对和B组的 PWM4 和 PWM5 对) (M05xxDN/DE )</b>

		<p>0 = PWM计数器的值等于0时，PWMIFx 将被置。如果此时PWM触发ADC功能使能(PWMxTEN=1)，ADC开始转换</p> <p>1 = PWM 计数器的值等于 CNRx 时，PWMIFx 将被置。如果此时 PWM 触发 ADC 功能使能(PWMxTEN=1)，ADC 开始转换</p> <p><b>Note:</b> 只有 PWM 工作在中心对齐模式时，才可以设置 INT01TYPE = 1.</p>
[15:12]	保留	保留
[11]	<b>PWMDIE3</b>	<p><b>PWM 通道 3 工作中断使能控制 (M05xxDN/DE )</b></p> <p>0 = 关闭PWM 通道 3 工作中断 .</p> <p>1 = 使能 PWM 通道 3 工作中断.</p>
[10]	<b>PWMDIE2</b>	<p><b>PWM 通道 2 工作中断使能控制 (M05xxDN/DE )</b></p> <p>0 = 关闭PWM 通道 2 工作中断 .</p> <p>1 = 使能 PWM 通道 2 工作中断.</p>
[9]	<b>PWMDIE1</b>	<p><b>PWM 通道 1 工作中断使能控制 (M05xxDN/DE )</b></p> <p>0 = 关闭PWM 通道 1 工作中断 .</p> <p>1 = 使能 PWM 通道 1 工作中断.</p>
[8]	<b>PWMDIE0</b>	<p><b>PWM 通道 0 工作中断使能控制 (M05xxDN/DE )</b></p> <p>0 = 关闭PWM 通道 0 工作中断 .</p> <p>1 = 使能 PWM 通道 0 工作中断.</p>
[7:4]	保留	保留
[3]	<b>PWMIE3</b>	<p><b>PWM 通道 3 周期中断使能控制</b></p> <p>0 = 关闭PWM 通道 3 周期中断 .</p> <p>1 = 使能 PWM 通道 3 周期中断.</p>
[2]	<b>PWMIE2</b>	<p><b>PWM 通道 2 周期中断使能控制</b></p> <p>0 = 关闭PWM 通道 2 周期中断 .</p> <p>1 = 使能 PWM 通道 2 周期中断.</p>
[1]	<b>PWMIE1</b>	<p><b>PWM 通道 1 周期中断使能控制</b></p> <p>0 = 关闭PWM 通道 1 周期中断 .</p> <p>1 = 使能 PWM 通道 1 周期中断.</p>
[0]	<b>PWMIE0</b>	<p><b>PWM 通道 0 周期中断使能控制</b></p> <p>0 = 关闭PWM 通道 0 周期中断 .</p> <p>1 = 使能 PWM 通道 0 周期中断.</p>



### PWM 中断标志寄存器(PIIR)

寄存器	偏移量	R/W	描述	复位后的值
PIIR	PWMA_BA+0x44	R/W	PWM A组中断标志寄存器	0x0000_0000
	PWMB_BA+0x44			

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				PWMDIF3	PWMDIF2	PWMDIF1	PWMDIF0
7	6	5	4	3	2	1	0
保留				PWMIF3	PWMIF2	PWMIF1	PWMIF0

Bits	描述	
[31:12]	保留	保留
[11]	PWMDIF3	<p><b>PWM 通道 3 工作中断状态 (M05xxDN/DE)</b></p> <p>当 INTDTYPE23 = 0时， PWM通道3下数计数器的值等于CMR3该位被置.</p> <p>当 INTDTYPE23 = 1时， PWM通道3上数计数器的值等于CMR3该位被置.</p> <p>注: 写 1 清除该位.</p>
[10]	PWMDIF2	<p><b>PWM 通道 2 工作中断状态 (M05xxDN/DE)</b></p> <p>当 INTDTYPE23 = 0时， PWM通道2下数计数器的值等于CMR2位被置.</p> <p>当 INTDTYPE23 = 1时， PWM通道2上数计数器的值等于CMR2该位被置.</p> <p>注: 写 1 清除该位.</p>
[9]	PWMDIF1	<p><b>PWM 通道 1 工作中断状态 (M05xxDN/DE)</b></p> <p>当 INTDTYPE01 = 0时， PWM通道1下数计数器的值等于CMR1该位被置.</p> <p>当 INTDTYPE01 = 1时， PWM通道1上数计数器的值等于CMR1该位被置.</p> <p>注: 写 1 清除该位.</p>
[8]	PWMDIF0	<p><b>PWM 通道 0 工作中断状态 (M05xxDN/DE)</b></p> <p>当 INTDTYPE01 = 0时， PWM通道0下数计数器的值等于CMR0该位被置.</p> <p>当 INTDTYPE01 = 1时， PWM通道0上数计数器的值等于CMR0该位被置.</p> <p>注: 写 1 清除该位.</p>
[7:4]	保留	保留
[3]	PWMIF3	<p><b>PWM 通道3周期中断状态</b></p> <p><b>M05xxBN中:</b></p> <p>如果PWM 通道3周期中断使能位(PWMIE3)为"1"时， 当PWM通道3向下计数至0时， 硬件将</p>

		<p>该位置1。软件写1清该位.</p> <p><b>M05xxDN/DE中:</b></p> <p>INTTYPE23 = 0, 当PWM通道3向下计数至0时, 硬件将该位置1。</p> <p>INTTYPE23 = 1, 当PWM通道3向上计数至CNR3时, 硬件将该位置1。</p> <p>注: 软件写1清该位.</p>
[2]	PWMIF2	<p><b>PWM 通道2周期中断状态</b></p> <p><b>M05xxBN中:</b></p> <p>如果PWM 通道2周期中断使能位(PWMIE2)为"1"时, 当PWM通道2向下计数至0时, 硬件将该位置1。软件写1清该位.</p> <p><b>M05xxDN/DE中:</b></p> <p>INTTYPE23 = 0, 当PWM通道2向下计数至0时, 硬件将该位置1。</p> <p>INTTYPE23 = 1, 当PWM通道2向上计数至CNR2时, 硬件将该位置1。</p> <p>注: 软件写1清该位.</p>
[1]	PWMIF1	<p><b>PWM 通道1周期中断状态</b></p> <p><b>M05xxBN中:</b></p> <p>如果PWM 通道1周期中断使能位(PWMIE1)为"1"时, 当PWM通道1向下计数至0时, 硬件将该位置1。软件写1清该位.</p> <p><b>M05xxDN/DE中:</b></p> <p>INTTYPE01 = 0, 当PWM通道1向下计数至0时, 硬件将该位置1。</p> <p>INTTYPE01 = 1, 当PWM通道1向上计数至CNR1时, 硬件将该位置1。</p> <p>注: 软件写1清该位.</p>
[0]	PWMIF0	<p><b>PWM 通道0周期中断状态</b></p> <p><b>M05xxBN中:</b></p> <p>如果PWM 通道0周期中断使能位(PWMIE0)为"1"时, 当PWM通道0向下计数至0时, 硬件将该位置1。软件写1清该位.</p> <p><b>M05xxDN/DE中:</b></p> <p>INTTYPE01 = 0, 当PWM通道0向下计数至0时, 硬件将该位置1。</p> <p>INTTYPE01 = 1, 当PWM通道0向上计数至CNR0时, 硬件将该位置1。</p> <p>注: 软件写1清该位.</p>

**捕获控制寄存器0(CCR0)**

寄存器	偏移量	R/W	描述				复位后的值
CCR0	PWMA_BA+0x50	R/W	PWM 捕获控制寄存器0				0x0000_0000
	PWMB_BA+0x50						

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

Bits	描述	
[31:24]	保留	保留
[23]	CFLRI1	<b>CFLR1锁存指示位</b> PWM组输入通道1有下降沿转变时， PWM向下计数器的值锁存到CFLR1寄存器，同时该位由硬件置位。 注：该位写1清0。
[22]	CRLRI1	<b>CRLR1锁存指示位</b> PWM组输入通道1有上升沿转变时， PWM向下计数器的值锁存到CFLR1寄存器，同时该位由硬件置位。 注：该位写1清0。
[5]	保留	保留
[20]	CAPIF1	<b>捕捉通道1中断标志</b> 如果PWM组通道1上升沿锁存中断使能(CRL_IE1=1)， PWM组通道1有上升沿转变将使CAPIF1为高；同样，如果下降沿锁存中断使能(CFL_IE1=1)，下降沿转变将使CAPIF1为高。 注：该位写1清0。
[19]	CAPCH1EN	<b>捕捉器通道1使能控制</b> 1 = 使能PWM组通道1的捕捉功能。 0 = 禁用PWM组通道1的捕捉功能 <b>注1：</b> 使能时，捕捉锁存PWM计数器并保存到CRLR（上升沿锁定）和CFLR（上升沿锁定）。 <b>注2：</b> 禁用时，捕捉器不更新CRLR和CFLR，并禁用PWM组通道1捕获中断。
[18]	CFL_IE1	<b>PWM 组通道1下降沿锁存中断使能控制</b> 1 = 使能下降沿锁存中断

		0 = 禁用下降沿锁存中断  注：使能时，如果捕捉器检测到PWM组通道1有下降沿转变，捕捉器产生中断。
[17]	<b>CRL_IE1</b>	<b>PWM 组通道1上升沿锁存中断使能控制</b> 1 = 使能上升沿锁存中断 0 = 禁用上升沿锁存中断  注：使能时，如果捕捉器检测到PWM组通道1有上升沿转变，捕捉器产生中断
[16]	<b>INV1</b>	<b>捕获通道1反向使能控制</b> 1 = 反向打开。I/O脚上的信号输入到捕获器前反转再输入。 0 = 反向关闭
[15:8]	保留	保留
[7]	<b>CFLRIO</b>	<b>CFLR0锁存指示位</b> 在PWM组输入通道0有下降沿转变时， PWM向下计数器的值锁存到CFLR0寄存器，同时该位由硬件置位。  注：该位写1清0.
[6]	<b>CRLRIO</b>	<b>CRLR0锁存指示位</b> 在PWM组输入通道0有上升沿转变时， PWM向下计数器的值锁存到CFLR0寄存器，同时该位由硬件置位。  注：该位写1清0.
[5]	保留	保留
[4]	<b>CAPIFO</b>	<b>捕捉通道0中断标志</b> 如果PWM组通道0上升沿锁存中断使能(CRL_IE0=1)， PWM组通道0有上升沿转变将使CAPIFO为高；同样，如果下降沿锁存中断使能(CFL_IE0=1)， 下降沿转变将使CAPIFO为高。  注：该位写1清0.
[3]	<b>CAPCH0EN</b>	<b>捕捉器通道0使能控制</b> 1 = 使能PWM组通道0的捕捉功能. 0 = 禁用PWM组通道0的捕捉功能  注1：使能时，捕捉锁存PWM计数器并保存到CRLR（上升沿锁定）和CFLR（上升沿锁定）。  注1：禁用时，捕捉器不更新CRLR和CFLR，并禁用PWM组通道0捕获中断。
[2]	<b>CFL_IE0</b>	<b>PWM 组通道0下降沿锁存中断使能控制</b> 1 = 使能下降沿锁存中断 0 = 禁用下降沿锁存中断  注：使能时，如果捕捉器检测到PWM组通道0有下降沿转变，捕捉器产生中断。
[1]	<b>CRL_IE0</b>	<b>PWM 组通道0上升沿锁存中断使能控制</b> 1 = 使能上升沿锁存中断 0 = 禁用上升沿锁存中断  注：使能时，如果捕捉器检测到PWM组通道0有上升沿转变，捕捉器产生中断
[0]	<b>INV0</b>	<b>捕获通道0反向使能控制</b> 1 = 反向打开。I/O脚上的信号输入到捕获器前反转再输入。 0 = 反向关闭

**捕获控制寄存器2(CCR2)**

寄存器	偏移量	R/W	描述	复位后的值
CCR2	PWMA_BA+0x54	R/W	PWM 捕获控制寄存器2	0x0000_0000
	PWMB_BA+0x54			

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

Bits	描述	
[31:24]	保留	保留
[23]	CFLRI3	<b>CFLR3锁存指示位</b> PWM组输入通道3有下降沿转变时， PWM向下计数器的值锁存到CFLR3寄存器， 同时该位由硬件置位。 注：该位写1清0。
[22]	CRLRI3	<b>CRLR3锁存指示位</b> PWM组输入通道3有上升沿转变时， PWM向下计数器的值锁存到CFLR3寄存器， 同时该位由硬件置位。 注：该位写1清0。
[21]	保留	保留
[20]	CAPIF3	<b>捕捉通道3中断标志</b> 如果PWM组通道3上升沿锁存中断使能(CRL_IE3=1)， PWM组通道3有上升沿转变将使CAPIF3为高； 同样，如果下降沿锁存中断使能(CFL_IE3=1)， 下降沿转变将使CAPIF3为高。 注：该位写1清0。
[19]	CAPCH3EN	<b>捕捉器通道3使能控制</b> 1 = 使能PWM组通道3的捕捉功能。 0 = 禁用PWM组通道3的捕捉功能 <b>注1：</b> 使能时，捕捉锁存PWM计数器并保存到CRLR（上升沿锁定）和CFLR（上升沿锁定）。 <b>注2：</b> 禁用时，捕捉器不更新CRLR和CFLR，并禁用PWM组通道3捕获中断。
[18]	CFL_IE3	<b>PWM 组通道3下降沿锁存中断使能控制</b> 1 = 使能下降沿锁存中断

		<p>0 = 禁用下降沿锁存中断  <b>注:</b> 使能时, 如果捕捉器检测到PWM组通道3有下降沿转变, 捕捉器产生中断.</p>
[17]	<b>CRL_IE3</b>	<p><b>PWM 组通道3上升沿锁存中断使能控制</b>  1 = 使能上升沿锁存中断  0 = 禁用上升沿锁存中断  <b>注:</b> 使能时, 如果捕捉器检测到PWM组通道3有上升沿转变, 捕捉器产生中断</p>
[16]	<b>INV3</b>	<p><b>捕获通道3反向使能控制</b>  1 = 反向打开。I/O脚上的信号输入到捕获器前反转再输入。  0 = 反向关闭</p>
[15:8]	保留	保留
[7]	<b>CFLRI2</b>	<p><b>CFLR2锁存指示位</b>  在PWM组输入通道2有下降沿转变时, PWM向下计数器的值锁存到CFLR2寄存器, 同时该位由硬件置位。  <b>注:</b> 该位写1清0.</p>
[6]	<b>CRLRI2</b>	<p><b>CRLR2锁存指示位</b>  在PWM组输入通道2有上升沿转变时, PWM向下计数器的值锁存到CFLR2寄存器, 同时该位由硬件置位。  <b>注:</b> 该位写1清0.</p>
[5]	保留	保留
[4]	<b>CAPIF2</b>	<p><b>捕捉通道2中断标志</b>  如果PWM组通道2上升沿锁存中断使能(CRL_IE2=1), PWM组通道2有上升沿转变将使CAPIF2为高; 同样, 如果下降沿锁存中断使能(CFL_IE2=1), 下降沿转变将使CAPIF2为高.  <b>注:</b> 该位写1清0.</p>
[3]	<b>CAPCH2EN</b>	<p><b>捕捉器通道2使能控制</b>  1 = 使能PWM组通道2的捕捉功能.  0 = 禁用PWM组通道2的捕捉功能  <b>注1:</b> 使能时, 捕捉锁存PWM计数器并保存到CRLR(上升沿锁定)和CFLR(上升沿锁定).  <b>注2:</b> 禁用时, 捕捉器不更新CRLR和CFLR, 并禁用PWM组通道0捕获中断.</p>
[2]	<b>CFL_IE2</b>	<p><b>PWM 组通道2下降沿锁存中断使能控制</b>  1 = 使能下降沿锁存中断  0 = 禁用下降沿锁存中断  <b>注:</b> 使能时, 如果捕捉器检测到PWM组通道2有下降沿转变, 捕捉器产生中断.</p>
[1]	<b>CRL_IE2</b>	<p><b>PWM 组通道2上升沿锁存中断使能控制</b>  1 = 使能上升沿锁存中断  0 = 禁用上升沿锁存中断  <b>注:</b> 使能时, 如果捕捉器检测到PWM组通道2有上升沿转变, 捕捉器产生中断</p>
[0]	<b>INV2</b>	<p><b>捕获通道2反向使能控制</b>  1 = 反向打开。I/O脚上的信号输入到捕获器前反转再输入。  0 = 反向关闭</p>

捕捉器上升沿锁存寄存器3-0 (CRLR3-0)

寄存器	偏移量	R/W	描述	复位后的值
CRLR0	PWMA_BA+0x58	R	PWM 捕捉上升沿锁存寄存器(通道 0)	0x0000_0000
	PWMB_BA+0x58			
CRLR1	PWMA_BA+0x60	R	PWM 捕捉上升沿锁存寄存器(通道1)	0x0000_0000
	PWMB_BA+0x60			
CRLR2	PWMA_BA+0x68	R	PWM 捕捉上升沿锁存寄存器(通道2)	0x0000_0000
	PWMB_BA+0x68			
CRLR3	PWMA_BA+0x70	R	PWM 捕捉上升沿锁存寄存器(通道3)	0x0000_0000
	PWMB_BA+0x70			

注: 当CPU时钟低于PWM/Capture时钟时, 写CRLRx不保证正确.

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
CRLRx [15:8]							
7	6	5	4	3	2	1	0
CRLRx [7:0]							

Bits	描述	
[31:16]	保留	保留
[15:0]	CRLRx	捕捉上升沿锁存寄存器 当通道0/1/2/3 有上升沿时转变时, 锁存PWM计数器的值到这个寄存器.

**捕捉下降沿锁存寄存器3-0 (CFLR3-0)**

寄存器	偏移量	R/W	描述	复位后的值
CFLR0	PWMA_BA+0x5C	R	PWM 捕捉下降沿锁存寄存器(通道 0)	0x0000_0000
	PWMB_BA+0x5C			
CFLR1	PWMA_BA+0x64	R	PWM 捕捉下降沿锁存寄存器(通道1)	0x0000_0000
	PWMB_BA+0x64			
CFLR2	PWMA_BA+0x6C	R	PWM 捕捉下降沿锁存寄存器(通道2)	0x0000_0000
	PWMB_BA+0x6C			
CFLR3	PWMA_BA+0x74	R	PWM 捕捉下降沿锁存寄存器(通道3)	0x0000_0000
	PWMB_BA+0x74			

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
CFLRx [15:8]							
7	6	5	4	3	2	1	0
CFLRx [7:0]							

Bits	描述		
[31:16]	保留	保留	
[15:0]	CFLRx	捕捉下降沿锁存寄存器 当通道0/1/2/3 有下降沿转变时，锁存PWM计数器的值到这个寄存器.	

捕获输入使能寄存器(CAPENR)

寄存器	偏移量	R/W	描述	复位后的值
CAPENR	PWMA_BA+0x78	R/W	PWM 捕捉输入0~3 使能寄存器	0x0000_0000
	PWMB_BA+0x78			

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				CAPENR			

Bits	描述
[3:0]	<p><b>CAPENR</b></p> <p><b>捕捉输入使能寄存器</b> 有4组捕捉输入。Bit0~Bit3 用于控制每个输入的打开 / 关闭。 0 = 关闭捕获输入(PWMx 复用脚输入对捕捉器不产生影响) 1 = 打开捕获输入 (PWMx 复用脚将影响捕捉器功能.)</p> <p><b>CAPENR</b></p> <p><b>Bit 3210 用于 PWM A组</b> Bit xxx1 → 使能捕捉通道0 从 P2.0或者P4.0 输入。通过设置多功能引脚寄存器用户只能选择1个引脚 Bit xx1x → 使能捕捉通道1 从 P2.1或者p4.1 输入。通过设置多功能引脚寄存器用户只能选择1个引脚 Bit x1xx → 使能捕捉通道2 从 P2.2或者P4.2 输入。通过设置多功能引脚寄存器用户只能选择1个引脚 Bit 1xxx → 使能捕捉通道3 从 P2.3或者P4.3 输入。通过设置多功能引脚寄存器用户只能选择1个引脚</p> <p><b>Bit 3210 用于 PWM B组</b> Bit xxx1 → 使能捕捉通道0 从 P2.4 输入 Bit xx1x → 使能捕捉通道1 从 P2.5 输入 Bit x1xx → 使能捕捉通道2 从 P2.6 输入 Bit 1xxx → 使能捕捉通道3 从 P2.7 输入</p>

PWM输出使能寄存器 (POE)

寄存器	偏移量	R/W	描述	复位后的值
POE	PWMA_BA+0x7C	R/W	PWM 输出使能寄存器（通道0~3）	0x0000_0000
	PWMB_BA+0x7C			

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				PWM3	PWM2	PWM1	PWM0

Bits	描述	
[31:4]	保留	保留
[3]	PWM3	<b>PWM 通道3输出使能寄存器</b> 1 = 使能PWM通道3输出 0 = 禁止PWM通道3输出 注: GPIO相应管脚必须切换到PWM功能
[2]	PWM2	<b>PWM 通道2输出使能寄存器</b> 1 = 使能PWM通道2输出 0 = 禁止PWM通道2输出 注: GPIO相应管脚必须切换到PWM功能
[1]	PWM1	<b>PWM 通道1输出使能寄存器</b> 1 = 使能PWM通道1输出 0 = 禁止PWM通道1输出 注: GPIO相应管脚必须切换到PWM功能
[0]	PWM0	<b>PWM 通道0输出使能寄存器</b> 1 = 使能PWM通道0输出 0 = 禁止PWM通道0输出 注: GPIO相应管脚必须切换到PWM功能

**PWM 触发控制寄存器 (TCON)**

寄存器	偏移量	R/W	描述	复位后的值
TCON	PWMA_BA+0x80	R/W	PWM 通道0~3触发控制寄存器 (M05xxDN/DE)	0x0000_0000
	PWMB_BA+0x80			

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				PWM3DTEN	PWM2DTEN	PWM1DTEN	PWM0DTEN
7	6	5	4	3	2	1	0
保留				PWM3TEN	PWM2TEN	PWM1TEN	PWM0TEN

Bits	描述	
[31:12]	保留	保留.
[11]	PWM3DTEN	<p><b>PWM 通道3工作触发 ADC 使能控制 (M05xxDN/DE )</b></p> <p>0 = 关闭 PWM 通道 3 工作触发 ADC 功能. 1 = 使能 PWM 通道 3 工作触发 ADC 功能.</p> <p>PWM 操作在边沿对齐类型时，当 PWM 通道3计数器下数等于 CMR3 时，使能该位可以使 PWM 触发 ADC 开始转换</p> <p>PWM 操作在中心对齐类新时，当 PWM 通道3计数器上数/下数(根据 INT23DTYPE 的设定)等于 CMR3 时，使能该位可以使 PWM 触发 ADC 开始转换.</p>
[10]	PWM2DTEN	<p><b>PWM 通道2工作触发 ADC 使能控制 (M05xxDN/DE )</b></p> <p>0 = 关闭 PWM 通道 2 工作触发 ADC 功能. 1 = 使能 PWM 通道 2 工作触发 ADC 功能.</p> <p>PWM 操作在边沿对齐类型时，当 PWM 通道2计数器下数等于 CMR2 时，使能该位可以使 PWM 触发 ADC 开始转换</p> <p>PWM 操作在中心对齐类新时，当 PWM 通道2计数器上数/下数(根据 INT23DTYPE 的设定)等于 CMR2 时，使能该位可以使 PWM 触发 ADC 开始转换.</p>
[9]	PWM1DTEN	<p><b>PWM 通道1工作触发 ADC 使能控制 (M05xxDN/DE )</b></p> <p>0 = 关闭 PWM 通道 1 工作触发 ADC 功能. 1 = 使能 PWM 通道 1 工作触发 ADC 功能.</p> <p>PWM 操作在边沿对齐类型时，当 PWM 通道1计数器下数等于 CMR1 时，使能该位可以使 PWM 触发 ADC 开始转换</p> <p>PWM 操作在中心对齐类新时，当 PWM 通道1计数器上数/下数(根据 INT01DTYPE 的设定)等于 CMR1 时，使能该位可以使 PWM 触发 ADC 开始转换.</p>
[8]	PWM0DTEN	<b>PWM 通道0工作触发 ADC 使能控制 (M05xxDN/DE )</b>

		0 = 关闭 PWM 通道 0 工作触发 ADC 功能. 1 = 使能 PWM 通道 0 工作触发 ADC 功能.  PWM 操作在边沿对齐类型时, 当 PWM 通道0计数器下数等于 CMR0 时, 使能该位可以使 PWM 触发 ADC 开始转换  PWM 操作在中心对齐类新时, 当 PWM 通道0计数器上数/下数(根据 INT01DTYPE 的设定)等于 CMR0 时, 使能该位可以使 PWM 触发 ADC 开始转换.
[7:4]	保留	保留.
[3]	PWM3TEN	<b>PWM通道3 周期触发ADC只能控制 (M05xxDN/DE )</b> 0 = 关闭 PWM 通道 3 周期触发 ADC 功能. 1 = 使能 PWM 通道 3 周期触发 ADC 功能.  PWM 操作在边沿对齐类型时, 当 PWM 通道3计数器下数等于 0 时, 使能该位可以使 PWM 触发 ADC 开始转换  PWM 操作在中心对齐类新时, 当 PWM 通道3计数器上数等于(CNR3+1)或者下数等于 0 时(根据 INT23DTYPE 的设定), 使能该位可以使 PWM 触发 ADC 开始转换.
[2]	PWM2TEN	<b>PWM通道2 周期触发ADC只能控制 (M05xxDN/DE )</b> 0 = 关闭 PWM 通道 2 周期触发 ADC 功能. 1 = 使能 PWM 通道 2 周期触发 ADC 功能.  PWM 操作在边沿对齐类型时, 当 PWM 通道2计数器下数等于 0 时, 使能该位可以使 PWM 触发 ADC 开始转换  PWM 操作在中心对齐类新时, 当 PWM 通道2计数器上数等于(CNR2+1)或者下数等于 0 时(根据 INT23DTYPE 的设定), 使能该位可以使 PWM 触发 ADC 开始转换.
[1]	PWM1TEN	<b>PWM通道1 周期触发ADC只能控制 (M05xxDN/DE )</b> 0 = 关闭 PWM 通道 1 周期触发 ADC 功能. 1 = 使能 PWM 通道 1 周期触发 ADC 功能.  PWM 操作在边沿对齐类型时, 当 PWM 通道1计数器下数等于 0 时, 使能该位可以使 PWM 触发 ADC 开始转换  PWM 操作在中心对齐类新时, 当 PWM 通道1计数器上数等于(CNR1+1)或者下数等于 0 时(根据 INT01DTYPE 的设定), 使能该位可以使 PWM 触发 ADC 开始转换.
[0]	PWM0TEN	<b>PWM通道0周期触发ADC只能控制 (M05xxDN/DE )</b> 0 = 关闭 PWM 通道 0 周期触发 ADC 功能. 1 = 使能 PWM 通道 0 周期触发 ADC 功能.  PWM 操作在边沿对齐类型时, 当 PWM 通道0计数器下数等于 0 时, 使能该位可以使 PWM 触发 ADC 开始转换  PWM 操作在中心对齐类新时, 当 PWM 通道0计数器上数等于(CNR0+1)或者下数等于 0 时(根据 INT01DTYPE 的设定), 使能该位可以使 PWM 触发 ADC 开始转换.

PWM 触发状态寄存器 (TSTATUS)

寄存器	偏移量	R/W	描述	复位后的值
TSTATUS	PWMA_BA+0x84	R/W	PWM 触发状态寄存器 (M05xxDN/DE )	0x0000_0000
	PWMB_BA+0x84			

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				PWM3TF	PWM2TF	PWM1TF	PWM0TF

Bits	描述	
[31:4]	保留	保留.
[3]	PWM3TF	<p><b>PWM 通道3触发 ADC 标志 (M05xxDN/DE)</b>  当PWM通道3触发ADC时，该位由硬件置为1。此时如果ADC选择由PWM触发转换，ADC转换将开始。  注：该位写1清0。</p>
[2]	PWM2TF	<p><b>PWM 通道2触发 ADC 标志 (M05xxDN/DE)</b>  当PWM通道2触发ADC时，该位由硬件置为1。此时如果ADC选择由PWM触发转换，ADC转换将开始。  注：该位写1清0。</p>
[1]	PWM1TF	<p><b>PWM 通道1触发 ADC 标志 (M05xxDN/DE)</b>  当PWM通道1触发ADC时，该位由硬件置为1。此时如果ADC选择由PWM触发转换，ADC转换将开始。  注：该位写1清0。</p>
[0]	PWM0TF	<p><b>PWM 通道0触发 ADC 标志 (M05xxDN/DE)</b>  当PWM通道0触发ADC时，该位由硬件置为1。此时如果ADC选择由PWM触发转换，ADC转换将开始。  注：该位写1清0。</p>

**PWM 同步控制寄存器 (PSCR)**

寄存器	偏移量	R/W	描述	复位后的值
PSCR	PWMA_BA+0x98	R/W	PWM 同步控制寄存器 (M05xxDN/DE)	0x0000_0000
	PWMB_BA+0x98			

31	30	29	28	27	26	25	24
保留							PSSEN3
23	22	21	20	19	18	17	16
保留							PSSEN2
15	14	13	12	11	10	9	8
保留							PSSEN1
7	6	5	4	3	2	1	0
保留							PSSEN0

Bits	描述	
[31:25]	保留	保留.
[24]	PSSEN3	<b>PWM 通道3 同步开始使能控制 (M05xxDN/DE )</b> 0 = 关闭PWM通道3 定时器 同步开始. 1 = 使能PWM通道3 定时器 同步开始. 如果该位设为1, 当软件写1到 PWMA 组的 CH0EN(PCR[0])启动定时器时, PWM 组通道3的定时器将和 PWMA 组通道0的定时器同步启动
[23:17]	保留	保留.
[16]	PSSEN2	<b>PWM 通道2 同步开始使能控制 (M05xxDN/DE )</b> 0 = 关闭PWM通道2 定时器 同步开始. 1 = 使能PWM通道2 定时器 同步开始. 如果该位设为1, 当软件写1到 PWMA 组的 CH0EN(PCR[0])启动定时器时, PWM 组通道2的定时器将和 PWMA 组通道0的定时器同步启动
[15:9]	保留	保留.
[8]	PSSEN1	<b>PWM 通道1 同步开始使能控制 (M05xxDN/DE )</b> 0 = 关闭PWM通道1 定时器 同步开始. 1 = 使能PWM通道1定时器 同步开始. 如果该位设为1, 当软件写1到 PWMA 组的 CH0EN(PCR[0])启动定时器时, PWM 组通道1的定时器将和 PWMA 组通道0的定时器同步启动
[7:1]	保留	保留.
[0]	PSSEN0	<b>PWM 通道0 同步开始使能控制 (M05xxDN/DE )</b> 0 = 关闭PWM通道0 定时器 同步开始.

		1 = 使能PWM通道0 定时器 同步开始。 如果该位设为1，当软件写1到 PWMA 组的 CH0EN(PCR[0])启动定时器时， PWM 组通道0的定时器将和 PWMA 组通道0的定时器同步启动
--	--	---

## 6.12 串行外设接口(SPI)控制器

### 6.12.1 概述

串行外设接口(SPI)是一个工作于全双工模式下的同步串行数据通讯协议。设备通过4线双向接口工作于主机/从机模式进行通讯。NuMicro™ M051系列包括最多2组SPI控制器，将从外设接收到的数据进行串并转换，或将要发送到外设的数据进行并串转换。每组SPI控制器都可被设置成主机；也可设置为被片外主机设备控制的从机。

### 6.12.2 特性

- 最多两组SPI控制器
- 支持主/从机模式
- 传输比特长度可配置
- 支持burst操作模式，在一次传输过程中，发送/接收最多一次可以传输两笔
- 提供FIFO缓存
- 支持MSB 或 LSB 优先传输
- 字节重排序功能
- 字节或字休眠模式
- 主机模式下支持两种可编程的串行时钟频率
- 从机模式下支持3线模式，没有从设备片选
- SPI时钟频率可以配置等于系统时钟频率

	M05xxBN	M05xxDN/DE
Variable Clock Function	●	-
Burst Mode	●	-
FIFO Mode	-	●
DIV_ONE Function	●	-
Width of DIVIDER	16 bits	8 bits
SPI Peripheral Clock Source	HCLK	HCLK or PLL

表 6-13 M05xxBN 和 M05xxDN/DE 功能差异列表(SPI)

### 6.12.3 SPI 框图

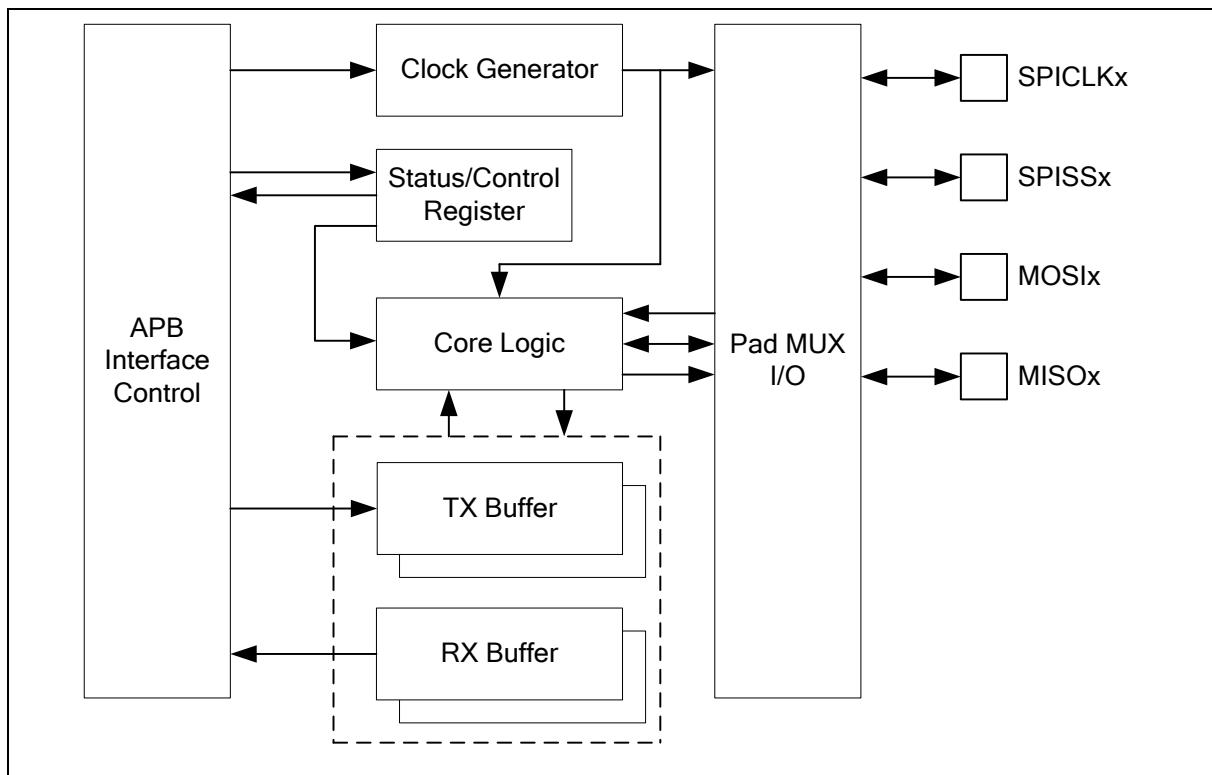


图 6-80 SPI 框图

### 6.12.4 基本配置

SPI 引脚功能在 P0\_MFP 和 P1\_MFP 寄存器中配置。SPI0 和 SPI1 外设时钟在 APBCLK[12] 和 APBCLK[13] 中使能。M05xxDN/DE 中，SPI 外设时钟源在 CLKSEL1[4] 和 CLKSEL1[5] 中选择。

## 6.12.5 SPI 功能描述

### 6.12.5.1 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 逻辑单元实现数据传输。SPI 总线时钟是SPICLK<sub>x</sub> 引脚上的时钟。.SPI 总线时钟频率计算M05xxBN 和 M05xxDN/DE是不同的。

#### M05xxBN中

M05xxBN SPI 主机模式， SPI 外设时钟频率由 DIVIDER (SPI\_DIVIDER[15:0]) 和 DIV\_ONE (SPI\_CNTRL2[0])的值决定。如果DIV\_ONE = 1， SPI 总线时钟频率=SPI 外设时钟频率= 系统时钟频率。当可变总线时钟频率功能关闭时， SPI 外设时钟频率= SPI 总线频率。如果VARCLK\_EN (SPI\_CTL[23]) =1 ， M05xxBN SPI 支持可变总线时钟。这种情况下， 根据 DIVIDER (SPI\_DIVIDER[15:0]) 和 DIVIDER2 (SPI\_DIVIDER[31:16])的设定，总线输出时钟频率可以编程为2个不同的频率中的1个。每个周期的时钟频率由SPI\_VARCLK 寄存器的值决定。

M05xxBN SPI 从机模式，片外主设备通过SPICLK引脚驱动总线时钟输入到该SPI 控制器。系统时钟作为SPI 外设时钟。SPI 外设时钟频率必须至少比SPI 总线时钟频率快5 倍以上.

#### M05xxDN/DE中

M05xxDN/DE SPI主机模式和从机模式， SPI 外设时钟频率由时钟源选择、 BCn (SPI\_CNTRL2[31]) 和时钟除频 (SPI\_DIVIDER[7:0]) 决定。CLKSEL1 寄存器的SPIx\_S 决定SPI外设的时钟源。时钟源可以是 HCLK 或者 PLL 输出。如果BCn = 0， SPI时钟频率的计算方法和以前的产品兼容。SPI\_DIVIDER 寄存器DIVIDER的值决定时钟频率的除数。

M05xxDN/DE SPI 主模式， SPI 外设时钟频率等于 SPI 总线时钟频率.

M05xxDN/DE SPI 从模式， SPI 总线时钟由片外主机提供。SPI从机外设时钟频率必须比主机输入的总线时钟频率快。无论主机还是从机， SPI 外设时钟频率不能比APB的时钟频率还快。

#### 主机/从机模式

SPI控制器可通过设置**SLAVE** 位(SPI\_CNTRL[18])被配置为主机或从机模式， 来与片外SPI从机或主机设备通讯。主机模式与从机模式下的应用框图如下所示。

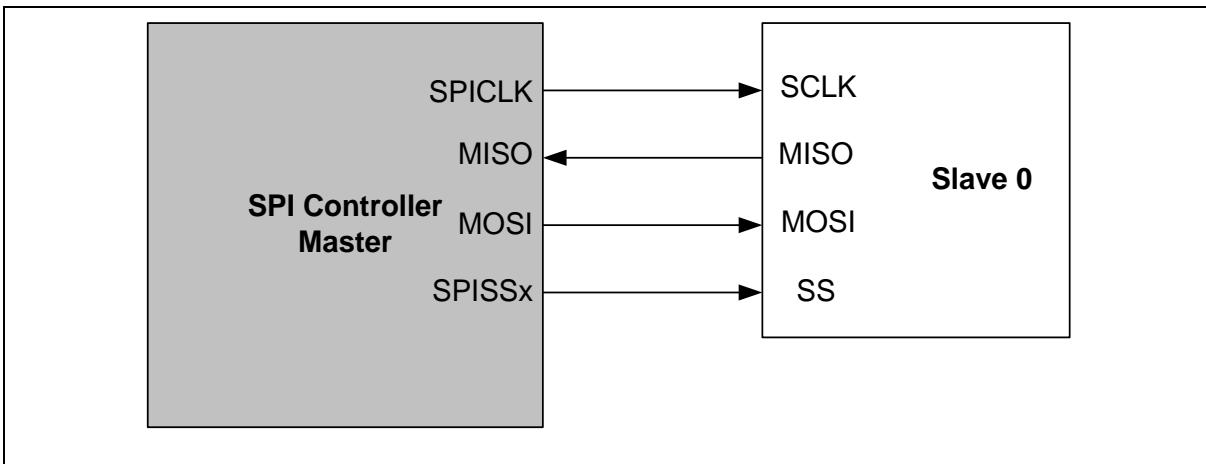


图6-81 SPI主机模式应用框图

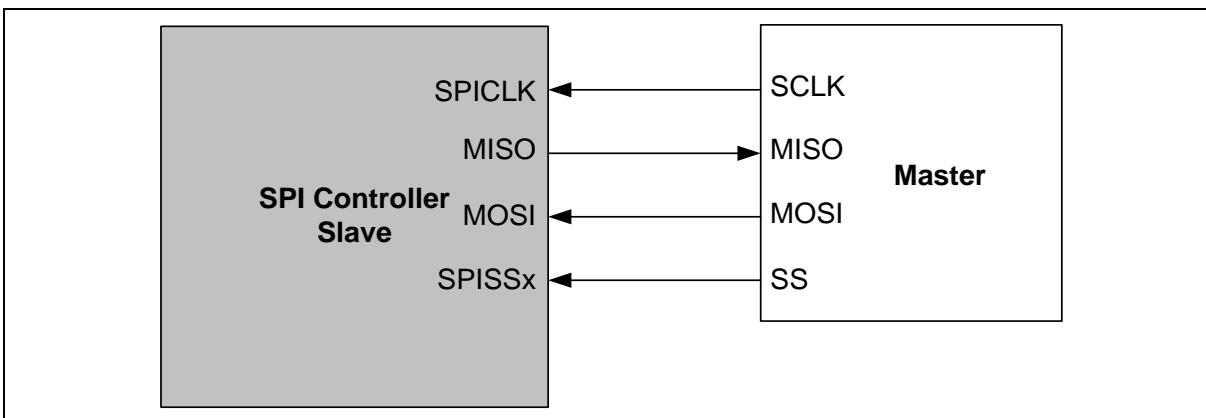


图6-82 SPI从机模式应用框图

### 时钟极性

在主机模式下，CLKP 位 (SPI\_CNTRL[11]) 定义总线时钟的空闲状态。如果 **CLKP = 1**，空闲状态下 SPICLK 输出高电平。**CLKP = 0** 时，SPICLK 在空闲状态下输出低电平。

### 发送/接收位长度

传输字的比特长度在 TX\_BIT\_LEN 位域(SPI\_CNTRL[7:3])中配置。对于发送和接收，一个传输字的比特长度可被配置为最多32位。

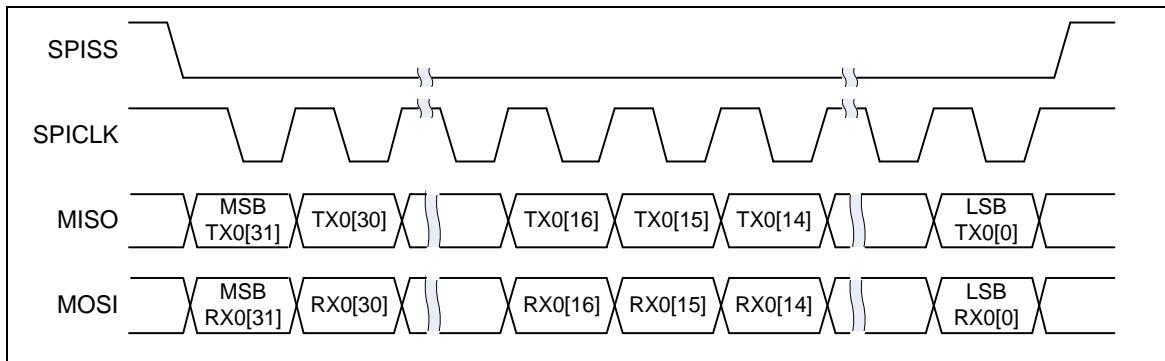


图6-83 一次传输32-Bit

### LSB 优先

LSB 位(SPI\_CNTRL[10]) 定义数据是从LSB还是从MSB开始发送/接收.

### 发送边沿

Tx\_NEG 位 (SPI\_CNTRL[2]) 定义数据发送是在串行时钟SPICLK的下降沿还是上升沿.

### 接收边沿

Rx\_NEG 位 (SPI\_CNTRL[1]) 定义数据接收是在串行时钟SPICLK的下降沿还是上升沿.

注：TX\_NEG 和 RX\_NEG的设定是互斥的。就是说不可以设定发送/接收都在同一个时钟沿

### 字休眠

在主机模式下，SP\_CYCLE (SPI\_CNTRL[15:12])的4位域负责配置在两个连续传输字之间的休眠间隔，有效值为2~17个串行时钟周期。休眠间隔是指从前一次传输字的最后一个时钟沿到下一次传输字的第一个时钟沿。

如果CLKP = 1，间隔为前一次传输字的上升沿到下一次传输字的下降沿。

M05xxBN中，SP\_CYCLE的默认值为0x0 (2 个串行时钟周期)，如果Tx\_NUM = 0x00，设置这些位对数据传输过程没有任何影响。M05xxDN/DE中，SP\_CYCLE的默认值为0x3 (3.5 个串行时钟周期)，如果软件关闭FIFO模式，SP\_CYCLE的设定值不会影响字休眠间隔。

### 从机选择

在主机模式下，SPI控制器能通过从机片选输出脚SPISS0或者SPISS1驱动一个片外从机设备。从机模式下，片外的主机设备驱动从机片选信号通过SPISS0或者SPISS1输入到SPI控制器。在主机/从机模式下，从机选择信号的有效电平可以在SS\_LVL位 (SPI\_SSR[2])被配置为低有效或高有效，SS\_LTRIG 位 (SPI\_SSR[4])配置从机选择信号SPISS 为电平触发或边沿触发。触发条件的选择取决于所连接的外围从机/主机的设备类型。

从模式下，如果SS\_LTRIG 位被配置为电平触发，LTRIG\_FLAG位(SPI\_SSR[5])可以用来指示一次传输完成时，是否接收次数和接收到的bit数满足TX\_NUM 和 TX\_BIT\_LEN的定义（一次传输完成意味着从设备片选已经处于非有效状态或者SPI 控制器已经完成一次传输）

### 电平触发/边沿触发

在从机模式下，从机片选信号可以配置成电平触发或边沿触发。边沿触发时，数据传输从有效边沿开始，到出现一个无效边沿结束。如果检测到无效边沿，传输完成中断标志(SPI\_CNTRL[16])将被设为1。如果主机不发送无效边沿信号给从机，传输将不能完成，从机的中断标志将不会被置位。电平触发时，下面两个情况可以导致从机的中断标志被置位。一，如果主机设置从机片选管脚为非有效电平，将迫使从机终止当前传输而不管已经传输多少位，并且中断标志将被置位。用户可以读取LTRIG\_FLAG位的状态来判断数据是否传输完毕。二，如果传输位数与TX\_NUM 和 TX\_BIT\_LEN 的设置匹配时(M05xxDN/DE 中没有TX\_NUM的设定，只考虑TX\_BIT\_LEN的设定就好)，从机的中断标志将被置位。

### 6.12.5.2 自动从机选择

在主机模式下，如果AUTOSS (SPI\_SSR[3])置位，从机片选信号自动产生，并根据SSR[0] (SPI\_SSR[0])是否使能，输出到SPISS0/1引脚上，这意味着，从机选择信号（由SSR[0]寄存器选择）由SPI控制器在发送/接收开始（通过置位GO\_BUSY位(SPI\_CNTRL[0])）时置为有效，在传输结束时置为无效。当AUTOSS位清零时，可以手动置位与清零寄存器SPI\_SSR的SSR位，来发出或取消从机选择输出信号。从机选择输出信号的有效电平在SS\_LVL位 (SPI\_SSR[2])指定。

### 6.12.5.3 可变串行时钟频率(M05xxBN)

在主机模式下，如果可变时钟使能位VARCLK\_EN (SPI\_CNTRL [23])使能，SPI总线时钟的输出可被编程为可变频率模式。每个周期的时钟频率由寄存器SPI\_VARCLK (SPI\_VARCLK [31:0])的值决定。当使能VARCLK\_EN位，TX\_BIT\_LEN必须设置成0x10，配置数据传输为16 bits模式。SPI\_VARCLK[31]决定第一个时钟周期，如果SPI\_VARCLK[31]=0，第一个时钟周期由DIVIDER的值决定；如果SPI\_VARCLK[31]=1，第一个时钟周期由DIVIDER2的值决定。VARCLK[30:1]每2个连续的位决定1个时钟周期。如果两个连续比特为00，时钟周期由DIVIDER的值决定；如果两个连续比特为11，时钟周期由DIVIDER2的值决定。VARCLK[30:29]决定SPI总线时钟的第二个时钟周期；VARCLK[28:27]决定SPI总线时钟的第三个时钟周期；以此类推。VARCLK[0]没有意义。下图显示了SPI总线时钟、VARCLK设定、DIVIDER设定和DIVIDER2设定之间的关系。

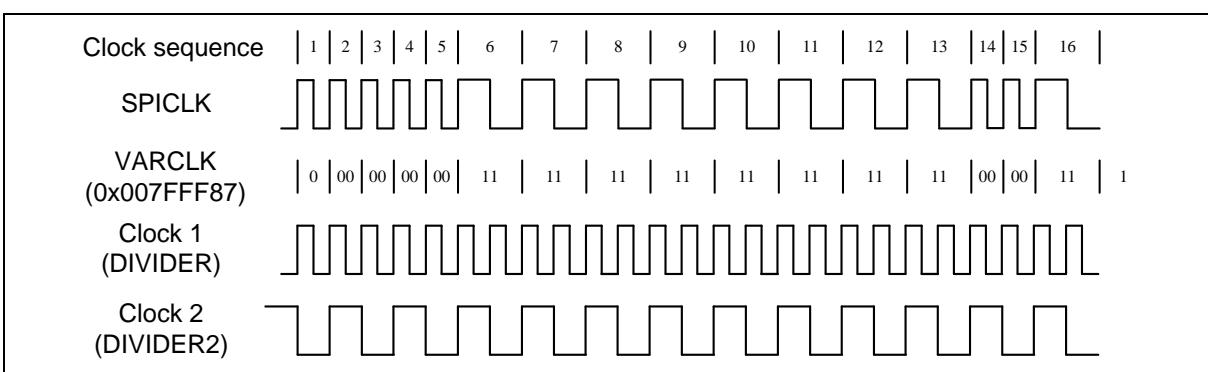


图 6-84 可变总线时钟频率

### 6.12.5.4 Burst 模式(M05xxBN)

SPI控制器可通过设置TX\_NUM (SPI\_CNTRL [9:8])为0X01，切换到burst模式。burst 模式下，SPI 可以在一次传输中进行两笔发送/接收处理。每笔传输的比特长度由TX\_BIT\_LEN 定义。SPI burst 模式波形

图如下：

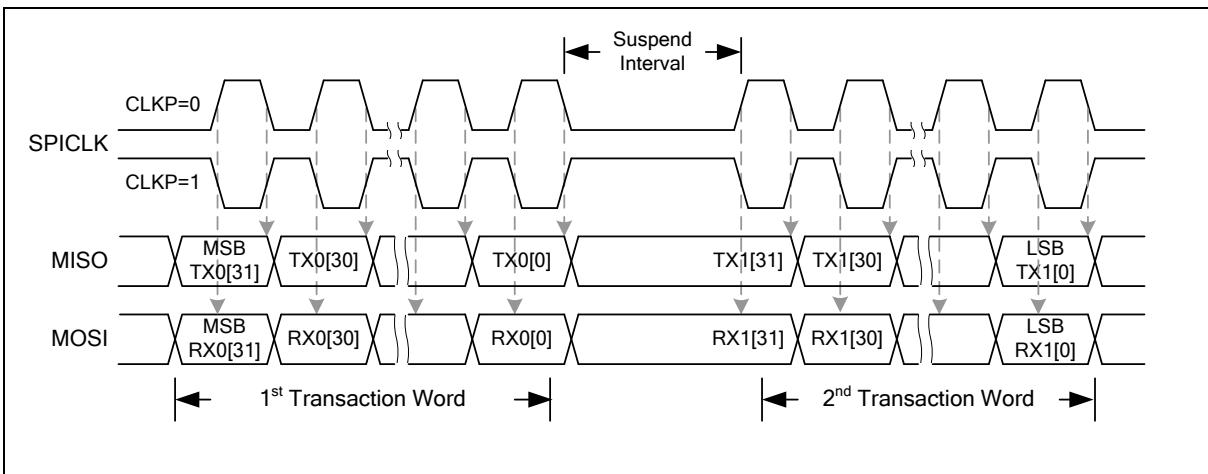


图6-85 一次传输两笔 (Burst Mode)

#### 6.12.5.5 字节重排序

当传输被设置为MSB优先(LSB = 0)，并且REORDER被使能时，TX\_BIT\_LEN = 32位模式下，存储在TX缓存与RX缓存中的数据将按[BYTE0, BYTE1, BYTE2, BYTE3]的次序重新排列，发送/接收数据将变成BYTE0, BYTE1, BYTE2, BYTE3的顺序。如果Tx\_BIT\_LEN被设置为24位模式，TX缓存与RX缓存的数据将被重新排列为[unknown byte, BYTE0, BYTE1, BYTE2]，BYTE0, BYTE1和BYTE2将按MSB优先的方式一个字节一个字节的被发送/接收。16位模式下规则与上述相同。字节重排序功能只有在TX\_BIT\_LEN等于16、24、32位时才有效。

其实字节重排序功能，就是数据大端(big-endian)和小端(little-endian)之间的转换

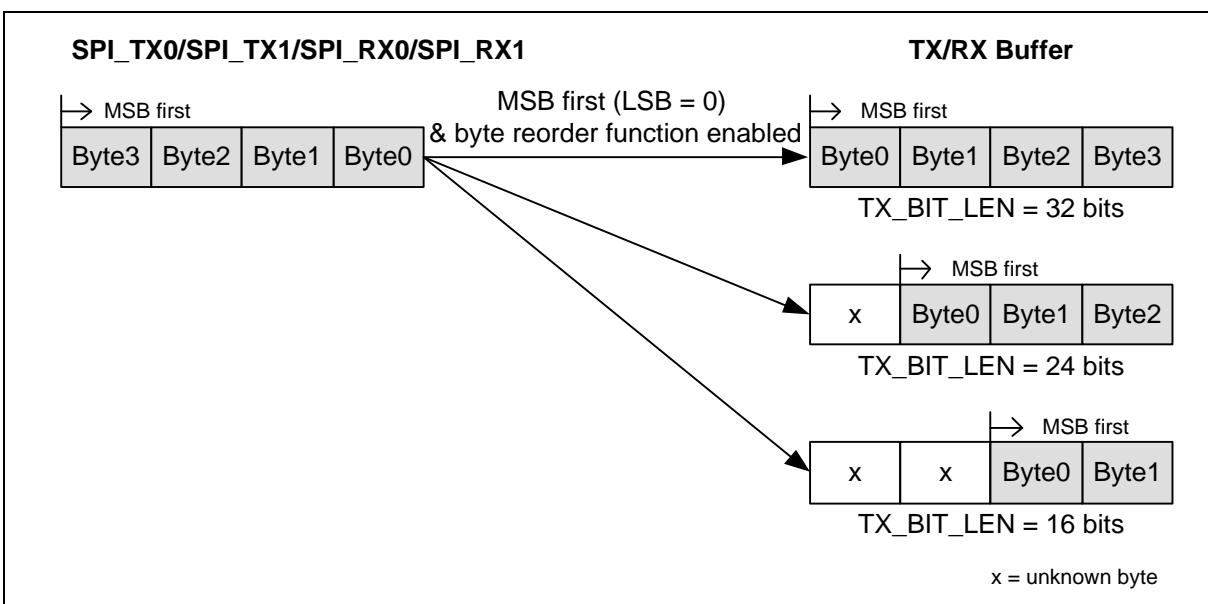


图6-86 字节重排列

### 6.12.5.6 字节休眠功能

主机模式下, 如果SPI\_CNTRL[19]被设置为1, 硬件将在一个传输字的两个连续传输字节之间插入2~17个串行时钟周期的休眠间隔。字节休眠的设定与字休眠设定一样, 二者使用共同的位域SP\_CYCLE, 注意当使能字节休眠功能时, TX\_BIT\_LEN必须被设置为0x00 (一个传输字32位)。

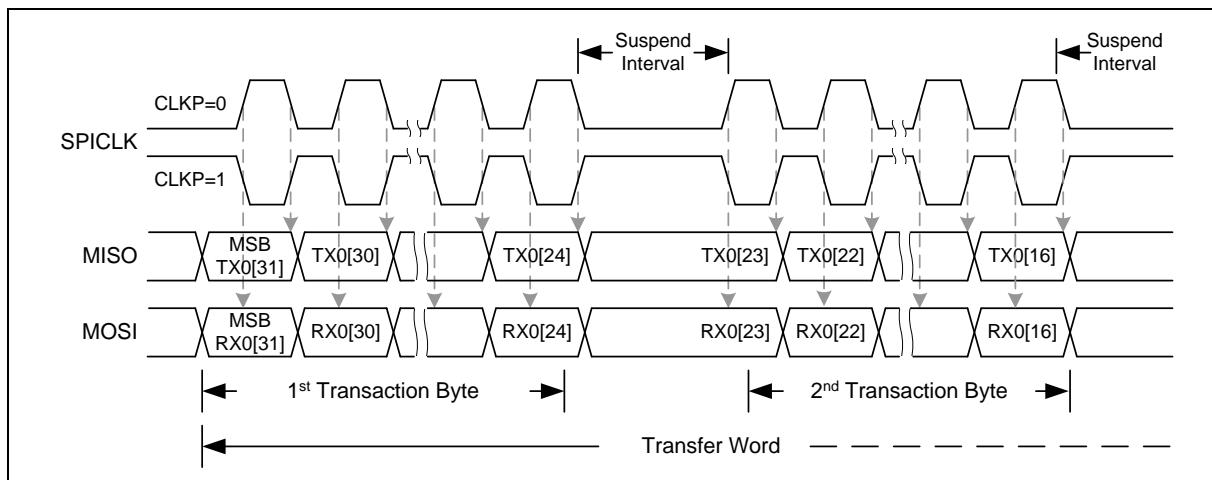


图6-87 字节休眠时序波形

REORDER	描述
00	禁用字节重排序功能和字节休眠.
01	使能字节重排序功能, 并在每个字节之间插入一个字节休眠间隔 (2~17串行时钟周期). TX_BIT_LEN 的设置必须配置成0x00 ( 32 bits/ word)
10	使能字节重排序功能但禁用字节休眠功能.
11	禁用字节重排序功能, 但在每个字节之间插入一个休眠间隔 (2~17串行时钟周期). TX_BIT_LEN的设置必须配置成 0x00 ( 32 bits/ word)

表6-14 M05xxBN中字节顺序和字节休眠条件

### 6.12.5.7 无从设备片选模式 (3-线模式)

从模式下, 可以忽略从设备片选信号。如果软件将NOSLVSEL 位 (SPI\_CNTRL2[8]) 设为1, 从设备3线模式将被使能, SPI控制器工作在无从设备片选信号模式 (3-线模式)。当MCU工作在从设备模式下时, 接口可以只包含 SPICLK, SPI\_MISO, 和 SPI\_MOSI 3根线。SPISS可以当作普通GPIO使用。当NOSLVSEL 位 被设为 1 时, GO\_BUSY 位被设为1后, SPI从设备就准备好接收/发送。一旦串行时钟出现, 控制器就开始发送/接收数据。当接收/发送比特数满足TX\_BIT\_LEN 和 TX\_NUM的定义时, 传输中断标志位IF (SPI\_CNTRL[16])将被置为1。

注: 在无从设备片选信号模式下, SS\_LTRIG位(SPI\_SSR[4]) 应该被设为1

### 6.12.5.8 FIFO 模式 (M05xxDN/DE)

当FIFO位(SPI\_CNTRL[21]) = 1时, SPI 控制器支持FIFO模式。SPI 控制器配有4个32-bit 宽收/发FIFO

缓冲。

发送 FIFO 缓冲有4层， 32-bit 宽，先进先出缓冲。通过写SPI\_TX0寄存器，软件可以将数据写到发送 FIFO缓冲中。保存在发送FIFO缓冲中的数据将被传输控制逻辑读出并发走。如果4-层发送FIFO 缓冲已满，TX\_FULL 位将等于1。当SPI发送逻辑将发送FIFO中的最后1个数据读出时，TX\_EMPTY 位将被设为1。注：当TX\_EMPTY = 1时，最后一个数据还在处理中。

接收 FIFO 缓冲也是4层， 32-bit 宽，先进先出缓冲。接收控制逻辑将收到的数据保存到缓冲中。通过读SPI\_RX0寄存器，软件可以将数据读出。FIFO 相关状态位： RX\_EMPTY 和 RX\_FULL指示FIFO 缓冲当前状态。

FIFO 模式下，通过设定TX\_THRESHOLD 和 RX\_THRESHOLD，软件可以设定发送和接收的阙值。当存在发送FIFO中的有效数据个数小于/等于TX\_THRESHOLD 的值时，TX\_INTSTS 位将被设为1。当存在接收FIFO中的有效数据个数大于RX\_THRESHOLD 的设定时，RX\_INTSTS 将被设为 1。

FIFO模式下，软件可以提前写4笔数据到SPI 发送FIFO 中。当SPI 控制器工作在FIFO 模式时，SPI\_CNTRL 寄存器的GO\_BUSY位由硬件控制，除了修改FIFO位关闭FIFO模式，软件不应该修改 SPI\_CNTRL 寄存器的内容。

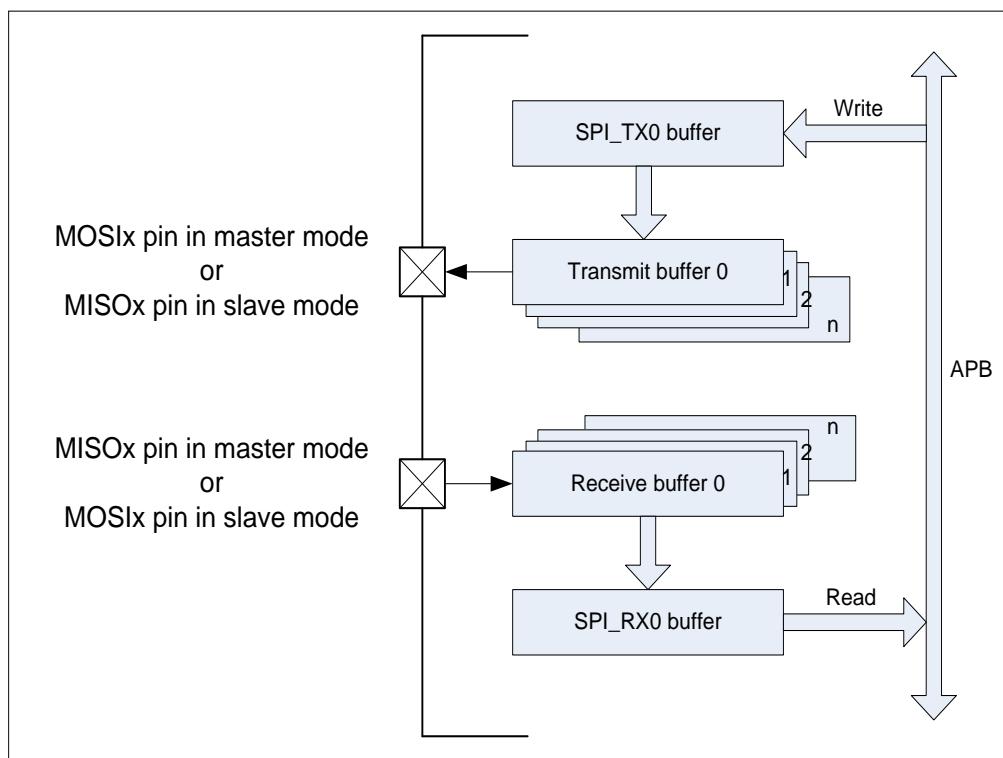


图 6-88 FIFO 模式框图

主机模式发送操作，当FIFO位等于1并且软件写第一笔数据到SPI\_TX0寄存器时，TX\_EMPTY 标志将被清成0。只要发送FIFO不为空，立即开始发送。用户可以接着写下一笔数据到SPI\_TX0 寄存器。

FIFO模式下，SPI控制器将在2笔连续传输之间插入休眠间隔，间隔时间由SP\_CYCLE (SPI\_CNTRL[15:12])决定。只要TX\_FULL等于0，用户可以一直写数据到SPI\_TX0寄存器中。

如果要发送的数据及时更新到SPI\_TX0寄存器中，发送将自动一直进行。如果发送FIFO中的数据全部发送完毕并且没有新的数据填入，发送将停止。

主模式接收操作，串行数据从MISOx引脚接收进来并保存在接收FIFO中。当接收FIFO中有数据没有被读走时，RX\_EMPTY标志将被清成0。只要RX\_EMPTY=0，软件可以通过SPI\_RX0寄存器读收到的数据。如果接收FIFO中有4笔数据没有读走，RX\_FULL标志将被设为1。SPI控制器将停止接收，直到软件读SPI\_RX0寄存器。

从模式下，当FIFO比特设为1，GO\_BUSY位由硬件控制。如果用户想停止从模式下SPI数据传输，软件应该将FIFO位和GO\_BUSY位都清成0。

从模式发送操作。当软件写数据到SPI\_TX0寄存器时，数据将被存到发送FIFO中，TX\_EMPTY标志将被清成0。当从设备收到来自主机的时钟信号时，发送将自动开始。只要TX\_FULL=0，软件可以一直写数据到SPI\_TX0寄存器中。所有数据都被SPI传输逻辑读走以后，软件没有再写SPI\_TX0寄存器，TX\_EMPTY标志将被设为1。

从模式接收操作，从MOSIx引脚接收串行数据，并保存到SPI\_RX0寄存器中。接收机制和主模式接收操作相似。

#### 6.12.5.9 中断

##### SPI 传输中断

数据传输完毕时，每一个SPI控制器会产生一个独立的中断，并且各自的中断事件标志IF (SPI\_CNTRL[16])将会被置位。如果中断使能位IE (SPI\_CNTRL[17])置位，则中断将发生。中断标志只能通过向其写1清除为零。

3-线模式下，当传输开始并且接收到的数据满足TX\_BIT\_LEN和TX\_NUM中的定义时，SLV\_START\_INTSTS中断标志将被设。如果接收到的bits数小于设定值，并且没有串行时钟输入超过用户设定的时间，用户可以设定SLV\_ABORT bit来迫使当前的传输完成，然后用户将得到一个传输完成中断。

##### SPI 从机 3 线模式开始中断

从机3线模式下，当从机检测到SPI总线时钟信号时，从机3线模式开始中断标志，SLV\_START\_INTSTS (SPI\_CNTRL2[11])，将被设为1。如果SSTA\_INTEN位 (SPI\_CNTRL2[10])等于1，SPI控制器将产生中断。如果接收到的比特数小于TX\_BIT\_LEN的设定，并且没有SPI总线时钟输入超过用户定义的时间，用户可以设置SLV\_ABORT位 (SPI\_CNTRL2[9]) 来终止当前传输。如果软件将SLV\_ABORT设为1，传输中断标志位IF，将被设为1。

##### 接收 FIFO 超时中断(M05xxDN/DE)

FIFO模式下，有超时功能通知用户。如果接收FIFO中有数据并且主模式下超过64个SPI外设时钟周期，从机模式下超过576个SPI外设时钟周期，没有再收到数据，如果超时中断使能位FIFO\_CTL[21]等于1，超时中断将发生。



### 发送FIFO 中断(M05xxDN/DE)

FIFO 模式下，如果发送FIFO中的有效数据个数小于/等于TX\_THRESHOLD的设定值，发送 FIFO 中断标志将被设成1。如果发送FIFO中断使能位SPI\_FIFO\_CTL[3]为1，SPI 控制器将产生一个发送 FIFO 中断。

### Receive FIFO interrupt (M05xxDN/DE Only)

FIFO 模式下，如果接收FIFO中的有效数据个数大于RX\_THRESHOLD的设定值，接收FIFO 中断标志将被设成1。如果接收FIFO中断使能位SPI\_FIFO\_CTL[2] 为1，SPI 控制器将产生一个接收FIFO 中断。

### 6.12.6 SPI 时序波形图

在主机/从机模式下，从机选择信号(SPISS)的有效电平可以在SS\_LVL 位 (SPI\_SSR[2])被编程为低电平有效或高电平有效，是电平触发还是边沿触发在SS\_LTRIG 位 (SPI\_SSR[4])中定义。总线时钟(SPICLK)的空闲状态可以通过CLKP位(SPI\_CNTRL[11])配置为高电平或低电平。在Tx\_BIT\_LEN (SPI\_CNTRL[7:3])中配置传输字的长度，在Tx\_NUM (SPI\_CNTRL[9:8])中配置传输的次数，在LSB bit (SPI\_CNTRL[10])中配置发送/接收数据是MSB还是LSB优先。用户还可以在寄存器Tx\_NEG/Rx\_NEG (SPI\_CNTRL[2:1])中选择在时钟的上升沿还是下降沿发送/接收数据。主机/从机的四种SPI操作时序图和相关的设定如下所示。

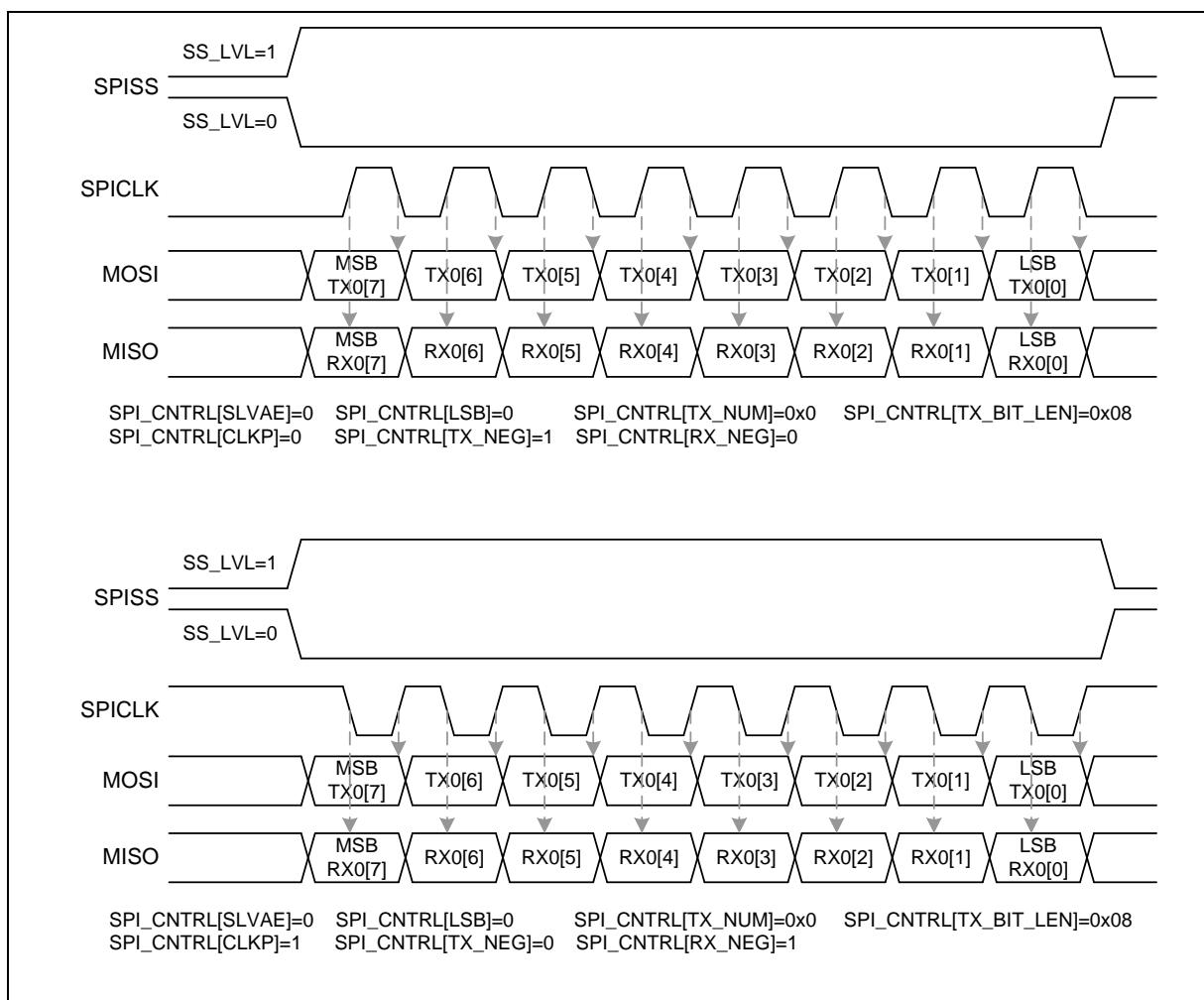


图 6-89 主机模式下 SPI 时序

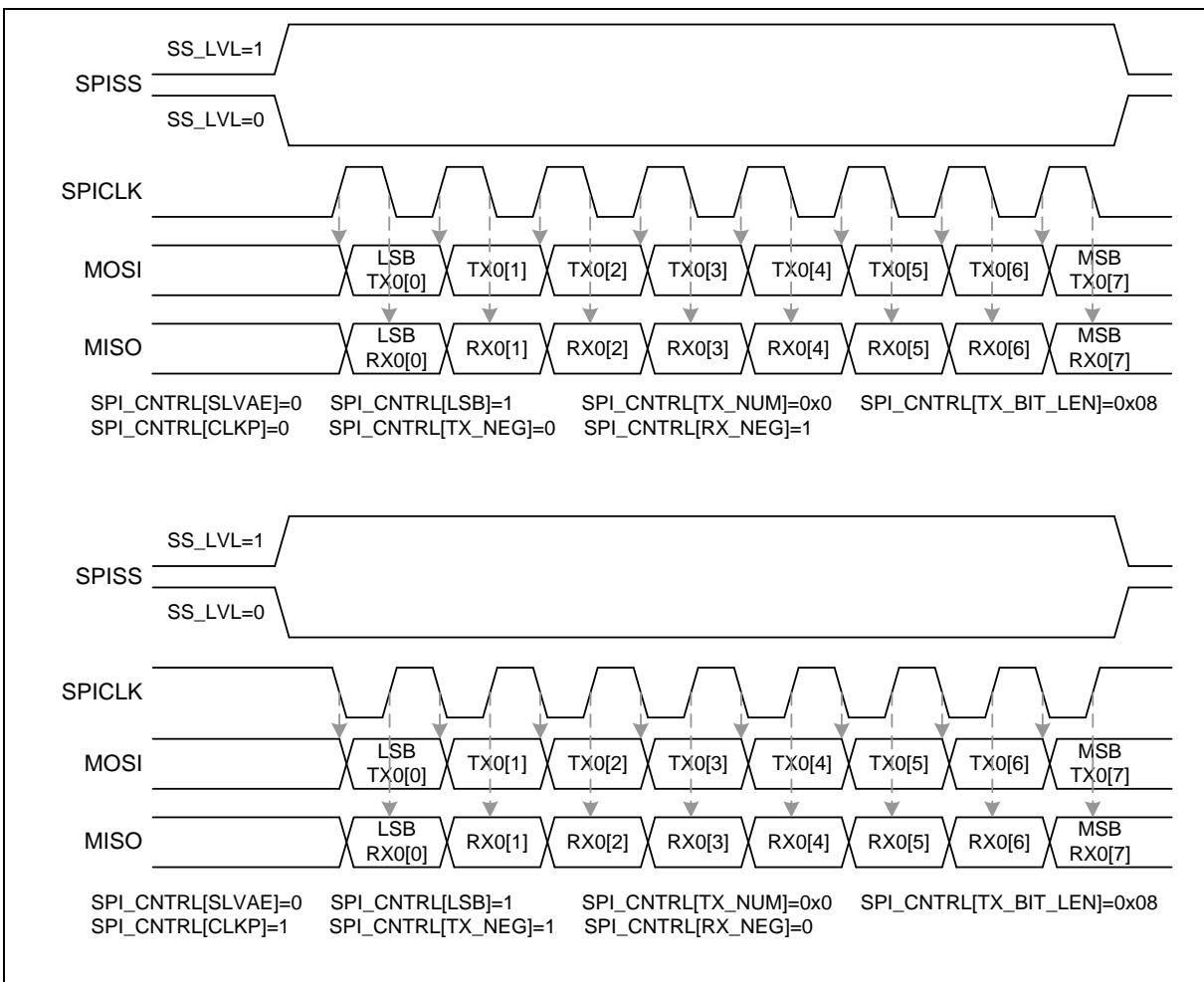


图 6-90 主机模式下 SPI 时序(Alternate Phase of SPICLK)

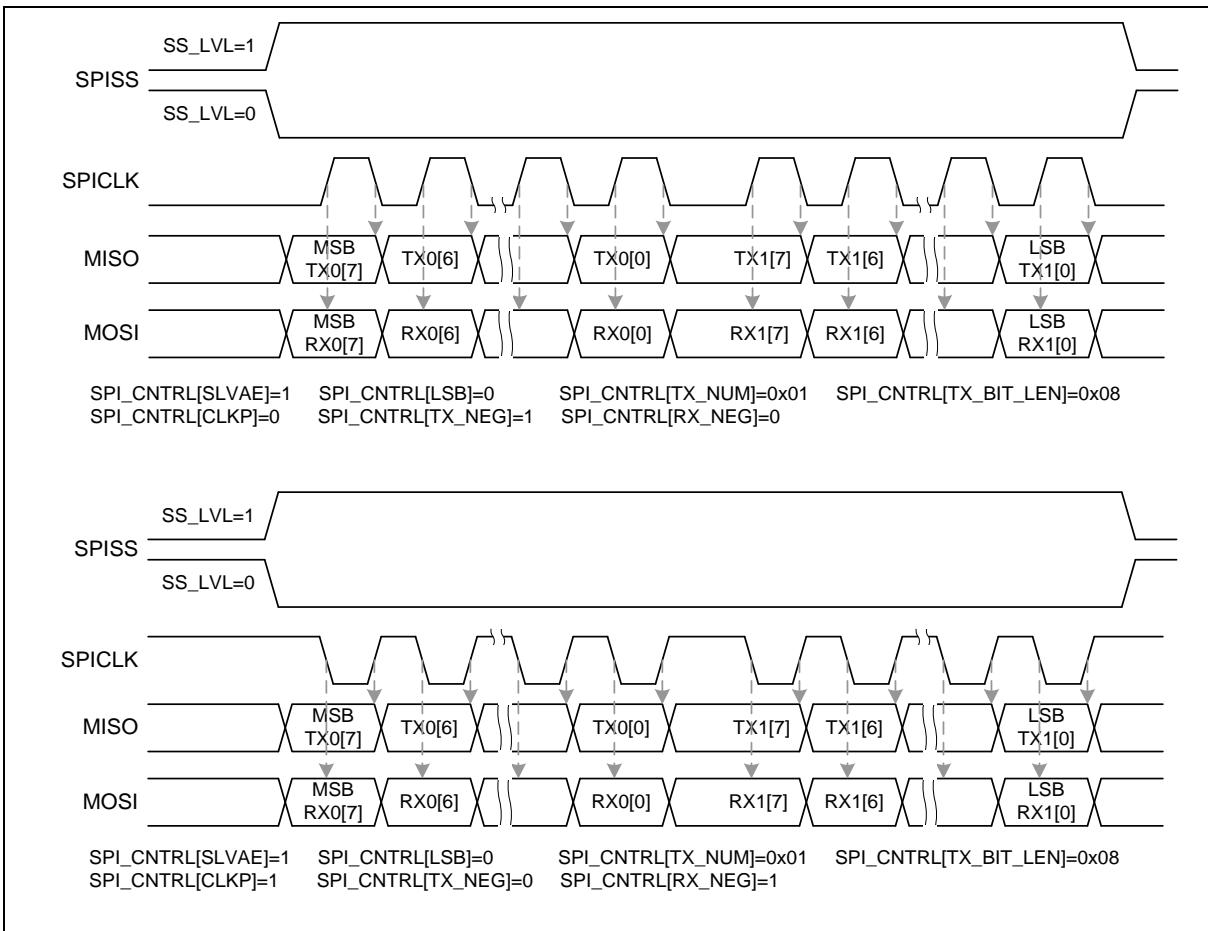


图 6-91 从机模式下 SPI 时序

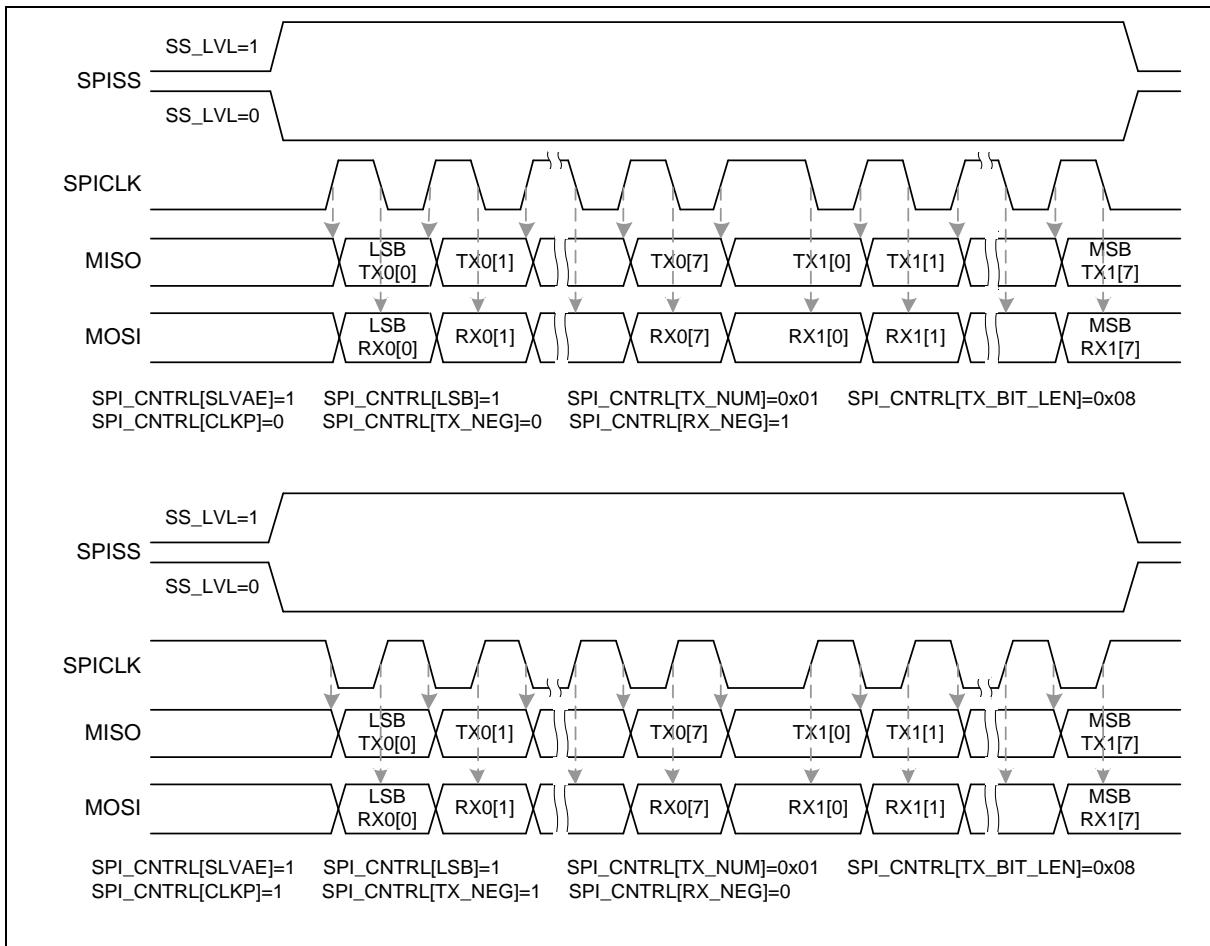


图 6-92 从机模式下 SPI 时序(Alternate Phase of SPICLK)

### 6.12.7 SPI编程范例

- 例 1, SPI 控制器作为主机, 按如下说明访问一个片外从机设备:

- 数据在时钟上升沿锁存
- 数据在时钟下将沿传输
- MSB优先
- 每次传输8bit
- 空闲时SPICLK为低电平
- 每次发送/接收一个字节
- 从机选择信号低有效

操作流程如下:

- 1) 设置DIVIDER (SPI\_DIVIDER[7:0])寄存器, 配置总线时钟输出频率.
- 2) 向寄存器SPI\_CNTRL 写入相应设置, 控制SPI主机动作

1. 通过**SLAVE** 位 (SPI\_CNTRL[18] = 0) 设置 SPI 控制器为主机设备
2. 通过**CLKP** 位 (SPI\_CNTRL[11] = 0) 设置串行时钟空闲状态为低
3. 通过**TX\_NEG** 位 (SPI\_CNTRL[2] = 1) 选择数据在串行时钟的下降沿发送
4. 通过**RX\_NEG** 位 (SPI\_CNTRL[1] = 0) 选择数据在串行时钟的上升沿锁存
5. 通过**TX\_BIT\_LEN** 位域 (SPI\_CNTRL[7:3] = 0x08) 设置传输字的长度为 8 位
6. 通过 **TX\_NUM** (SPI\_CNTRL[9:8] = 0x0) 设置为一次字传输
7. 通过 **LSB** 位 (SPI\_CNTRL[10] = 0) 设置为 MSB 优先传输，不必关心 SP\_CYCLE (SPI\_CNTRL[15:12]) 的设置，因为在本例中不是 FIFO 模式
- 3) 向 SPI\_SSR 写入适当的值，用于主机模式的相应设置
  1. 禁用自动片选 AUTOSS (SPI\_SSR[3] = 0)
  2. 配置从机片选有效电平位 **SS\_LVL** (SPI\_SSR[2] = 0)，低电平有效
  3. 通过设置从机片选寄存器位 **SSR[0]** (SPI\_SSR[0])，来使从机片选信号在 I/O 引脚上输出有效，以激活片外从机设备
- 4) 如果 SPI 主机要发送一个字节的数据到片外从机设备，把将要发送的字节数据写入 SPI\_TX0 寄存器。
- 5) 如果 SPI 主机要从外设接收一个字节的数据，不关心什么数据被发送，不需要写寄存器 SPI\_RX0.
- 6) 使能 **GO\_BUSY** 位 (SPI\_CNTRL[0] = 1)，以开始数据传输。
- 7) 等待 SPI 中断发生（如果 IE 使能），或轮询 **GO\_BUSY** 位直到其被硬件自动清零
- 8) 从寄存器 SPI\_RX0[7:0] 读出所接收到的一个字节的数据。
- 9) 跳转到步聚 4) 继续其他数据的传输或设置 **SSR[0]** 为 0 取消片选，以停止外设访问。

下面是基于 M051 系列 BSP 实现的示例代码

```
/* 设置时钟除频。SPI 时钟频率 = HCLK / ((11+1)*2) = 2MHz */
SPI0->DIVIDER = SPI0->DIVIDER & (~SPI_DIVIDER_DIVIDER_Msk) | SPI_DIVIDER_DIV(11);
/* 配置 SPI0 作为主机，MSB 优先，空闲时钟为低电平，下降沿发送，上升沿接收，8-bit 传输长度 */
SPI0->CNTRL = SPI_CNTRL_MASTER_MODE | SPI_CNTRL_MSB_FIRST | SPI_CNTRL_CLK_IDLE_LOW |
               SPI_CNTRL_TX_FALLING | SPI_CNTRL_RX_RISING | SPI_CNTRL_TX_BIT_LEN(8);
/* 关闭自动片选，选择 SS 引脚并配置为低电平有效 */
SPI0->SSR = SPI_SSR_SW_SS_PIN_LOW;
/* 写一个数据到 SPI0 的发送寄存器。*/
_SPI_WRITE_TX_BUFFER0(SPI0, SourceData[0]);
/* 启动发送 */
_SPI_SET_GO(SPI0);
/* 检查 GO_BUSY 位 */
While(SPI0->CNTRL & SPI_CNTRL_GO_BUSY);
/* 读收到的数据 */
DestData[0] = _SPI_GET_RX0_DATA(SPI0)&0xFF;
...
```

- 例 2, SPI控制器作为从机设备, 由片外主机设备控制, 片外主机设备依如下说明通过SPI接口访问片上SPI从机:

- 数据在时钟上升沿锁存
- 数据在时钟下降沿传输
- LSB优先
- 每次传输8位
- 空闲时SPICLK为高电平
- 每次发送/接收一个字节
- 从机片选信号高电平有效

操作流程如下:

- 1) 设置DIVIDER (SPI\_DIVIDER[7:0])寄存器, 配置从机外设时钟频率。从机外设时钟频率必须大于SPI总线时钟频率
- 2) 向SPI\_SSR寄存器写入适当的值, 用于从机模式的相应设置。写SS\_LVL (SPI\_SSR[2] = 1)配置从机片选有效电平为高电平, 写SS\_LTRIG (SPI\_SSR[4] = 1).配置从机片选为电平触发
- 3) 向SPI\_CNTRL 寄存器写入相应配置以控制SPI从机
  1. 通过SLAVE位(SPI\_CNTRL[18] = 1)设置SPI控制器为从机设备
  2. 通过CLKP位(SPI\_CNTRL[11] = 1)选择串行时钟空闲状态为高电平
  3. 通过 TX\_NEG位(SPI\_CNTRL[2] = 1)选择数据在串行时钟下降沿发送
  4. 通过RX\_NEG位(SPI\_CNTRL[1] = 0)选择数据在串行时钟的上升沿锁存
  5. 通过TX\_BIT\_LEN位域 (SPI\_CNTRL[7:3] = 0x08)设置字传输长度为8位
  6. 通过TX\_NUM (SPI\_CNTRL[9:8] = 0x0)设置为仅一次字传输
  7. 通过 LSB 位 (SPI\_CNTRL[10] = 1) 设置为 LSB 优先 传输, 不必关心 SP\_CYCLE (SPI\_CNTRL[15:12])的设置, 因为在本例中没有burst模式
- 4) 如果SPI从机要发送(被读取)一个字节数据到片外主机设备, 把将要发送的数据写入寄存器 SPI\_TX0[7:0]。
- 5) 如果SPI从机仅从外设主机接收一字节数据(被写), 用户不必关心什么数据将被传输, 不必更新 SPI\_RX0寄存器.
- 6) 使能GO\_BUSY bit (SPI\_CNTRL[0] = 1) , 等待片外主机输入从机片选信号和总线时钟, 开始将数据传输到SPI接口。
- 7) --等待SPI中断发生 (IE使能) , 或轮询GO\_BUSY 位直到其被硬件自动清零
- 8) 从SPI\_RX0[7:0] 寄存器中读出所接收到的一个字节的数据
- 9) 跳转到步聚3) 继续其他数据传输或清除GO\_BUSY 位停止数据传输.



下面是基于M051系列BSP实现的示例代码.

```
/* 配置 SPI0 片选高电平有效 */
SPI0->SSR = SPI_SSR_SLAVE_HIGH_LEVEL_ACTIVE;
/* 配置SPI0 作为从设备, LSB 优先, 时钟空闲高电平, 下降沿发送, 上升沿接收, 每次传输8-位*/
SPI0->CNTRL = SPI_CNTRL_SLAVE_MODE | SPI_CNTRL_LSB_FIRST | SPI_CNTRL_CLK_IDLE_HIGH |
    SPI_CNTRL_TX_FALLING | SPI_CNTRL_RX_RISING | SPI_CNTRL_TX_BIT_LEN(8);
/* 写数据到SPI0发送寄存器. */
_SPI_WRITE_TX_BUFFER0(SPI0, SourceData[0]);
/* 设置SPI0的 GO_BUSY */
_SPI_SET_GO(SPI0);
/* 轮询GO_BUSY 位 */
While(SPI0->CNTRL & SPI_CNTRL_GO_BUSY);
/* 读收到的数据 */
DestData[0] = _SPI_GET_RX0_DATA(SPI0)&0xFF;
...
```

### 6.12.8 寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>SPI基址:</b>				
<b>SPI0_BA = 0x4003_0000</b>				
<b>SPI1_BA = 0x4003_4000</b>				
<b>SPI_CNSR</b>	SPIx_BA + 0x00	R/W	控制及状态寄存器	0x0500_0004
<b>SPI_DIVIDER</b>	SPIx_BA + 0x04	R/W	时钟分频寄存器	0x0000_0000
<b>SPI_SSR</b>	SPIx_BA + 0x08	R/W	从机片选寄存器	0x0000_0000
<b>SPI_RX0</b>	SPIx_BA + 0x10	R	数据接收寄存器0	0x0000_0000
<b>SPI_RX1</b>	SPIx_BA + 0x14	R	数据接收寄存器1 (M05xxBN )	0x0000_0000
<b>SPI_TX0</b>	SPIx_BA + 0x20	W	数据发送寄存器0	0x0000_0000
<b>SPI_TX1</b>	SPIx_BA + 0x24	W	数据发送寄存器1(M05xxBN )	0x0000_0000
<b>SPI_VARCLK</b>	SPIx_BA + 0x34	R/W	可变时钟模型控制寄存器(M05xxBN )	0x007F_FF87
<b>SPI_CNSR2</b>	SPIx_BA+0x3C	R/W	控制和状态寄存器 2	0x0000_0000
<b>SPI_FIFO_CTL</b>	SPIx_BA+0x40	R/W	SPI FIFO 控制寄存器 (M05xxDN/DE )	0x2200_0000
<b>SPI_STATUS</b>	SPIx_BA+0x44	R/W	SPI 状态寄存器 (M05xxDN/DE)	0x0500_0000

注 1: 当软件编程CNTRL 寄存器时, GO\_BUSY 位必须最后写入.

注 2: x=0,1

### 6.12.9 寄存器描述

#### SPI 控制与状态寄存器(SPI\_CNTRL)

寄存器	偏移量	R/W	描述	复位后的值
SPI_CNTRL	SPIx_BA + 0x00	R/W	控制与状态寄存器	0x0500_0004

31	30	29	28	27	26	25	24
保留				TX_FULL	TX_EMPTY	RX_FULL	RX_EMPTY
23	22	21	20	19	18	17	16
VARCLK_EN	保留	FIFO	REORDER		SLAVE	IE	IF
15	14	13	12	11	10	9	8
SP_CYCLE				CLKP	LSB	TX_NUM	
7	6	5	4	3	2	1	0
TX_BIT_LEN					TX_NEG	RX_NEG	GO_BUSY

Bits	描述	
[31:28]	保留	保留
[27]	TX_FULL	发送 FIFO 缓冲满标志(只读) (M05xxDN/DE) 该位是SPI_STATUS[27]的镜像. 0 = 指示发送FIFO 没有满. 1 =指示发送FIFO 已满.
[26]	TX_EMPTY	发送 FIFO 缓冲空标志(只读) (M05xxDN/DE) 该位是SPI_STATUS[26]的镜像. 0 =指示发送FIFO 非空. 1 =指示发送FIFO为空.
[25]	RX_FULL	接收 FIFO 缓冲满标志(只读) (M05xxDN/DE) 该位是SPI_STATUS[25]的镜像. 0 = 指示接收 FIFO没有满. 1 =指示接收 FIFO已满.
[24]	RX_EMPTY	接收 FIFO 缓冲空标志(只读) (M05xxDN/DE) 该位是SPI_STATUS[24]的镜像. 0 = 指示接收 FIFO非空. 1 = 指示接收 FIFO为空.
[23]	VARCLK_EN	可变时钟使能 (仅M05xxBN主机模式) 0 = 总线时钟输出仅由DIVIDER的值决定. 1 = 总线时钟输出可变. 输出频率由VARCLK, DIVIDER, 和 DIVIDER2的值决定. 注: 当使能VARCLK_EN, TX_BIT_LEN 必须设置成 0x10 (16 bits 模式)

[22]	保留	保留
[21]	FIFO	<p><b>FIFO 模式使能控制 (M05xxDN/DE )</b></p> <p>主模式下，如果使能FIFO 模式，写数据到发送FIFO后，GO_BUSY 位由硬件自动设为1。当发送FIFO中的所有数据都被发送完毕，GO_BUSY 将被清为0。</p> <p>0 = 关闭FIFO 模式. 1 = 使能FIFO 模式.</p> <p><b>注:</b> 使能 FIFO 模式之前，其它相关设定要提前设好</p>
[20:19]	REORDER	<p><b>字节重排序功能和字节休眠功能选择</b></p> <p><b>M05xxBN</b></p> <p>00 = 禁用字节重排序和字节休眠功能. 01 = 使能字节重排序，并在每两个字节之间插入一个字节休眠间隔 (2~17 串行时钟周期). TX_BIT_LEN 必须设置成0x00. (32 bits/word) 10 = 使能字节重排序功能，但禁用字节休眠功能. 11 = 禁用字节重排序功能，但在每两个字节之间插入一个休眠间隔(2~17 串行时钟周期). TX_BIT_LEN 必须设置成0x00. (32 bits/word)</p> <p><b>注 1:</b> 字节重排序功能只是在 TX_BIT_LEN 被配置为 16, 24, 和 32 bits时才有效. <b>注 2:</b> 从模式下，电平触发时，如果使能字节休眠功能，连续4个字节传输期间，从机片选信号必须保持有效。</p> <p><b>M05xxDN/DE:</b></p> <p>00 = 禁用字节重排序和字节休眠功能. 01 = 使能字节重排序，字节休眠间隔时间由SP_CYCLE决定。设置 SP_CYCLE 为 0 关闭字节休眠功能 10 = 保留. 11 = 保留.</p> <p><b>注:</b> 字节重排序功能只是在 TX_BIT_LEN 被配置为 16, 24, 和 32 bits时才有效.</p>
[18]	SLAVE	<p><b>从机模式选择</b></p> <p>0 = 主机模式 1 = 从机模式</p>
[17]	IE	<p><b>传输中断使能控制</b></p> <p>0 = 禁用SPI 传输中断. 1 = 使能SPI传输中断.</p>
[16]	IF	<p><b>传输中断标志</b></p> <p>0 = 表示传输未结束 1 = SPI控制器已经完成1次传输</p> <p><b>注:</b> 该位写1清除为零.</p>
[15:12]	SP_CYCLE	<p><b>休眠间隔 (仅主机模式)</b></p> <p>该四位用于设定两次连续传输间的间隔时间。间隔时间定义为从当前传输的最后一个时钟沿到下次传输的第一个时钟沿。</p> <p>M05xxBN:</p> <p>默认值为0x0。当TX_NUM = 00, 该位无效。下列公式可获得所需的间隔时间。</p> <ul style="list-style-type: none"> <li>■ 用于配置字节休眠间隔和burst模式的休眠间隔:</li> </ul>

		<p><math>(SP\_CYCLE[3:0] + 2) * SPICLK</math>时钟周期+1个系统时钟周期</p> <p>例如:</p> <p><math>SP\_CYCLE = 0x0 \dots 2</math>个SPICLK时钟周期+1个系统时钟周期  <math>SP\_CYCLE = 0x1 \dots 3</math>个SPICLK时钟周期+1个系统时钟周期  <math>\dots</math>  <math>SP\_CYCLE = 0xE \dots 16</math>个SPICLK时钟周期+1个系统时钟周期  <math>SP\_CYCLE = 0xF \dots 17</math>个SPICLK时钟周期+1个系统时钟周期</p> <p>如果SPI总线时钟频率等于系统时钟频率，就是说DIV_ONE特性被使能，burst模式休眠间隔周期为</p> <p style="text-align: center;"><math>(SP\_CYCLE[3:0] * 2 + 3.5) * \text{系统时钟周期}</math></p> <p>M05xxDN/DE:</p> <p>默认值是 0x3。休眠间隔周期根据下列公式获得:</p> <p style="text-align: center;"><math>(SP\_CYCLE[3:0] + 0.5) * SPICLK</math> 时钟周期</p> <p>例如:</p> <p><math>SP\_CYCLE = 0x0 \dots 0.5</math> SPICLK 时钟周期  <math>SP\_CYCLE = 0x1 \dots 1.5</math> SPICLK 时钟周期  <math>\dots</math>  <math>SP\_CYCLE = 0xE \dots 14.5</math> SPICLK 时钟周期  <math>SP\_CYCLE = 0xF \dots 15.5</math> SPICLK clock cycle</p>
[11]	CLKP	<p><b>时钟极性</b></p> <p>0 = 空闲时SPICLK 为低电平      1 = 空闲时SPICLK 为高电平.</p>
[10]	LSB	<p><b>优先传送LSB</b></p> <p>0 = 优先发送/接收MSB      1 = 优先发送/接收 LSB.</p>
[9:8]	TX_NUM	<p><b>发送/接收次数(M05xxBN)</b></p> <p>该寄存器用于说明一次传输中，传输的笔数。</p> <p>00 = 每次传输(GO_BUSY=1)仅完成一次发送/接收      01 = 每次传输(GO_BUSY=1)完成两次连续的发送/接收      10 = 保留.      11 = 保留.</p> <p><b>注:</b> 从机模式下，电平触发时，如果 TX_NUM 等于 01，在连续的数据传输之间，从设备片选引脚必须保持有效</p>
[7:3]	TX_BIT_LEN	<p><b>传输位长度</b></p> <p>该寄存器用于说明每笔传输的bit长度，最高为32位。</p> <p><math>Tx\_BIT\_LEN = 0x01 \dots 1</math>位  <math>Tx\_BIT\_LEN = 0x02 \dots 2</math>位  <math>\dots</math>  <math>Tx\_BIT\_LEN = 0x1f \dots 31</math>位  <math>Tx\_BIT\_LEN = 0x00 \dots 32</math>位</p>

[2]	<b>TX_NEG</b>	下降沿发送数据 0 = 发送的数据在SPICLK的上升沿改变。 1 = 发送的数据在SPICLK的下降沿改变.
[1]	<b>RX_NEG</b>	下降沿接收数据 0 = 接收到的数据在SPICLK上升沿锁存 1 = 接收到的数据在SPICLK下降沿锁存.
[0]	<b>GO_BUSY</b>	<p>开始传输和忙状态标志</p> <p><b>M05xxBN:</b> 数据传输期间，该位维持为1。当发送完成，该位由硬件自动清0 0 = 当SPI正在传输时，对该位写0会使数据传输停止 1 = 主机模式下，对该位写1启动SPI开始传输数据；从机模式下，对该位写1表明从机已准备好与主机通讯。</p> <p><b>注:</b> 在对CNTRL寄存器的GO_BUSY置1之前，必须先配置相应的寄存器。已经开始传输才对其他位进行配置，无法影响当前传输。</p> <p><b>M05xxDN/DE:</b> FIFO 模式下，该位由硬件控制，软件不要修改该位。 如果关闭FIFO 模式，在数据传输期间，该位维持为1。当发送完成，该位由硬件自动清0。 0 = 当SPI正在传输时，对该位写0会使数据传输停止 1 = 主机模式下，对该位写1启动SPI开始传输数据；从机模式下，对该位写1表明从机已准备好与主机通讯。</p> <p><b>注 1:</b> 当关闭FIFO 模式时，GO_BUSY置1之前，所有相应寄存器应该先的配置好。</p> <p><b>注 2:</b> M05xxDN/DE 中SPI 从模式下，数据传输期间，如果 FIFO 模式关闭并且SPI总线时钟维持空闲状态，如果从机片选信号变为无效，GO_BUSY位不会被清为0。</p>

**SPI 分频寄存器 (SPI\_DIVIDER)**

寄存器	偏移量	R/W	描述	复位后的值
SPI_DIVIDER	SPIx_BA + 0x04	R/W	时钟分频寄存器(仅主机模式)	0x0000_0000

31	30	29	28	27	26	25	24
DIVIDER2[15:8]							
23	22	21	20	19	18	17	16
DIVIDER2[7:0]							
15	14	13	12	11	10	9	8
DIVIDER[15:8]							
7	6	5	4	3	2	1	0
DIVIDER[7:0]							

Bits	描述
[31:16]	<b>DIVIDER2</b> <b>时钟分频器2(仅M05xxBN主机模式)</b> 这个域的值是第二个频率除数，用于产生总线时钟输出SPICLK。可以根据下列公式获得期望的频率： $f_{sclk} = \frac{f_{psclk}}{(DIVIDER2+1)*2}$ 如果VARCLK_EN被清除成0，这个域的设定没有意义
[15:0]	<b>DIVIDER</b> <b>时钟分频器(仅主机模式)</b> <b>M05xxBN:</b> 只用于主机模式。这个域的值是频率除数，用于决定主机总线时钟输出SPICLK的频率。根据下列方程获得期望的频率 $f_{sclk} = \frac{f_{psclk}}{(DIVIDER+1)*2}$ 从机模式下，由主机提供的SPI总线时钟周期，应该大于或等于PCLK周期的5倍。换言之，输入的SPI时钟的最大频率为从机PCLK的1/5。 <b>M05xxDN/DE:</b> 只有DIVIDER[7:0] 可得。该域的值是频率除数，用于决定SPI 外设时钟频率， $f_{spi}$ ，和SPI 主机总线时钟频率 SPICLK。频率由下列公式决定。 如果BCn, SPI_CNTRL2[31], 等于0 0, $f_{spi} = \frac{f_{SPI\_clock\_src}}{(DIVIDER+1)*2}$ 如果BCn 等于 1,

		$f_{spi} = \frac{f_{SPI\_clock\_src}}{(DIVIDER + 1)}$ <p><math>f_{SPI\_clock\_src}</math> 是SPI外设时钟源的频率，在CLKSEL1 寄存器中选择.</p>
--	--	---



### SPI从机片选寄存器(SPI\_SS\_R)

寄存器	偏移量	R/W	描述	复位后的值
SPI_SS_R	SPI0_BA + 0x08	R/W	从机片选寄存器	0x0000_0000

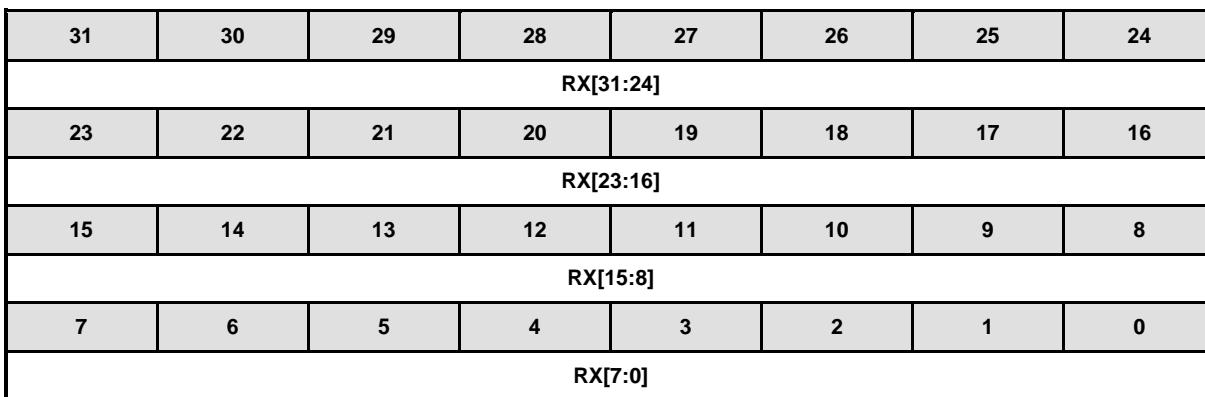
31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留		LTRIG_FLAG	SS_LTRIG	AUTOSS	SS_LVL	保留	SSR

Bits	描述	
[31:6]	保留	保留
[5]	LTRIG_FLAG	<p><b>电平触发标志（从机模式）</b>            在从机模式下如果SS_LTRIG置位，该标志能够表示接收到的位数量是否达到要求。            1 = 接收数量和接收位长度达到TX_NUM 及 TX_BIT_LEN 定义的值。            0 = 接收数量或接收位长度没有达到TX_NUM 及 TX_BIT_LEN 定义的值。.</p> <p><b>注：</b>该位只读，只用于从机模式</p>
[4]	SS_LTRIG	<p><b>从机片选电平触发使能（从机模式）</b>            0 = 从机片选信号为边沿触发。根据 SS_LVL选择是上升沿/下降沿触发            1 = 从机片选信号为电平触发。根据 SS_LVL选择是高电平/低电平触发。</p>
[3]	AUTOSS	<p><b>自动从机片选功能使能(主机模式)</b>            0 = 该位清0，从机选择信号是否有效，由设置或清除SSR位决定。.            1 = 该位置位，SPISSx信号自动产生。这意味着片选信号在置位GO_BUSY开始发送/接收时由SPI控制器自动置为有效，并且在传输结束后自动变为无效。</p>
[2]	SS_LVL	<p><b>从机片选激活电平</b>            该位决定从机片选信号的激活电平(SPISSx)            0 = SPISSx 从机片选为低电平/下降沿时有效。            1 = SPISSx从机片选为高电平/上升沿时有效。</p>
[1]	保留	保留
[0]	SSR	<p><b>从机片选控制(主机模式)</b>            当AUTOSS位被清除，            0 = 片选变为无效状态            1 = 设置SPISSx线激活片选信号            当AUTOSS位被置1，</p>

		<p>0 = SPISSx处于无效状态 1 = 将会使SPISSx线在传输/接受数据时自动驱动至激活状态。在其他时间驱动至非有效状态。SPISSx的有效状态由SS_LVL决定</p>
--	--	---

**SPI数据接收寄存器 (SPI\_RX)**

寄存器	偏移量	R/W	描述	复位后的值
<b>SPI_RX0</b>	SPIx_BA + 0x10	R	数据接收寄存器 0	0x0000_0000
<b>SPI_RX1</b>	SPIx_BA + 0x14	R	数据接收寄存器1 (M05xxBN )	0x0000_0000



Bits	描述	
[31:0]	<b>RX</b>	<p><b>数据接收寄存器</b></p> <p>数据接收寄存器内保存最后一次传输所接收的数据。数据的长度根据<b>SPI_CNSTRL</b>寄存器内配置的长度决定。</p> <p>例如，Tx_BIT_LEN 设定为 0x08 且 TX_NUM 设定为 0x0， RX0[7:0] 内保存收到的数据，其它bit为无效值。</p> <p><b>注:</b> 数据接收寄存器为只读寄存器.</p>

**SPI 数据发送寄存器(SPI\_TX)**

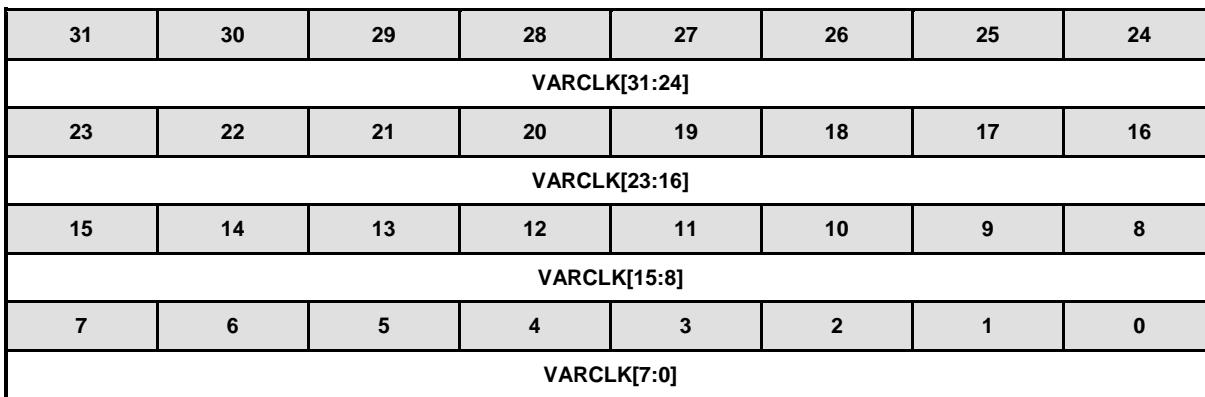
寄存器	偏移量	R/W	描述	复位后的值
<b>SPI_TX0</b>	SPIx_BA + 0x20	W	数据发送寄存器0	0x0000_0000
<b>SPI_TX1</b>	SPIx_BA + 0x24	W	数据发送寄存器1 (M05xxBN Only)	0x0000_0000

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

Bits	描述	
[31:0]	TX	<p><b>数据发送寄存器</b></p> <p>数据发送寄存器内存储下一次被发送的数据。数据的长度根据CNTRL寄存器内配置的长度决定。</p> <p>例如，TX_BIT_LEN设定为 0x08 且TX_NUM 设定为0x0，TX0[7:0] 内的数据将被发送。如果TX_BIT_LEN 设定为 0x00 且TX_NUM 设定为0x1，SPI控制器将用同种设置连续进行两笔32位数据发送/接收，发送顺序是先TX0[31:0],再TX1[31:0]。</p>

**SPI 可变时钟模型寄存器(SPI\_VARCLK)**

寄存器	偏移量	R/W	描述	复位后的值
SPI_VARCLK	SPIx_BA + 0x34	R/W	可变时钟模型寄存器(M05xxBN)	0x007F_FF87



Bits	描述	
[31:0]	VARCLK	<p><b>可变时钟模型</b></p> <p>该值决定SPI总线时钟频率模型。如果VARCLK为'0'， SPICLK的输出频率取决于DIVIDER的值。如果VARCLK为'1'， SPICLK的输出频率取决于DIVIDER2的值。参考寄存器 <a href="#">SPI_DIVIDER</a>.</p> <p><b>注：</b>详细请参考上面可变总线时钟频率一节</p>

**SPI 控制和状态寄存器2 (SPI\_CNTRL2)**

寄存器	偏移量	R/W	描述	复位后的值
SPI_CNTRL2	SPIx_BA+0x3C	R/W	控制与状态寄存器2	0x0000_0000

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

Bits	描述
[31]	<b>BCn</b> 时钟配置后向兼容选项 (M05xxDN/DE ) 0 = 时钟配置和 M05xxBN后向兼容。 1 = 时钟配置和M05xxBN不兼容。 细节参考SPI_DIVIDER 寄存器的描述.
[30:17]	保留
[16]	<b>SS_INT_OPT</b> 从机片选无效中断选项 (M05xxDN/DE ) 该设定只有在 SPI 控制器被配置为电平触发从机模式才起作用 0 = 当从机片选信号变成无效时, IF 位不会被设为1. 1 = 当从机片选信号变成无效时, IF 位将被设为1.
[15:12]	保留
[11]	<b>SLV_START_INTSTS</b> 从机3线模式开始传输中断状态 从模式下, 没有片选信号时 (3线模式) 用来指示传输已经开始。该位是 SPI_STATUS[11]的镜像。 1 = 从机3线模式下, 传输已经开始, 当传输结束或者写1可以清除该位。 0 = 自从SSTA_INTEN设为1, 从机没有检测到任何SPI总线时钟, 传输还没有开始.
[10]	<b>SSTA_INTEN</b> 从机3线模式开始中断使能 从机3线模式下, 如果设定该位为1, 当传输开始时中断将发生。如果传输开始以后, 在用户定义的时间内没有发生传输结束中断, 用户可以设定SLV_ABORT 位来迫使传输结束. 1 = 使能传输开始中断。当传输结束或者SLV_START_INTSTS bit 被清0, 这个 bit将被清成0. 0 = 禁止传输开始中断.

[9]	<b>SLV_ABORT</b>	<p><b>从机3线模式终止传输控制</b></p> <p>正常情况下，当接收到的数据数量满足TX_BIT_LEN 和 TX_NUM 的设定时，中断将发生。M05xxDN/DE下没有TX_NUM，只要TX_BIT_LEN满足就可以。</p> <p>如果接收到的比特数 小于请求并且没有总线时钟输入超过1次传输时间，用户可以设定该位为1来迫使当前传输结束，然后用户将得到一个传输结束中断。</p> <p>注：软件将该位设为1之后，硬件自动将这个bit清0</p>
[8]	<b>NOSLVSEL</b>	<p><b>从机3线模式使能</b></p> <p>从模式下，用来忽略从机片选信号。SPI控制器可以工作在3线模式，包括SPICLK, SPI_MISO, 和 SPI_MOSI.</p> <p>0 = 控制器为4-线双向接口。</p> <p>1 = 从模式下，控制器为3-线双向接口。当这个比特被设为“1”时，设定GO_BUSY 位为1并且有串行时钟输入，控制器就开始收/发数据</p> <p>注：从机3线模式下，SS_LTRIG位 (SPI_SSR[4]) 应该被设为 1.</p>
[7:1]	保留	保留
[0]	<b>DIV_ONE</b>	<p><b>SPI 总线时钟除频控制(M05xxBN 主机模式)</b></p> <p>0 = SPI 总线时钟频率由SPI_DIVIDER 寄存器决定.</p> <p>1 = 使能 DIV_ONE 特性. SPI 总线时钟频率等于系统时钟频率.</p> <p>注意:</p> <p>1. 当该位设为1时， REORDER 域和VARCLK_EN 域必须都被配置为0。换句话说，字节重新排序功能，字节休眠功能和可变时钟功能必须关闭.</p>

**SPI FIFO 控制寄存器 (SPI\_FIFO\_CTL)**

寄存器	偏移量	R/W	描述				复位后的值
SPI_FIFO_CTL	SPIx_BA+0x40	R/W	SPI FIFO 控制寄存器 (M05xxDN/DE)				0x2200_0000

31	30	29	28	27	26	25	24
保留		TX_THRESHOLD			保留		RX_THRESHOLD
23	22	21	20	19	18	17	16
保留		TIMEOUT_INTEN	保留				
15	14	13	12	11	10	9	8
保留							
7	6	5	4	3	2	1	0
保留	RXOV_INTEN	保留		TX_INTEN	RX_INTEN	TX_CLR	RX_CLR

Bits	描述	
[31:30]	保留	保留.
[29:28]	TX_THRESHOLD	发送 FIFO 阈值 如果发送 FIFO 中有效数据个数小于或者等于 TX_THRESHOLD 的值, TX_INTSTS 位将被设为 1, 否则 TX_INTSTS 将被清为0
[27:26]	保留	保留.
[25:24]	RX_THRESHOLD	接收 FIFO 阈值 如果接收 FIFO 中的有效数据个数大于 RX_THRESHOLD 的值, RX_INTSTS 位将被设为1, 否则 RX_INTSTS 将被清为0
[23:22]	保留	保留.
[21]	TIMEOUT_INTEN	接收 FIFO 超时中断使能控制 0 = 禁止超时中断. 1 = 使能超时中断.
[20:7]	保留	保留.
[6]	RXOV_INTEN	接收 FIFO 溢出中断使能控制 0 = 禁止接收 FIFO 溢出中断 1 = 使能接收 FIFO 溢出中断.
[5:4]	保留	保留.
[3]	TX_INTEN	发送阈值中断使能控制 0 = 禁止发送阈值中断. 1 = 使能发送阈值中断.
[2]	RX_INTEN	接收阈值中断使能控制

		0 = 禁止接收阈值中断。 1 = 使能接收阈值中断.
[1]	TX_CLR	<b>清除发送 FIFO 缓冲</b> 0 = 没有作用. 1 = 清除发送 FIFO 缓冲。TX_FULL 标志将被清为 0， TX_EMPTY 标志将被设为1。软件设该位为1以后，该位由硬件清为0，发送FIFO 被清除.
[0]	RX_CLR	<b>清除接收 FIFO缓冲</b> 0 = 没有作用. 1 = 清除接收 FIFO 缓冲。RX_FULL标志将被清为 0， RX_EMPTY标志将被设为1。软件设该位为1以后，该位由硬件清为0，接收FIFO 被清除.

**SPI 状态寄存器 (SPI\_STATUS)**

寄存器	偏移量	R/W	描述	复位后的值
SPI_STATUS	SPIx_BA+0x44	R/W	SPI 状态寄存器 (M05xxDN/DE )	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
保留			TIMEOUT	保留			IF
15	14	13	12	11	10	9	8
RX_FIFO_COUNT				SLV_START_INTSTS	保留		
7	6	5	4	3	2	1	0
保留			TX_INTSTS	保留	RX_OVERRUN	保留	RX_INTSTS

Bits	描述	
[31:28]	TX_FIFO_COUNT	发送 FIFO 中数据个数 (只读) 指示发送FIFO中有效的数据个数。
[27]	TX_FULL	发送FIFO Buffer Full Indicator (只读) 该位是 SPI_CNTRL[27]的镜像. 0 = 发送FIFO 缓冲没有满. 1 = 发送 FIFO buffer is full.
[26]	TX_EMPTY	发送FIFO Buffer Empty Indicator (只读) 该位是 SPI_CNTRL[26]的镜像. 0 = 发送 FIFO 缓冲不是空的. 1 = 发送 FIFO 缓冲是空的.
[25]	RX_FULL	接收 FIFO Buffer Full Indicator (只读) 该位是 SPI_CNTRL[25]的镜像. 0 = 接收 FIFO 没有满. 1 = 接收FIFO 已满.
[24]	RX_EMPTY	接收FIFO Buffer Empty Indicator (只读) 该位是 SPI_CNTRL[24]的镜像. 0 = 接收FIFO 不是空的. 1 = 接收FIFO 是空的.
[23:21]	保留	保留
[20]	TIMEOUT	超时中断标志 0 = 没有发生接收 FIFO 超时事件. 1 = 接收 FIFO 缓冲不是空的并且主模式下超过64个SPI时钟周期或者从模式下超过576 个SPI 外设时钟周期没有读。当接收缓冲被软件读时，超时状态将被自

		自动清除。 注: 该位写1清0.
[19:17]	保留	保留.
[16]	IF	<b>SPI 传输中断标志</b> 该位是 SPI_CNTRL[16]的镜像. 0 = 传输还没有完成. 1 = SPI 控制器完成1次传输. 注: 该位写1清0.
[15:12]	RX_FIFO_COUNT	<b>接收 FIFO 中数据个数 (只读)</b> 指示接收 FIFO 中有效数据个数.
[11]	SLV_START_INTSTS	<b>从设备开始中断状态</b> 从机3线模式下用来指示传输已经开始。该位是 SPI_CNTRL[11]的镜像. 0 = 传输还没有开始 1 = 从机3线模式下传输已经开始。当传输完成或者写1到该位该位将被清0.
[10:5]	保留	保留.
[4]	TX_INTSTS	<b>发送 FIFO 阈值中断状态 (只读)</b> 0 = 发送 FIFO 中有效数据个数大于 TX_THRESHOLD 的设定. 1 = 发送 FIFO 中有效数据个数小于或者等于 TX_THRESHOLD 的设定. 注: 如果 TX_INTEN = 1 并且 TX_INTSTS = 1, SPI 控制器将产生SPI 中断.
[3]	保留	保留.
[2]	RX_OVERRUN	<b>接收 FIFO 溢出状态</b> 当接收 FIFO 已经满了, 后续的数据将被丢弃并且该位将被设为1. 注: 该位写1清0.
[1]	保留	保留.
[0]	RX_INTSTS	<b>接收 FIFO 阈值中断状态(只读)</b> 0 = 接收FIFO中有效数据个数小于或者等于RX_THRESHOLD 的值. 1 = 接收FIFO中有效数据个数大于RX_THRESHOLD 的设定值. 注: 如果 RX_INTEN = 1 并且 RX_INTSTS = 1, SPI 控制器将产生SPI中断.

## 6.13 定时器控制器(TMR)

### 6.13.1 概述

定时器控制器包括4组32位的定时器，TIMER0~TIMER3，方便用户实现定时控制应用。定时器模块可支持例如频率测量，时间延迟，时钟产生，时间计数和间隔测量等功能。

### 6.13.2 特性

- 4 组 32-位定时器，带24位上数计数器和一个8位的预分频计数器
- 每个定时器都有独立的时钟源
- 4种工作模式：单脉冲模式(one-shot)，周期模式(periodic)，反转输出模式(toggle)和连续计数(continuous counting)模式
- 超时周期= (定时器时钟源的周期) \* (8-bit 预分频 + 1) \* (24-bit TCMP)
- 最大计数周期 =  $(1 / T \text{ MHz}) * (2^8) * (2^{24})$ , T 是定时器时钟源的周期
- 24位上数计数器的值，可通过TDR (定时器数据寄存器) 读取
- 支持事件计数功能，可以数外部输入信号的事件个数(T0~T3)
- 24位捕获值可以通过TCAP(定时器捕获数据寄存器)读取
- 支持外部捕获引脚(T0EX~T3EX)用于间隔测量
- 支持外部捕获引脚(T0EX~T3EX)用于复位24位上数计数器
- 下列新特性只有M05xxDN/DE才有：
  - ◆ 定时器中断支持将芯片从空闲/睡眠模式唤醒
  - ◆ 当ACMP输出信号发生反转时可以触发定时器内部捕获
  - ◆ 支持Inter-Timer触发模式

### 6.13.3 定时器方框图

定时器控制器框图和时钟源控制如下所示。

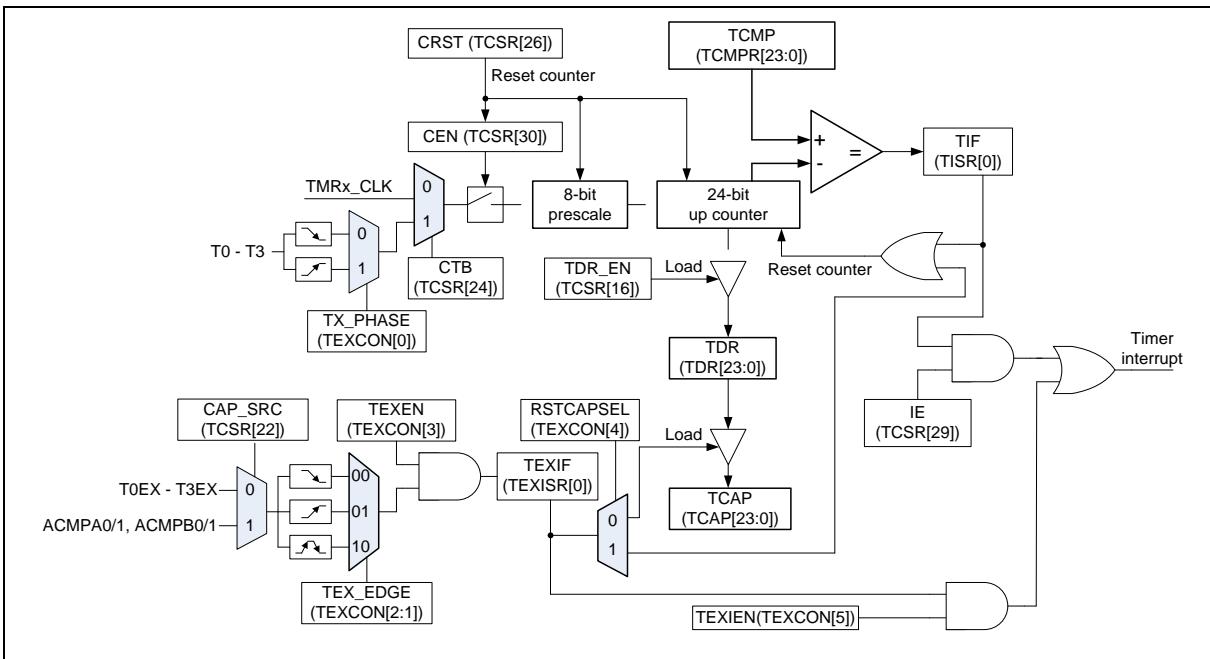


图 6-93 定时器控制器框图

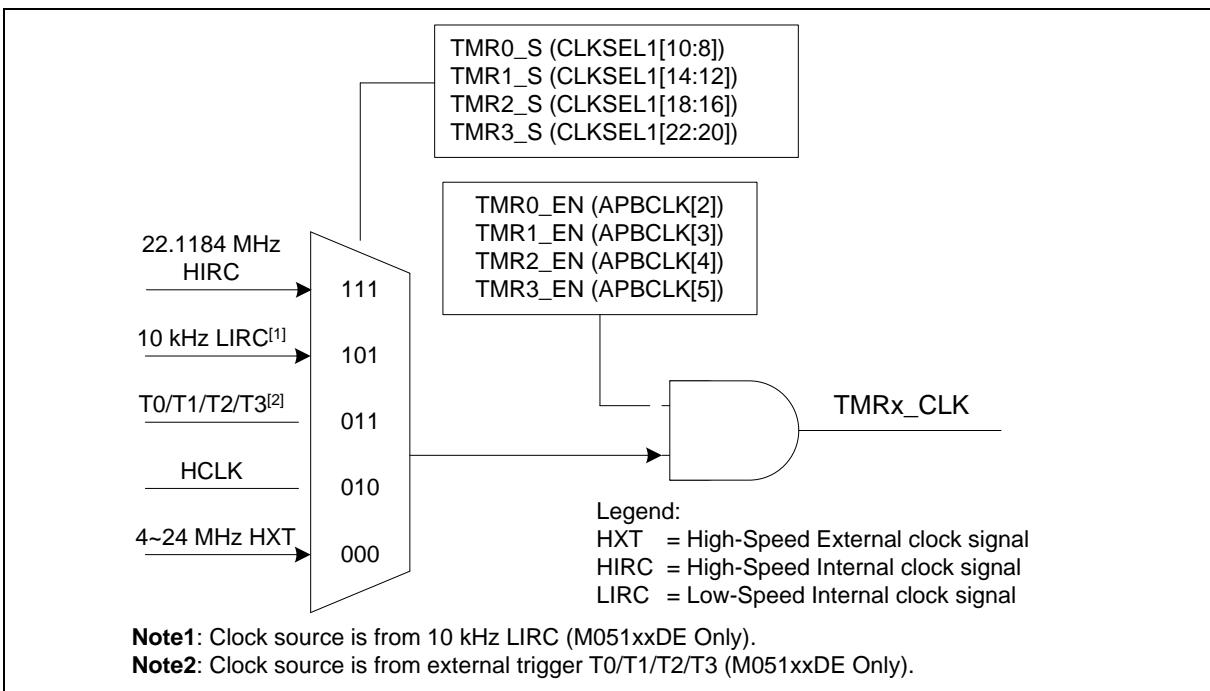


图 6-94 定时器控制器时钟源

### 6.13.4 基本配置

Timer0 ~ Timer3 的时钟在APBCLK[5:2] 中使能，Timer0的时钟源在CLKSEL1[10:8]中选择，Timer1的时钟源在CLKSEL1[14:12]中选择，Timer2的时钟源在CLKSEL1[18:16]中选择，Timer3的时钟源在CLKSEL1[22:20]中选择。

### 6.13.5 功能描述

#### 6.13.5.1 定时器中断标志

定时器控制器支持2种中断标志：一个是TIF标志，当计数器的值(TDR)等于定时器比较寄存器的值(TCMP)时被置位；另一个是TEXIF标志，当TxEX引脚上的信号发生符合TEX\_EDGE的设定的转变时被置位。

#### 6.13.5.2 定时器操作模式

定时器控制器提供4种工作模式，单脉冲(one-shot)模式、周期(periodic)模式、反转(toggle)和连续计数(continuous counting)模式。每种操作模式如下所述：

##### 单脉冲模式

如果定时器工作在单脉冲模式(TCSR[28:27] = 00)且CEN (TCSR[30]定时器使能位)置1，定时器的计数器开始上数。一旦TDR的值达到定时器比较寄存器 (TCMPR) 的值，TDR的值和CEN位都被硬件清为0，然后定时器停止计数。如果IE (中断使能位)置1，则定时器中断标志TIF置位，中断发生。

##### 周期模式

如果定时器工作在周期模式(TCSR[28:27] = 01)且CEN (定时器使能位)置1，定时器计数器开始上数。一旦TDR的值达到定时器比较寄存器 (TCMPR) 的值，TDR的值将被硬件清0，计数器再次开始计数。同时如果IE使能，TIF标志将被设为1，定时器中断将发送。该模式下定时器一直计数直到TCMP比较，直到CEN被清0。

##### 反转输出模式

如果定时器工作在反转输出模式(TCSR[28:27] = 10)且CEN (定时器使能位)置1，定时器计数器开始上数。反转输出模式和周期模式几乎相同，除了当TIF被置为1时反转输出将从T0 ~ T3输出信号，就是说引脚上的信号将发生反转，所以反转输出模式输出的信号占空比是50%。

##### 连续计数模式

如果定时器工作在连续计数模式(TCSR[28:27] = 11)且CEN (定时器使能位)置1，定时器开始上数。一旦TDR的值等于TCMP的值，TIF被置并且TDR继续上数。同时，如果IE使能，定时器中断将发生，用户可以立即改变TCMP的值而无需关闭计数器并重新计数。

例如，TCMPR的值先被设置为80(TCMPR的值应当小于 $2^{24}$ -1并且大于1)，当TDR的值等于80时，如果IE (中断使能位)设置为1，定时器产生中断，TIF(定时器中断标志)将被置位，且CEN 保持为1(持续使能计数)，但是TDR的值不会返回到零，而是继续计数81, 82, 83, ··· to  $2^{24}$ -1, 0, 1, 2, 3, ··· to  $2^{24}$ -1。接下来，如果用户设置TCMPR为200，且清除TIF，当TDR的值达到200，定时器再次中断发生，TIF被置位，并发生中断。最后，用户设置TCMPR为500，并再一次清零TIF，当TDR的值达到500，定时器中断发生，TIF被置位，再次产生中断。

从应用的角度看，中断的产生取决于TCMPR。在该模式下，定时器计数是连续的，所以这种操作

模式被称为连续计数模式。

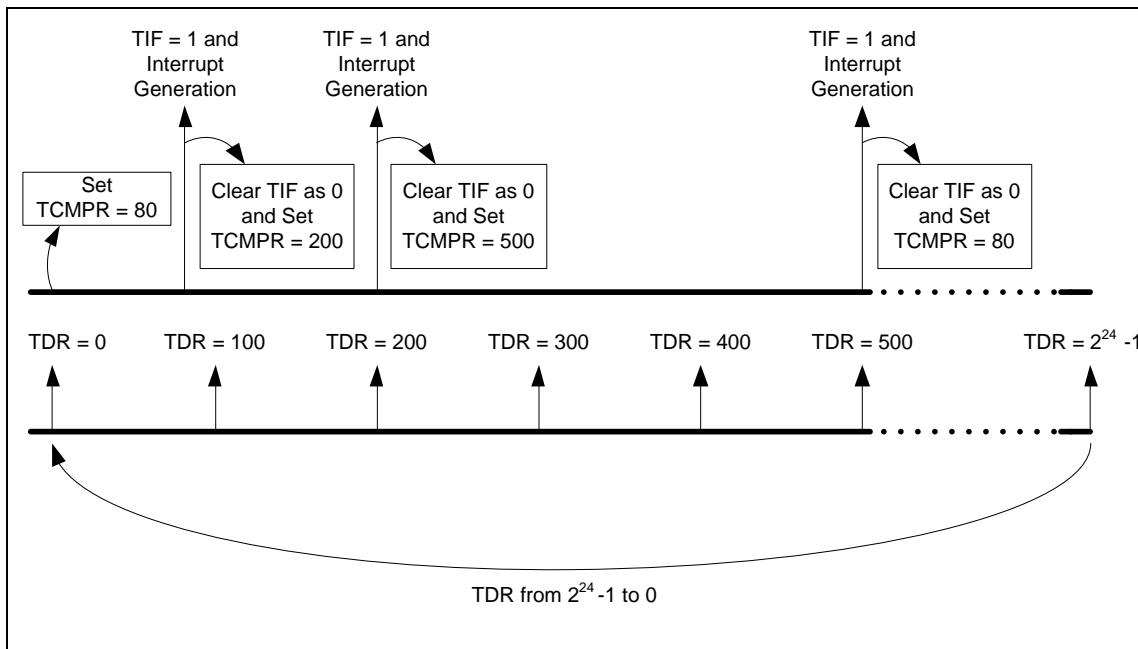


图6-95 连续计数模式

#### 6.13.5.3 事件计数功能

定时器提供一个功能可以用来数外部T0~T3引脚的事件个数，被称为事件计数功能。事件的个数反映在TDR寄存器中。该功能中，CTB(TCSR[24])位应该被设为1，外设时钟源应该选HCLK。

软件可以通过TCDB(TEXCON[7])位使能/关闭Tx引脚的防弹跳功能。如果禁止防弹跳功能，输入事件的频率应该小于HCLK频率的1/3；如果使能防弹跳功能，输入事件的频率应该小于HCLK频率的1/8。否则，返回的TDR的值将不正确。软件通过TX\_PHASE (TEXCON[0])位可以选择检测的Tx引脚的边沿相位。

事件计数模式下，定时器计数模式可以选择单脉冲模式、周期模式和连续计数模式。

#### 6.13.5.4 事件捕获功能

计数器提供输入捕获功能，当TxEx引脚(x= 0~3)上的信号发生改变时，可以捕获TDR的值存到TCAP寄存器中。为了将TxEX引脚用作事件捕获，RSTCAPSEL (TEXCON[4])应该设为0并且定时器时钟源应该选择HCLK。

软件可以通过TEXDB(TEXCON[6])位使能/关闭TxEx引脚的防弹跳功能。如果禁止防弹跳功能，输入事件的频率应该小于HCLK频率的1/3；如果使能防弹跳功能，输入事件的频率应该小于HCLK频率的1/8。否则，捕获功能可能工作的不正常。软件通过TEX\_EDGE (TEXCON[2:1])位可以选择检测的TxEx引脚的边沿相位。

事件计数模式下，软件不必考虑计数器工作在哪种模式下，只要TxEX引脚上的信号发生变化(符合用户设定)，捕获事件就会发生。

#### 6.13.5.5 事件复位计数器功能

定时器提供复位计数器功能，当TxEX引脚(x= 0~3)的信号发生边沿转变，可以复位TDR的值。该模

式下大多设定和事件捕获功能一样，除了RSTCAPSEL (TEXCON[4]) 应该设为1。

#### 6.13.5.6 Inter-Timer 触发捕获功能 (M05xxDN/DE)

该模式下，Timer0/2 被迫使工作在计数模式，计数外部事件，并且产生一个内部信号 (INTR\_TMR\_TRG) 来触发 Timer1/3 开始或者停止计数。Timer1/3 也被迫工作在捕获模式，由 Timer0/2计数器状态触发开始/停止计数。

使能 Timer0 Inter-timer 触发捕获功能，Timer1也被迫工作在触发-计数捕获功能；使能 Timer2 Inter-timer 触发捕获功能，Timer3也被迫工作在触发-计数捕获功能。

##### 启动触发

当Timer0/2 中的INTR\_TRG\_EN 被置位时，当Timer0/2 24-位计数器的值(TDR)从0x0变为0x1时，INTR\_TMR\_TRG信号将变为高电平，Timer1/3计数器将立即自动开始计数。

##### 停止触发

当 Timer0/2 TDR 的值等于 Timer0/2 TCMPR 的值时，Timer0/2 的INTR\_TMR\_TRG 信号将变为低电平。然后 Timer0/2 计数器功能将被关闭并且INTR\_TRG\_EN 将被硬件清为0。之后Timer1/3 也将停止计数。同时，Timer1/3 TDR 的值将被存到 Timer1/3 TCAP 寄存器中。

用户可以使用 inter-timer 触发功能来精确测量外部事件(Tx) 的周期。下图显示了Timer0 Inter-Timer触发捕获模式的流程，此模式下，Timer0自动工作在事件计数功能，Timer1自动当作触发计数捕获功能。

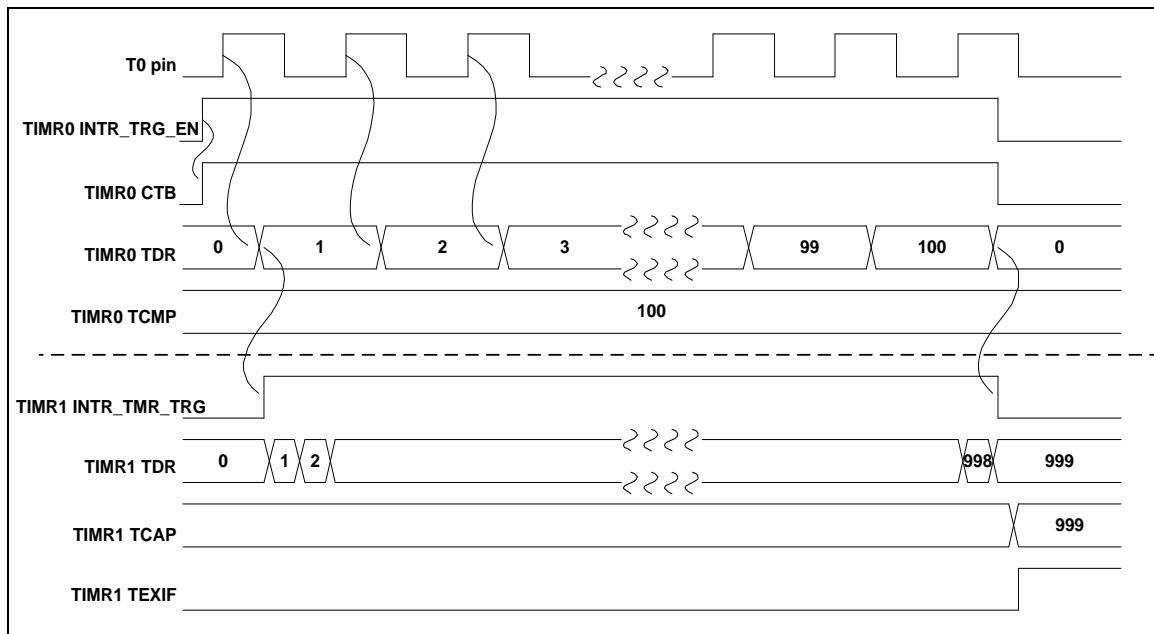


图 6-96 Inter-Timer 触发捕获时序图

#### 6.13.5.7 来自ACMP的内部捕获触发(只 M05xxDN/DE)

事件捕获功能也可以由内部输出信号ACMPA0(Timer0), ACMPA1(Timer1), ACMPB0(Timer2) 或者ACMPB1(Timer3)的反转触发。捕获功能细节设定和事件捕获功能一样。

#### 6.13.6 寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
定时器基地址:				
TMR_BA01 = 0x4001_0000				
TMR_BA23 = 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 捕获数据寄存器	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捕获数据寄存器	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捕获数据寄存器	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捕获数据寄存器	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
---------	---------------	-----	-----------------	-------------

### 6.13.7 寄存器描述

#### 定时器控制与状态寄存器 (TCSR)

寄存器	偏移量	R/W	描述				复位后的值
TCSR0	TMR_BA01+000	R/W	Timer0 控制与状态寄存器				0x0000_0005
TCSR1	TMR_BA01+020	R/W	Timer1控制与状态寄存器				0x0000_0005
TCSR2	TMR_BA23+000	R/W	Timer2控制与状态寄存器				0x0000_0005
TCSR3	TMR_BA23+020	R/W	Timer3控制与状态寄存器				0x0000_0005

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

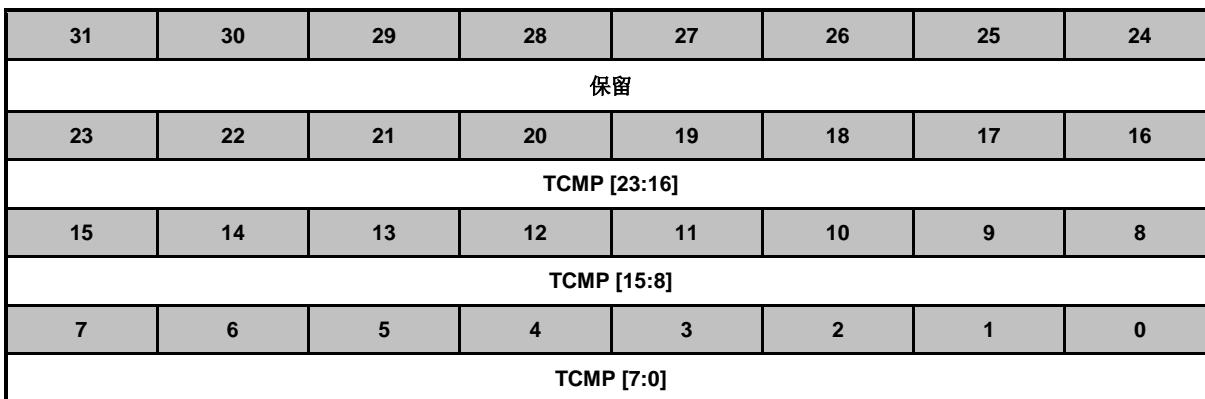
Bits	描述		
[31]	<b>DBGACK_TMR</b>  ICE 调试模式下禁止应答 (写保护) 0 = ICE 调试模式应答影响定时器计数。 ICE调试模式下，定时器计数器的值将维持不变。 1 = ICE 调试模式应答关闭。 无论是否在ICE调试模式下，定时器计数器的值都保持正常计数。		
[30]	<b>CEN</b>  计数器使能控制 0 = 停止/暂停计数 1 = 开始计数 <b>注1:</b> 在停止状态，设置CEN为1，使能24位计数器从上次停止的计数值继续计数。 <b>注2:</b> 在单脉冲模式下(MODE[28:27]=00b)，当相应的定时器中断产生时(IE[29]=1b)，TIF等于1，该位由硬件自动清零。		
[29]	<b>IE</b>  中断使能 0 = 禁止定时器中断 1 = 使能定时器中断 如果定时器中断使能，当计数值与TCMPR寄存器内数值相同时，TIF被置为1，并触发中断。		
[28:27]	<b>MODE</b>  定时器工作模式 <table border="1" style="width: 100%;"><tr><td style="width: 50%;">模式</td><td style="width: 50%;">定时器工作模式</td></tr></table>	模式	定时器工作模式
模式	定时器工作模式		

		00 01 10 11	单脉冲模式(one-shot)。仅触发一次中断 (IE 使能),然后CEN自动清除为0. 周期模式(period)。中断(IE 使能)将周期性产生。 反转输出模式。如果IE使能, 中断将周期性产生, 反转信号 (tout) 不停输出, 占空比为50%。 连续计数模式。当TDR等于TCMPR时, 中断将产生 (如果IE使能) . 然而, 24-bit上数计数器仍继续上数..	
[26]	<b>CRST</b>	计数器复位  设置该位将复位定时器预分频器, 计数器, 并使CEN为0. 0 = 无动作. 1 = 复位定时器的预分频计数器, 内部24位向上计数器和CEN位		
[25]	<b>CACT</b>	定时器工作状态 (只读)  该位表示当前定时器计数器的状态。 0 = 定时器未工作。 1 = 定时器工作中。		
[24]	<b>CTB</b>	计数模式使能  该位是外部计数功能使能位。当计数器用作事件计数功能时, 这个比特应该被设为1, 并选择HCLK作为时钟源。计数器检测的外部引脚相位可以通过TX_PHASE域设定为上升沿或者下降沿。 1 = 使能事件计数功能 0 = 关闭事件计数功能		
[23]	<b>WAKE_EN</b>	唤醒功能使能控制 (M05xxDN/DE)  0 = 禁止唤醒触发事件. 1 = 使能唤醒触发功能		
[22]	<b>CAP_SRC</b>	捕获引脚源选择 (M05xxDN/DE)  0 = 捕获功能来自TxEx引脚 . 1 = 捕获功能来自内部 ACMPx 信号输出.		
[21]	<b>TOUT_SEL</b>	反转输出引脚选择 (M05xxDN/DE)  0 = 反转输出到Tx引脚. 1 = 反转输出到TxEx 引脚.		
[20]	<b>PERIODIC_SEL</b>	周期模式行为选择使能 (M05xxDN/DE )  0 = 周期模式行为选择被关闭. 周期模式下当定时器正在工作, 此时用户更新 TCMP, TDR将被复位成默认值。 1 = 使能周期模式行为选择. 周期模式下当定时器正在工作, 此时用户更新 TCMP, 定时器行为如下: 如果更新的TCMP 的值 > TDR, TCMP 将被更新并且TDR继续计数。 如果更新的TCMP 的值 = TDR, 定时器超时中断将立即发生。 如果更新的TCMP 的值< TDR, TDR将被复位成默认值。		
[19]	<b>INTR_TRG_EN</b>	Inter-Timer 触发模式使能控制 (M05xxDN/DE)  设置该位将使能 inter-timer 触发捕获功能. Timer0/2 将工作在事件计数模式, 计数外部事件。Timer1/3 将工作在捕获功能-触发计数模式		

		模式. 0 = 禁止Inter-Timer 触发捕获模式. 1 = 使能Inter-Timer 触发捕获模式. 注: 对于Timer1/3, 该位被忽略并且读到的值总是0.
[23:17]	保留	保留
[16]	<b>TDR_EN</b>	<b>数据读取使能</b> 当置位TDR_EN, TDR (定时器数据寄存器) 将不断更新为24位向上计数器的值 1 = 使能定时器数据寄存器 更新 0 = 禁止定时器数据寄存器 更新
[15:8]	保留	保留
[7:0]	<b>PRESCALE</b>	<b>预分频计数器</b> 输入的时钟源被除以(PRESCALE+1)进行预分频。如果PRESCALE =0, 不进行预分频.

定时器比较寄存器(TCMPR)

寄存器	偏移量	R/W	描述	复位后的值
TCMPRO	TMR_BA01+0x04	R/W	Timer0 比较寄存器	0x0000_0000
TCMPR1	TMR_BA01+0x24	R/W	Timer1比较寄存器	0x0000_0000
TCMPR2	TMR_BA23+0x04	R/W	Timer2比较寄存器	0x0000_0000
TCMPR3	TMR_BA23+0x24	R/W	Timer3比较寄存器	0x0000_0000



Bits	描述	
[31:24]	保留	保留
[23:0]	TCMP	<p><b>定时器比较值</b></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 的值匹配时，如果 TCSR.IE[29]=1，TIF 将被置为 1，并产生定时器中断请求。</p> <p>定时溢出周期 = (定时器输入时钟周期) * (8-bit PRESCALE + 1) * (24-bit TCMP)</p> <p><b>注1:</b> 不能在 TCMP 里写 0x0 或 0x1，否则内核将运行到未知状态。</p> <p><b>注2:</b> 如果定时器工作在连续计数模式，用户写一个新的值到 TCMP，24 位上数计数器将保持继续上数；如果计数器工作在其它模式包括周期模式，软件向该寄存器写入新的值，24-bit 上数计数器将被复位并重新开始计数。M05xxDN/DE 中周期模式的行为由 PERIODIC_SEL (TCSR[20]) 决定</p>

**定时器中断状态寄存器(TISR)**

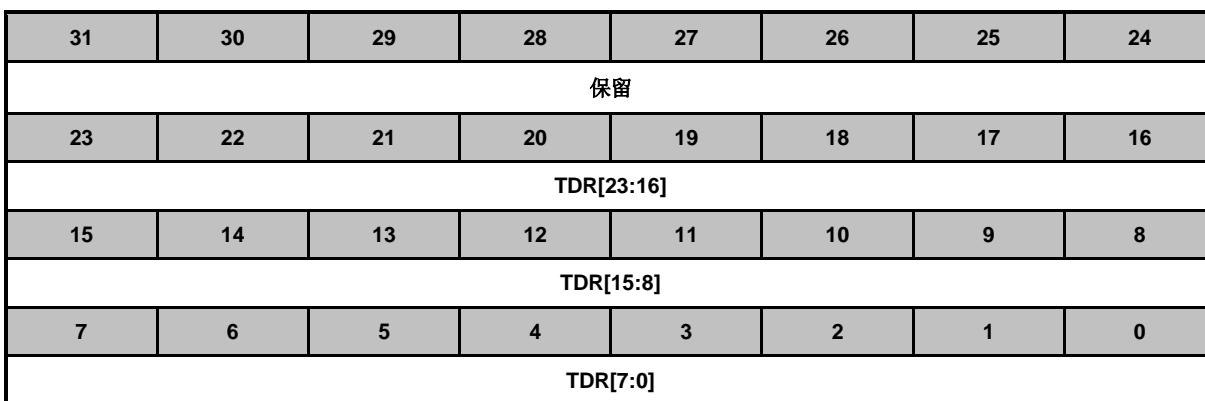
寄存器	偏移量	R/W	描述	复位后的值
TISR0	TMR_BA01+0x08	R/W	Timer0 中断状态寄存器	0x0000_0000
TISR1	TMR_BA01+0x28	R/W	Timer1 中断状态寄存器	0x0000_0000
TISR2	TMR_BA23+0x08	R/W	Timer2 中断状态寄存器	0x0000_0000
TISR3	TMR_BA23+0x28	R/W	Timer3 中断状态寄存器	0x0000_0000

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						TWF	TIF

Bits	描述	
[31:2]	保留	保留
[2]	TWF	<p><b>定时器唤醒标志 (M05xxDN/DE )</b></p> <p>该位指示中断唤醒标志状态.</p> <p>0 = 定时器没有导致芯片唤醒.</p> <p>1 = 定时器将芯片从空闲/睡眠模式唤醒.</p> <p>注: 该位写1清0</p>
[0]	TIF	<p><b>定时器中断标志</b></p> <p>定时器中断状态位.</p> <p>当内部24位计数器与TCMP的值匹配时, TIF由硬件置位, 写1清该位.</p> <p>0 = 无影响</p> <p>1 = TDR的值等于TCMP的值</p> <p>注: 该位写1清0</p>

定时器数据寄存器(TDR)

寄存器	偏移量	R/W	描述	复位后的值
TDR0	TMR_BA01+0x0C	R/W	Timer0数据寄存器	0x0000_0000
TDR1	TMR_BA01+0x2C	R/W	Timer1数据寄存器	0x0000_0000
TDR2	TMR_BA23+0x0C	R/W	Timer2数据寄存器	0x0000_0000
TDR3	TMR_BA23+0x2C	R/W	Timer3数据寄存器	0x0000_0000



Bits	描述	
[31:24]	保留	保留
[23:0]	TDR	定时器数据寄存器 TCSR.TDR_EN 置1时， 用户可以读TDR寄存器取得当前24位上数计数器的值

定时器捕捉数据寄存器 (TCAP)

寄存器	偏移	R/W	描述	复位后的值
TCAP0	TMR_BA01+0x10	R/W	Timer0 捕获数据寄存器	0x0000_0000
TCAP1	TMR_BA01+0x30	R/W	Timer1捕获数据寄存器	0x0000_0000
TCAP2	TMR_BA23+0x10	R/W	Timer2捕获数据寄存器	0x0000_0000
TCAP3	TMR_BA23+0x30	R/W	Timer3捕获数据寄存器	0x0000_0000

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

Bits	描述	
[31:24]	保留	保留
[23:0]	TCAP	<p>定时器捕获数据寄存器</p> <p>当 TEXEN (TEXCON[3]) 为1, RSTCAPSEL (TTXCON[4]) 为 0时, TEX引脚上的信号发生了TEX_EDGE(TEXCON[2:1])说明的转变时, 内部 24-位上数计数器的值将被加载到 TCAP。 用户可以读这个寄存器得到计数器的值。</p>

定时器外部控制寄存器 (TEXCON)

寄存器	偏移	R/W	描述	复位后的值
TEXCON0	TMR_BA01+0x14	R/W	Timer0 外部控制寄存器	0x0000_0000
TEXCON1	TMR_BA01+0x34	R/W	Timer1外部控制寄存器	0x0000_0000
TEXCON2	TMR_BA23+0x14	R/W	Timer2外部控制寄存器	0x0000_0000
TEXCON3	TMR_BA23+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	

Bits	描述	
[31:8]	保留	保留
[7]	TCDB	<p>定时器外部计数输入引脚防反弹使能控制 1 = 使能Tx引脚防反弹功能 0 = 关闭Tx引脚防反弹功能 如果这个bit 被使能, T0~T3 引脚的输入信号边沿将被防反弹电路检测.</p>
[6]	TEXDB	<p>定时器外部捕获输入引脚防反弹使能控制 1 = 使能TxEX引脚防反弹功能 0 = 关闭TxEX引脚防反弹功能 如果这个bit 被使能, T0EX~T3EX 引脚的输入信号边沿将被防反弹电路检测</p>
[5]	TEXIEN	<p>定时器外部捕获中断使能控制 1 = 使能TxEX引脚定时器外部中断 0 = 禁止TxEX引脚定时器外部中断 如果定时器外部捕获中断TEXIEN被使能, 当TxEX引脚上发生TEX_EDGE(TEXCON[2:1])设定的信号转变时, 中断将发生 例如, 当 TEXIEN = 1, TEXEN = 1, TEX_EDGE = 00时, TX引脚上1 到 0的信号转变将导致TEXIF(TEXISR[0]) 中断标志被设, 然后中断将发生.</p>
[4]	RSTCAPSEL	<p>定时器外部复位计数器/外部捕获模式选择 1 = TxEX 引脚上的信号转变被用来复位内部24位计数器. 0 = TxEX 引脚上的信号转变被用作定时器捕获功能.</p>

[3]	<b>TEXEN</b>	定时器外部引脚功能使能。 该位用来使能TxEX引脚上的复位/捕获功能。 1 = TxEX引脚上的信号转变将导致定时器计数器捕获或者复位。 0 = TxEX 引脚被忽略。
[2:1]	<b>TEX_EDGE</b>	定时器外部捕获引脚检测边沿选择 00 = TxEX引脚上1到0的转变将被检测。 01 = TxEX引脚上0到1的转变将被检测。 10 = TxEX引脚上1到0和0到1的转变都将被检测。 11 = 保留。
[0]	<b>TX_PHASE</b>	定时器外部计数引脚相位 检测选择 这个比特用来指示检测的Tx引脚的信号相位。 1 = Tx引脚上一个上升沿转变将被计数。 0 = Tx引脚上一个下降沿转变将被计数。

定时器外部中断状态寄存器 (TEXISR)

寄存器	偏移	R/W	描述	复位后的值
TEXISR0	TMR_BA01+0x18	R/W	Timer0 外部中断状态寄存器	0x0000_0000
TEXISR1	TMR_BA01+0x38	R/W	Timer1外部中断状态寄存器	0x0000_0000
TEXISR2	TMR_BA23+0x18	R/W	Timer2外部中断状态寄存器	0x0000_0000
TEXISR3	TMR_BA23+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

Bits	描述	
[31:1]	保留	保留
[0]	TEXIF	<p><b>定时器外部捕获中断标志</b> 这个比特指示定时器外部捕获中断状态. 当TEXEN (TEXCON[3]) 等于 1, TxEX 引脚上发生TEX_EDGE(TEXCON[2:1]) 设定的信号 转变时, 硬件将置位这个bit。这个bit写1清0. 例如, TEXEN = 1, TEX_EDGE = 00, TX引脚上一个 1 到 0的转变将导致TEXIF 被置。 1 = TxEX引脚中断发生 0 = TxEX引脚中断没有发生 注: 该位写1清0.</p>



## 6.14 UART接口控制器(UART)

### 6.14.1 概述

NuMicro M05xxBN/DN/DE提供2个通用异步收/发器（UART）通道，UART支持普通速度UART，并支持流控制。UART控制器对从外设收到的数据执行串到并的转换，对来自CPU的数据执行并到串的转换。UART控制器同时支持IrDA SIR 功能、LIN主/从功能和RS-485功能。每个UART控制器支持7种类型的中断。

### 6.14.2 特性

- 全双工，异步通信
- 独立的接收/发送16字节FIFO用户装载数据
- 支持硬件自动流控/流控制功能(CTS, RTS)和可编程的RTS流控制触发电平
- 可编程的接收缓冲触发级别
- 每个通道都支持独立的可编程的波特率发生器
- 支持CTS 唤醒功能
- 支持8位接收缓冲超时功能
- 通过设置DLY (UA\_TOR [15:8]) 寄存器可以编程在上一个停止与下一个开始位之间数据发送的延迟时间
- 支持break错误，帧错误，奇偶校验错误和接收/发送缓冲溢出检测功能
- 完全可编程的串行接口特性
  - 可编程的数据位, 5, 6, 7, 8位
  - 可编程的奇偶校验位, 偶校验、奇校验、无校验位或stick校验位 发生和检测
  - 可编程停止位, 1, 1.5, 或 2 停止位 产生
- 支持IrDA SIR 功能
  - 普通模式下支持 3/16位持续时间
- 支持LIN功能
  - 支持LIN主/从模式
  - 支持发送端可编程的break产生功能
  - 支持接收端break检测功能
- 支持RS-485 模式.
  - 支持 RS-485 9位模式
  - 支持硬件或软件编程RTS引脚控制收发器的传输方向

### 6.14.3 UART 框图

UART 时钟控制和框图如下所示。

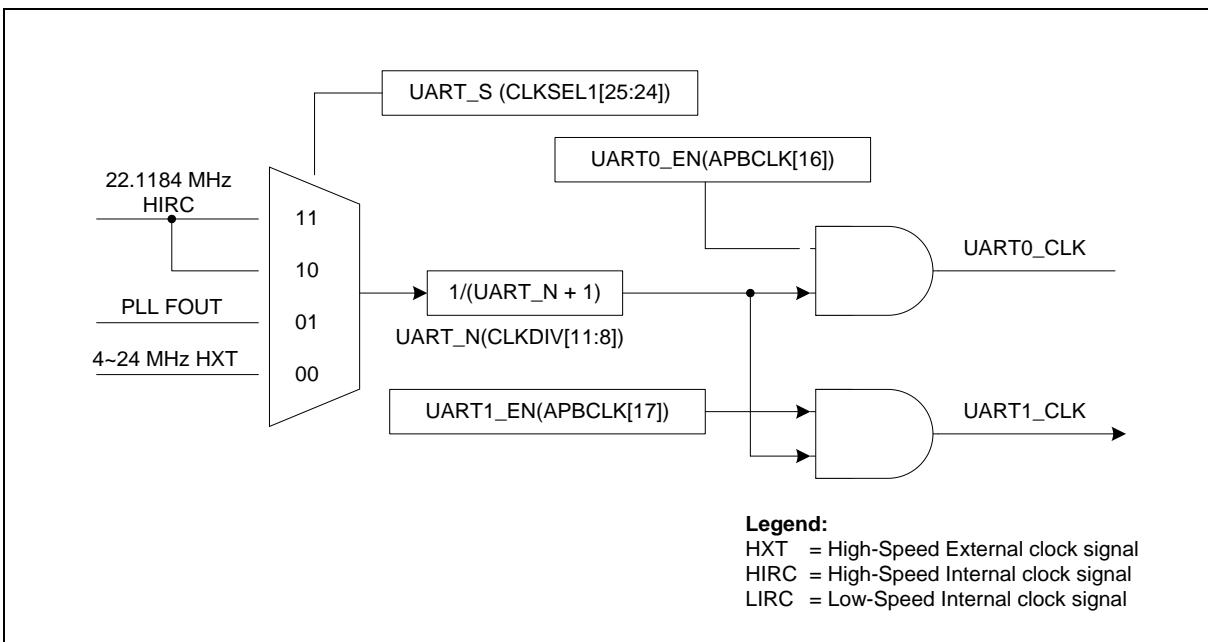


图 6-97 UART 时钟控制框图

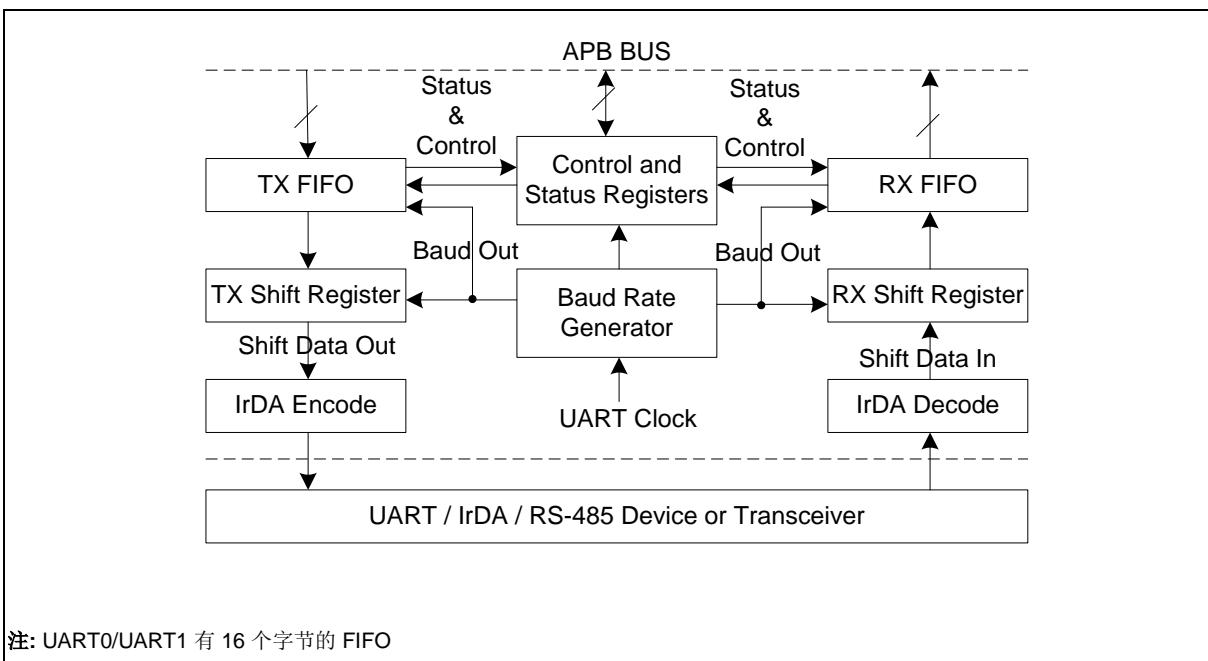


图 6-98 UART 控制器框图

每个方块的细节如下所述:

#### TX\_FIFO

发送用一个16字节的FIFO做缓存来降低CPU的中断数量。

**RX\_FIFO**

接收用一个16字节(每个字节加3个比特的错误比特)的FIFO做缓存来降低CPU的中断数量.

**TX移位寄存器**

此模块控制移位正在发送的数据串行输出.

**RX移位寄存器**

此模块控制移位正在接收的数据串行输入.

**Modem控制寄存器**

该寄存器控制与MODEM 或者数据传输转换器(或者一个MODEM模拟器)的接口.

**波特率发生器**

将外部时钟除以一个除数来获得需要的波特率时钟,参考波特率方程.

**IrDA 编码**

IrDA 编码控制模块.

**IrDA 解码**

IrDA 解码控制模块.

**控制和状态寄存器**

该域是一组寄存器, 包括用于发送和接收的FIFO 控制寄存器 (UA\_FCR), FIFO 状态寄存器 (UA\_FSR), 和线控制寄存器 (UA\_LCR). 超时控制寄存器 (UA\_TOR) 应用于标识超时中断产生的条件. 该组寄存器还包括中断使能寄存器 (UA\_IER) 和中断状态寄存器 (UA\_ISR) 来使能或者禁止中断响应并且识别发生的中断. 共有7种中断: 发送FIFO为空中断 (THRE\_INT), 接收阀值到达中断 (RDA\_INT), 线状态中断 (校验错误, 帧错误和break中断) (RLS\_INT), 超时中断 (TOUT\_INT), MODEM/唤醒状态中断 (MODEM\_INT), 缓冲错误中断 (BUF\_ERR\_INT) 和LIN接收break域中断(LIN\_RX\_BREAK\_INT).

自动流控框图如下所示。

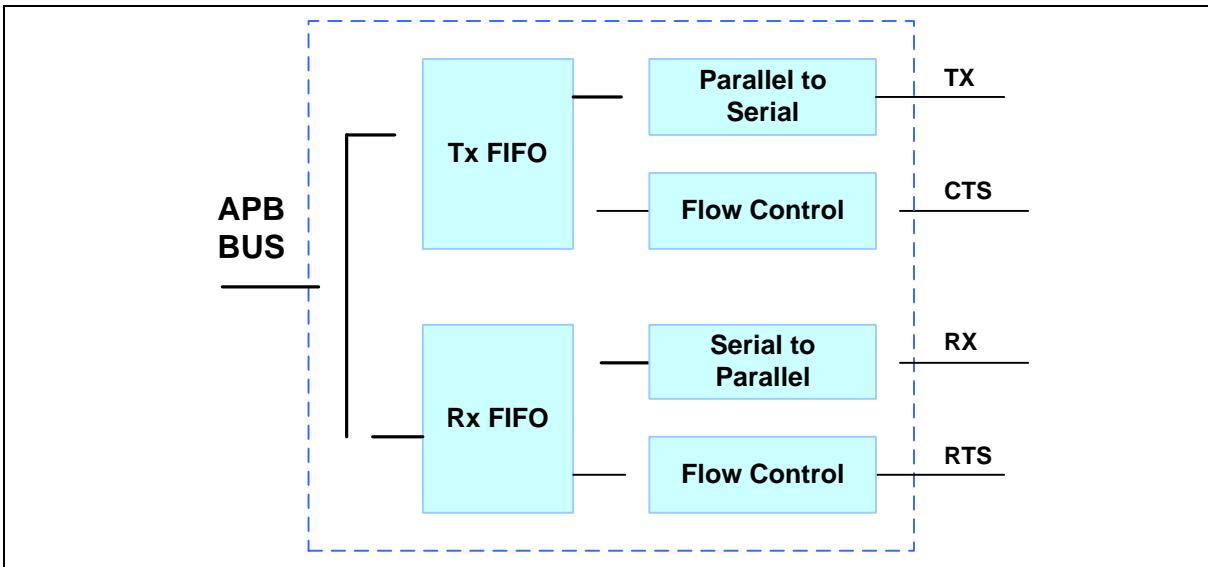


图 6-99 自动流控制框图

#### 6.14.4 基本配置

UART0功能引脚在P0\_MFP 寄存器中配置，UART1的引脚在P3\_MFP 寄存器中配置.

UART0 的时钟在UART0\_EN(APBCLK[16]) 中使能，UART1在UART1\_EN (APBCLK[17]) 中使能.

UART 控制器时钟源在UART\_S(CLKSEL[25:24])中选择.

UART 控制器时钟预分频值由UART\_N(CLKDIV[11:8])决定.

#### 6.14.5 功能描述

UART 控制器支持4种功能包括：UART、IrDA、LIN 和 RS-485 模式。用户可以通过设定UA\_FUN\_SEL寄存器选择一种功能。4种功能模式将在下面的章节描述。

##### 6.14.5.1 UART 控制器波特率发生器

UART控制器包括一个可编程的波特率发生器，它可以将输入时钟分频来得到收/发器需要的时钟。 波特率公式是 波特率 =  $\text{UART\_CLK} / M * [BRD + 2]$ ， 其中M和BRD在波特率分频寄存器UA\_BAUD中定义. 下表列出了不同条件下的波特率方程和UART波特率参数设置。IrDA模式下，波特率发生器必须工作在模式0

Mode	DIV_X_EN	DIV_X_ONE	Divider X	BRD	M	Baud Rate Equation
Mode 0	0	0	B	A	16	$\text{UART\_CLK} / [16 * (A+2)]$
Mode 1	1	0	B	A	B+1	$\text{UART\_CLK} / [(B+1) * (A+2)]$ , B must >= 8
Mode 2	1	1	Don't care	A	1	$\text{UART\_CLK} / (A+2)$ , A must >=8

表 6-15 UART 波特率方程表

UART Peripheral Clock = 22.1184 MHz			
Baud Rate	Mode 0	Mode 1	Mode 2
921600	Not support	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6-16 UART波特率参数设置表

UART Peripheral Clock = 22.1184 MHz			
Baud Rate	Mode 0	Mode 1	Mode 2
921600	Not support	0x2B00_0000	0x3000_0016
460800	0x0000_0001	0x2F00_0001 0x2B00_0002	0x3000_002E
230400	0x0000_0004	0x2F00_0004 0x2B00_0006	0x3000_005E
115200	0x0000_000A	0x2F00_000A 0x2B00_000E	0x3000_00BE
57600	0x0000_0016	0x2F00_0016 0x2B00_001E	0x3000_017E
38400	0x0000_0022	0x2800_003E 0x2B00_002E 0x2F00_0022	0x3000_023E
19200	0x0000_0046	0x2800_007E 0x2B00_005E 0x2F00_0046	0x3000_047E
9600	0x0000_008E	0x2800_00FE 0x2B00_00BE 0x2F00_008E	0x3000_08FE

4800	0x0000_011E	0x2800_01FE 0x2B00_017E 0x2F00_011E	0x3000_11FE
------	-------------	---	-------------

表 6-17 UART 波特率寄存器设置表

#### 6.14.5.2 UART 控制器 FIFO 控制与状态

UART控制器内嵌一个16字节的发送FIFO (TX\_FIFO) 和一个16字节的接收 FIFO (RX\_FIFO)，可以降低CPU的中断次数。CPU可以随时读UART的状态。返回的状态信息包括正在被UART执行的传输操作的类型和条件，在接收数据时还可能发生3种错误(奇偶校验错误、帧错误、break中断)状况。FIFO控制与状态也支持所有的UART、IrDA、LIN 和 RS-485功能模式。

#### 6.14.5.3 UART 控制器唤醒功能

当芯片进入睡眠模式时，外部的 CTS 状态改变将唤醒芯片。任何模式下都可以使用该唤醒功能。用户必须使能MODEN\_INT 中断才能使用该唤醒功能。

#### 6.14.5.4 UART 控制器中断与状态

每个 UART 控制器支持7种中断，包括：

- 接收阙值级别到达中断(RDA\_INT)
- 发送FIFO 空中断 (THRE\_INT)
- 线状态中断 (校验错误，帧错误或者 break 中断) (RLS\_INT)
- MODEM/唤醒状态中断 (MODEM\_INT)
- 接收缓冲超时中断 (TOUT\_INT)
- 缓冲错误中断 (BUF\_ERR\_INT)
- LIN 检测到 break 域中断 (LIN\_RX\_BREAK\_INT)

下表描述了M05xxBN/DN/DE的中断源和标志。当中断标志被置并且中断使能位也被置时，中断将发生。中断发生时用户必须清除中断标志。

Interrupt Source	Interrupt Indicator	Interrupt Bit	Enable	Interrupt Flag	Flag Cleared By
Receive Data Available Interrupt	RDA_INT		RDA_IEN	<b>RDA_IF</b>	Read UA_RBR
Transmit Register Holding Empty Interrupt	THRE_INT		THRE_IEN	<b>THRE_IF</b>	Write UA_THR
Receive Line Status Interrupt	RLS_INT	RLS_IEN		<b>RLS_IF</b> = (BIF or FEF or PEF)	Writing '1' to RFR
				<b>RLS_IF</b> = RS485_ADD_DETF	Writing '1' to RS485_ADD_DETF
Modem Status Interrupt	MODEM_INT		MODEM_IEN	<b>MODEM_IF</b> = DCTS	Write '1' to DCTS
RX Time-out Interrupt	TOUT_INT		RTO_IEN	<b>TOUT_IF</b>	Read UA_RBR
Buffer Error Interrupt	BUF_ERR_INT	BUF_ERR_IEN		<b>BUF_ERR_IF</b> = (TX_OVER_IF or RX_OVER_IF)	Writing '1' to TX_OVER_IF / RX_OVER_IF
				<b>BUF_ERR_IF</b> = (BIF or FEF or PEF) (M05xxBN Only)	Writing '1' to RFR
LIN RX Break Field Detected interrupt	LIN_RX_BREAK_INT		LIN_RX_BRK_IEN	<b>LIN_RX_BREAK_IF</b>	Write '1' to LIN_RX_BREAK_IF

表 6-18 M05xxBN 中 UART 控制器中断源与标志列表

Interrupt Source	Interrupt Indicator	Interrupt Enable Bit	Interrupt Flag	Flag Cleared By
Receive Data Available Interrupt	RDA_INT	RDA_IEN	<b>RDA_IF</b>	Read UA_RBR
Transmit Register Empty	THRE_INT	THRE_IEN	<b>THRE_IF</b>	Write UA_THR
Receive Line Status Interrupt	RLS_INT	RLS_IEN	<b>RLS_IF</b> = (BIF or FEF or PEF)	Writing '1' to BIF/FEF/ PEF (M05xxDN/DE Only)
			<b>RLS_IF</b> = RS485_ADD_DETF	= Writing '1' to RS485_ADD_DETF
Modem Status Interrupt	MODEM_INT	MODEM_IEN	<b>MODEM_IF</b> = DCTS	Write '1' to DCTS
RX Time-out Interrupt	TOUT_INT	RTO_IEN	<b>TOUT_IF</b>	Read UA_RBR
Buffer Error Interrupt	BUF_ERR_INT	BUF_ERR_IEN	<b>BUF_ERR_IF</b> = (TX_OVER_IF or RX_OVER_IF)	Writing '1' to TX_OVER_IF / RX_OVER_IF
LIN RX Break Field Detected interrupt	LIN_RX_BREAK_INT	LIN_RX_BRK_IEN	<b>LIN_RX_BREAK_IF</b>	Write '1' to LIN_RX_BREAK_IF

表 6-19 M05xxDN/DE 中 UART 控制器中断源与标志列表

#### 6.14.5.5 UART 功能模式

UART 控制器提供UART 功能 (设置 UA\_FUN\_SEL [1:0] = '00' 使能 UART 功能模式)。UART 波特率最高达 1 Mbps。

UART 提供全双工异步传输。发送和接收都包含16 个字节的 FIFO 用于存放数据。接收时，用户可以编程接收缓冲触发级别和接收缓冲超时中断。发送数据时，上一个停止位和下一个开始位之间的延迟时间 可以通过 DLY (UA\_TOR [15:8]) 寄存器设定。UART 支持硬件自动流控和流控制功能 (CTS, RTS)，RTS 流控触发电平可编程，并且串行接口的所有特性都可以编程。

#### UART 线控功能

通过设置UA\_LCR寄存器，UART 控制器支持全部的串行接口特性。用户可以通过UA\_LCR 寄存器编程字长度、停止位和校验位。下表列出了UART 字长度和停止位长度设定和校验位设定。

NSB (UA_LCR[2])	WLS (UA_LCR[1:0])	Word Length (Bit)	Stop Length (Bit)
0	00	5	1
0	01	6	1
0	10	7	1
0	11	8	1
1	00	5	1.5
1	01	6	2
1	10	7	2
1	11	8	2

表 6-20 UART 字和停止位长度线控

Parity Type	SPE (UA_LCR[5])	EPE (UA_LCR[4])	PBE (UA_LCR[3])	Description
No Parity	x	x	0	No parity bit output.
Odd Parity	0	0	1	Odd Parity is calculated by adding all the "1's" in a data stream and adding a parity bit to the total bits, to make the total count an odd number.
Even Parity	0	1	1	Even Parity is calculated by adding all the "1's" in a data stream and adding a parity bit to the total bits, to make the count an even number.
Forced Mask Parity	1	0	1	Parity bit always logic 1. Parity bit on the serial byte is set to "1" regardless of total number of "1's" (even or odd counts).
Forced Space Parity	1	1	1	Parity bit always logic 0. Parity bit on the serial byte is set to "0" regardless of total number of "1's" (even or odd counts).

表 6-21 UART 校验位设定线控

### UART 自动流控功能

UART 通过两个信号支持自动流控功能，CTS (clear-to-send) 和 RTS (request-to-send)，来控制 UART 和外设(例如：Modem)之间的数据传输。当使能自动流控时，直到UART发送RTS到外设，UART 才被允许接收数据。当RX FIFO中的字节数量等于RTS\_TRI\_lev (UA\_FCR [19:16])的值时，RTS 信号变为无效。当UART检测到来自外设的CTS信号时，UART 才会发出数据。

下图说明了UART模式下 CTS 自动流控功能。用户必须设置AUTO\_CTS\_EN (UA\_IER [13]) 来使能 CTS 自动流控功能。LEV\_CTS (UA\_MCR [8]) 可以设置 CTS 引脚输入有效状态。当CTS引脚状态改变时，DCTS (UA\_MSR [0]) 将被置，然后 TX FIFO中的数据将被自动发出。

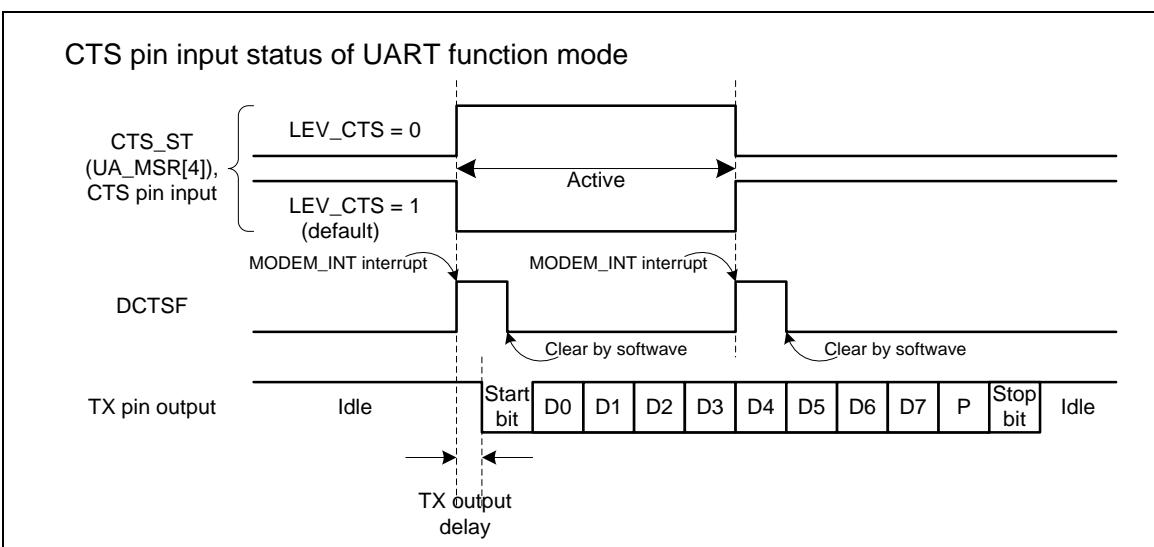


图 6-100 UART CTS 自动流控功能

如下图所示，UART RTS 自动流控模式(AUTO\_RTS\_EN(UA\_IER[12])=1)下，内部nRTS信号由UART FIFO 控制器控制，根据RTS\_RTI\_lev(UA\_FCR[19:16]) 的设定触发电平。

设定 LEV\_RTS(UA\_MCR[9]) 可以控制 RTS 引脚将nRTS信号反转输出还是不反转输出。用户可以读RTS\_ST(UA\_MCR[13]) 位来得到实际的 RTS 引脚输出电平。

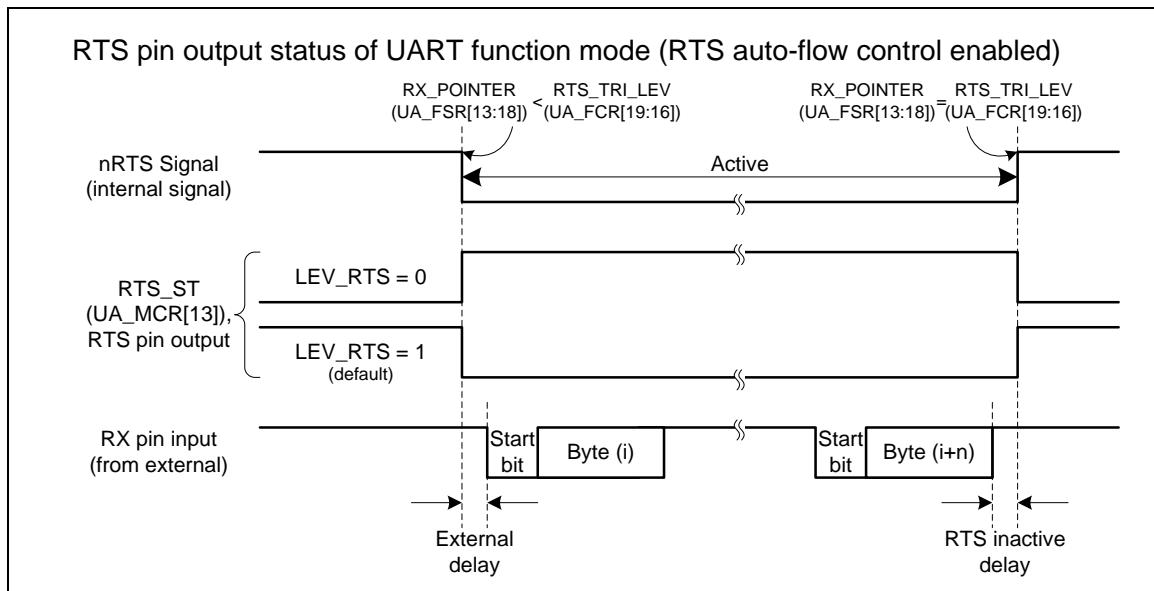


图 6-101 UART RTS 自动流控使能

如下图所示，软件模式下 (AUTO\_RTS\_EN(UA\_IER[12])=0)，软件通过编程RTS(UA\_MCR[1])位，RTS 流控直接由软件控制。

设置 LEV\_RTS(UA\_MCR[9]) 可以控制RTS 引脚将RTS(UA\_MCR[1])位反转输出，还是不反转输出。用户可以读RTS\_ST(UA\_MCR[13]) 位来得到实际的 RTS 引脚输出电平。

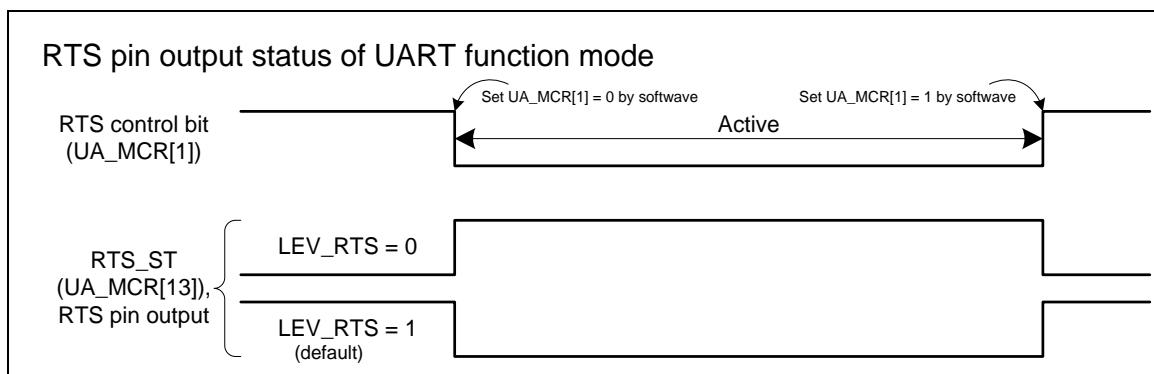


图 6-102 UART RTS 软件流控

#### 6.14.5.6 IrDA 功能模式

UART 控制器也支持 IrDA SIR (串行红外)功能，用户必须设置UA\_FUN\_SEL [1:0] = '10'来使能IrDA功能。SIR规格定义了一个近距离异步串行传输模式，数据格式为1个起始位，8个数据位和1个停止位。最大数据率为115.2 kbps。IrDA SIR框图包含一个 IrDA SIR协议编码/解码。IrDA协议是半双工的，所以不能同时收/发。IrDA SIR物理层说明发送和接收之间最小需要10ms的传输延迟，该延迟特性必须由软件实现。.

IrDA 模式下， UA\_BAUD[DIV\_X\_EN] 位需禁用。

波特率 = Clock / (16 \* BRD), BRD 为UA\_BAUD 寄存器中的波特率分频器.

下图为IrDA 控制框图.

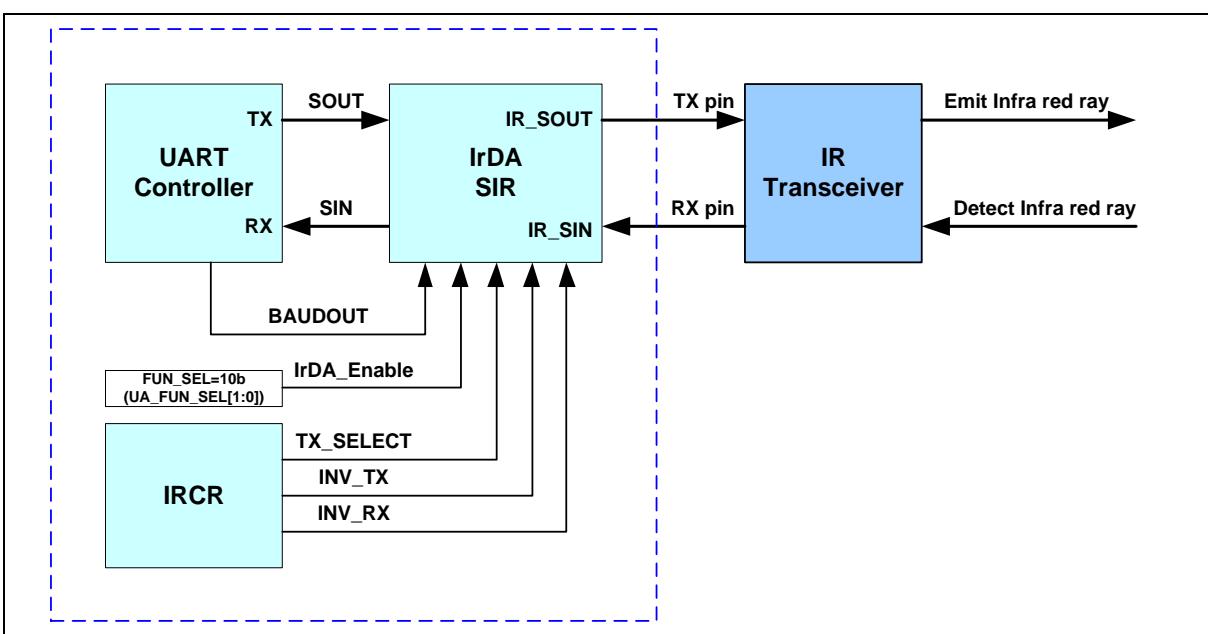


图 6-103 IrDA 框图

#### IrDA SIR发送编码器

IrDA SIR发送编码器以非归零(NRZ) 调制方式从 UART 输出比特流。. IrDA SIR 物理层指定使用归零, 反向 (RZI) 调制方式, 红外光脉冲代表逻辑0。被调制的输出脉冲流被发送到外部输出驱动器和红外光发射二极管

在正常模式下, 传输脉冲的宽度为 3/16 波特率周期.

#### IrDA SIR接收解码器

IrDA SIR 接收解码器以归零(Return-to-Zero)方式解调由输入探测器输入的比特流，并输出 NRZ比特流到 UART 作为数据输入。解码器在空闲模式输入通常为高。 (因为IRCR (INV\_RX [6]) 默认设定为1) 当解码器输入低时，表示一个起始位。

#### IrDA SIR操作

IrDA SIR 编码/解码器提供UART数据流和半双工串行SIR接口间互相转换的功能。. 下图是IrDA编码/解

码器波形图

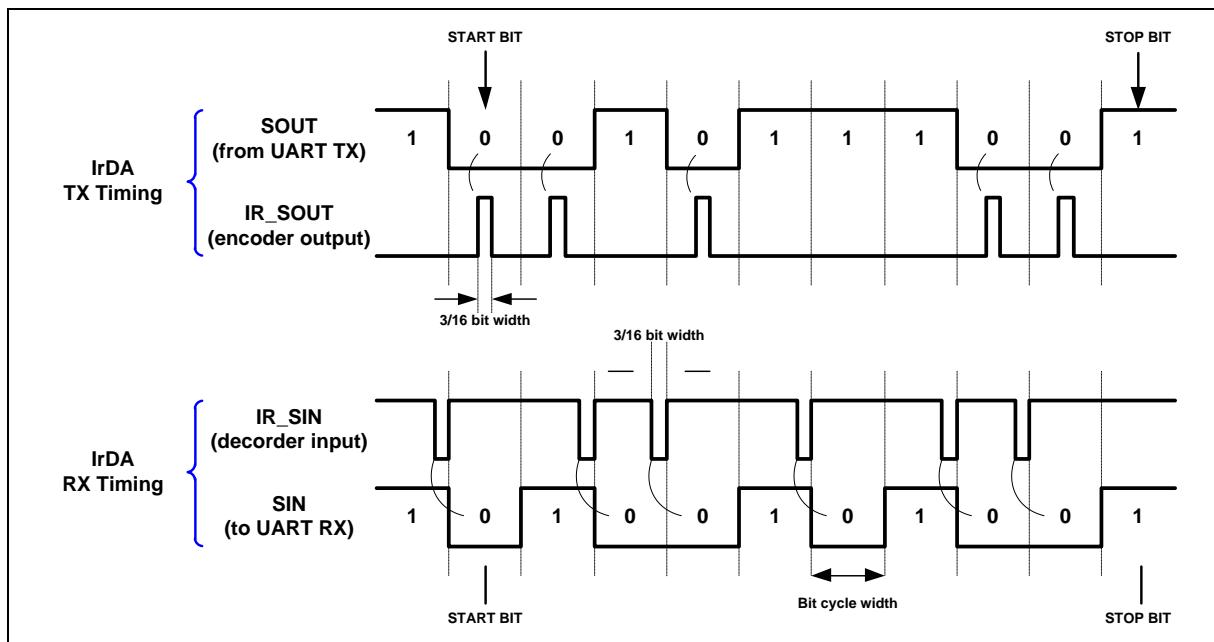


图 6-104 IrDA TX/RX 时序框图

#### 6.14.5.7 LIN (Local Interconnection Network) 模式

UART 控制器支持 LIN 功能，用户必须设定UA\_FUN\_SEL[1:0] 为 '01' 来选择LIN功能。LIN 模式下，根据LIN的标准，每个字节包含1个值为0的起始位 (dominant)，后面是8 个比特 的数据(LSB优先)，最后以1个值为1的停止位 (recessive) 结束。下图是LIN的帧结构：

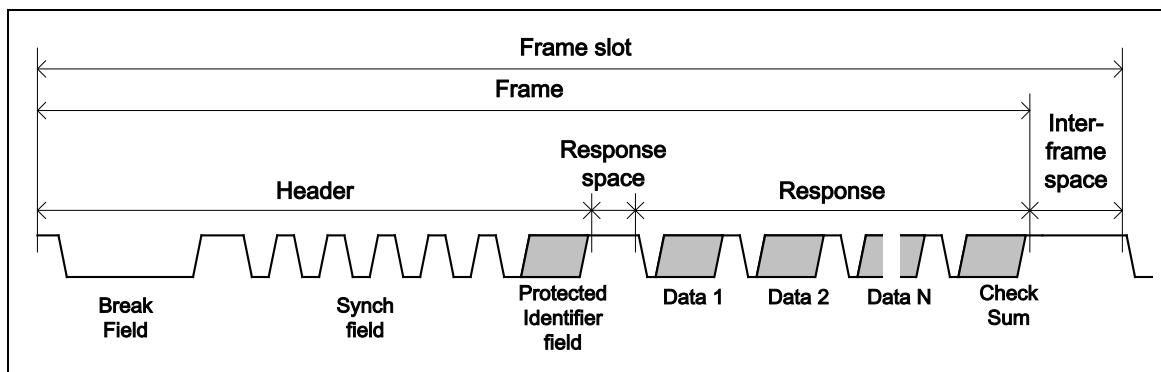


图 6-105 LIN 帧结构

#### LIN总线发送(TX)编程流程如下

1. 设定UA\_FUN\_SEL[1:0] 为 '01' 使能 LIN 总线模式
2. 填UA\_LIN\_BCNT寄存器中的 UA\_LIN\_BKFL 选择break 域的长度. (break 域的长度为 UA\_LIN\_BKFL + 2).
3. 填0x55 到 UA THR寄存器请求发送SYNCH域.

4. 将protected identifier 的值填到UA\_THR 寄存器，请求Identifier 域被发送
5. 设定 UA\_LIN\_BCNT 寄存器的 LIN\_TX\_EN 位开始发送（当break 域操作结束时，LIN\_TX\_EN 将被自动清为0）。
6. 当THR中字节的 STOP 位被发送到总线上时，硬件将设定UA\_FSR中的 TE\_FLAG 为 1.
7. 填 N 个字节数据和Checksum 到UA\_THR，然后重复步骤 5 和 6 发送数据.

#### LIN总线接收(RX)编程流程如下

1. 设定 **UA\_FUN\_SEL[1:0]** 为 '01' 使能LIN 总线模式.
2. 设定UA\_LIN\_BCNT寄存器的LIN\_RX\_EN 来使能 LIN RX 模式.
3. 等待UA\_ISR 中的LIN\_RX\_BREAK\_IF 标志被置，接收到Break 域.
4. 等待UA\_ISR中的 RDA\_IF 标志被置，从 UR\_RBR 寄存器读回收到的数据.

#### 6.14.5.8 RS-485 功能模式

UART 控制器支持RS-485 9位模式，用户可以通过设置UA\_FUN\_SEL [1:0] = '11'来选择RS-485功能。方向由RTS引脚控制，或者通过编程GPIO (RTS0使用P0.3 和 RTS1使用 P0.1)由软件实现控制。RS-485收发器由RTS控制信号控制使能和方向。RS-485模式下，收发的许多特性和UART一样。

RS-485 模式下，控制器可以配置成可寻址的RS-485从机，RS-485主机发送器将通过设置校验位 (第9位)为1来标识一个地址字符。对于数据字符，校验位设置为0. 软件可通过设置寄存器UA\_LCR 控制第9位 (**PBE** , **EPE** 和 **SPE**置位, 第9位发送0 , **PBE** 和 **SPE** 置位, **EPE**清零, 第9位发送1).

该控制器支持三种操作模式：RS-485 普通多点模式(NMM)，RS-485 自动地址识别模式 (AAD) 和RS-485 自动方向控制模式(AUD)。软件可通过编程UA\_ALT\_CSR寄存器选择任何工作模式，通过设置UA\_TOR [DLY] 可以设置上一个停止位与下一个开始位之间的延迟时间。

#### **RS-485 普通多点模式 (NMM)**

RS-485 普通多点模式下，首先软件必须决定在收到地址字节之前数据是否存储到RX FIFO中。如果软件想忽略地址字节检测到之前的所有数据，则流程为设置RX\_DIS (UA\_FCR [8])，然后使能RS485\_NMM (UA\_ALT\_CSR [8])，这样，接收器忽略所有数据直至检测到地址字节 (bit9=1) 并将地址字节数据存储于RX FIFO中。如果软件想接收检测到地址位之前的所有数据，则流程为禁止RX\_DIS (UA\_FCR [8])，然后使能RS485\_NMM (UA\_ALT\_CSR [8])，这样，接收器就会接收所有数据。

如果检测到地址位，RS-485控制器会向CPU产生一个中断，软件可通过设定RX\_DIS (UA\_FCR [8])来决定是否接收后面的数据字节并保存到RX FIFO中。如果关闭RX\_DIS (UA\_FCR [8])使能接收器接收，所有接收的数据都将被接收并存储于RX-FIFO中；如果打开RX\_DIS (UA\_FCR [8])禁止接收数据，则接收到的所有数据都将被忽略直至下一个地址位被检测到。若软件设置RX\_DIS (UA\_FCR [8])关闭接收功能，当检测到下一个地址字节时，控制器会自动清RX\_DIS (UA\_FCR [8])位（就是说，自动打开接收功能），并将地址字节数据存储到RX-FIFO。

#### **RS-485 自动地址识别模式 (AAD)**

RS-485自动地址识别模式下，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，忽略所有数据。然后地址字节数据将被存储在RX-FIFO中。其后所有接收到的字节数据将被接收，并存储于RX-FIFO，直到收到的地址字节不匹配ADDR\_MATCH (UA\_ALT\_CSR[31:24]) 的值为止。

#### **RS-485 自动方向模式 (AUD)**

RS-485控制器的另一个可选的功能是自动方向控制。 用户必须设置RS485\_AUD(UA\_ALT\_CSR[10])等于0来使能RS-485自动方向模式。 使用来自异步串行口的RTS控制信号来控制RS-485驱动器。 RTS线被连接到RS-485驱动器使能引脚上，以便设置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3态状态。 用户通过设置寄存器UA\_MCR 中的LEV\_RTS位改变 RTS 驱动电平。

下图说明AUD模式下，RS-485 RTS驱动电平。当传输数据时RTS引脚将被自动驱动。

设置LEV\_RTS(UA\_MCR[9])可以控制RTS引脚输出驱动电平。用户可以读RTS\_ST(UA\_MCR[13])得到RTS引脚上实际输出的电平。

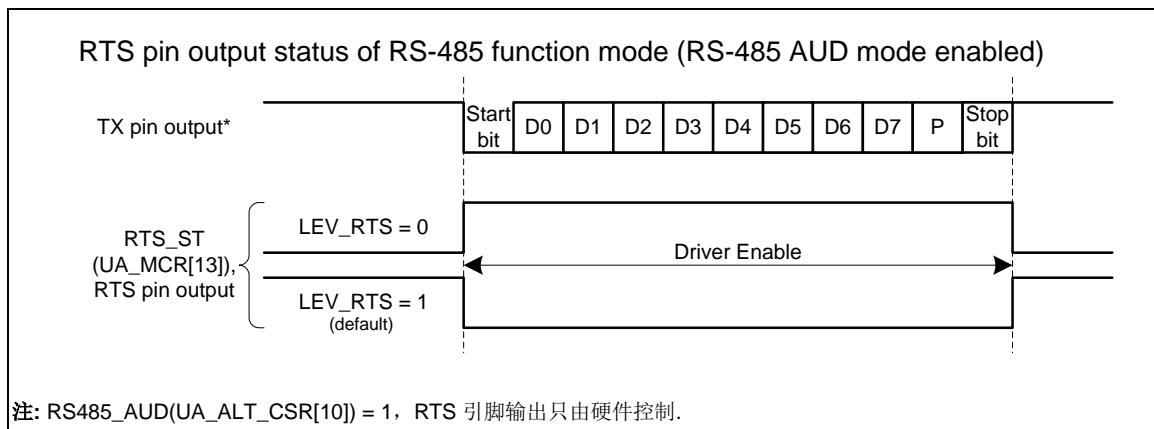


图 6-106 自动方向控制模式下 RS-485 RTS 驱动电平

下图说明软件控制(RS45\_AUD(UA\_ALT\_CSR[10])=0) 模式下，RS-485 RTS 的驱动电平。RTS 驱动电平由RTS(UA\_MCR[1])位控制。

设置 LEV\_RTS(UA\_MCR[9]) 可以控制RTS(UA\_MCR[1])位控制输出的RTS 引脚输出电平反转/不反转。用户可以读RTS\_ST(UA\_MCR[13]) 位得到RTS 引脚实际输出的电平状态。

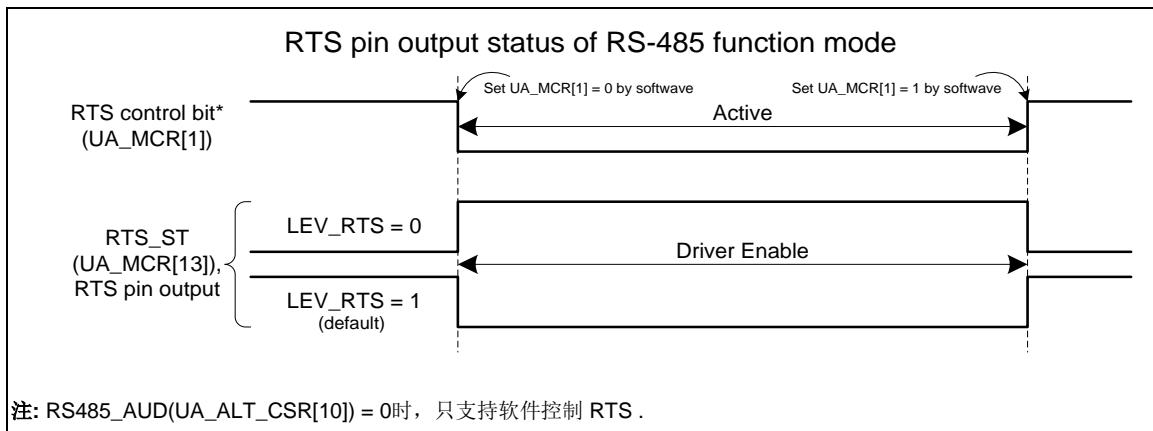


图 6-107 软件控制 RS-485 RTS 驱动电平

编程流程示例:

1. 设置寄存器UA\_FUN\_SEL中的FUN\_SEL位选择RS-485功能.
2. 设置寄存器UA\_FCR 中的RX\_DIS 位使能或禁止RS-485 接收数据
3. 设置RS-485\_NMM 或 RS-485\_AAD 模式.
4. 如果选择RS-485\_AAD 模式, ADDR\_MATCH填入自动地址匹配值.
5. 设置RS45\_AUD选择自动方向控制.

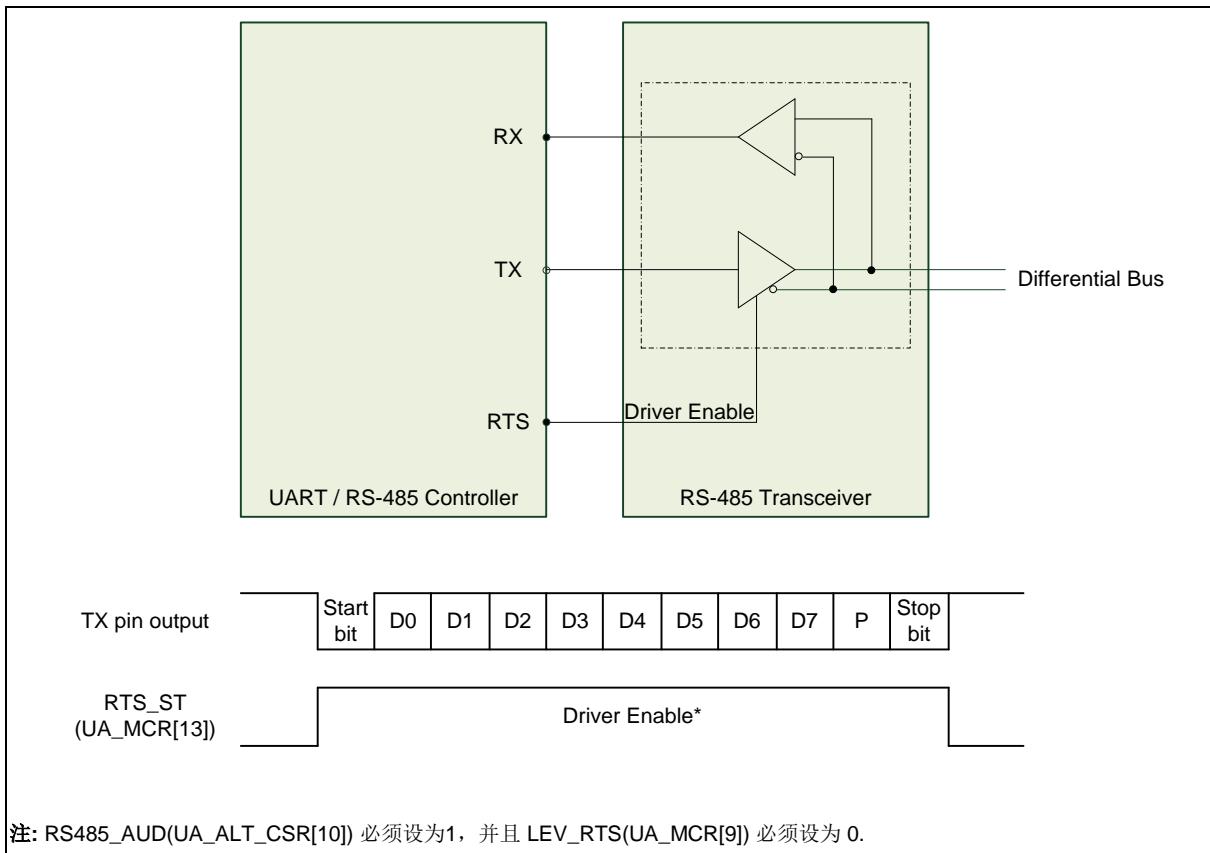


图 6-108 RS-485 帧结构

### 6.14.6 寄存器映射

R:只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>UART 基地址:</b>				
<b>UART0_BA = 0x4005_0000</b>				
<b>UART1_BA = 0x4015_0000</b>				
UA_RBR	UARTx_BA+0x00	R	UART接收缓存寄存器.	Undefined
UA_THR	UARTx_BA+0x00	W	UART发送保持寄存器.	Undefined
UA_IER	UARTx_BA+0x04	R/W	UART中断使能寄存器.	0x0000_0000
UA_FCR	UARTx_BA+0x08	R/W	UART FIFO 控制寄存器.	0x0000_0101
UA_LCR	UARTx_BA+0x0C	R/W	UART 线 控制寄存器.	0x0000_0000
UA_MCR	UARTx_BA+0x10	R/W	UART Modem 控制寄存器.	0x0000_0200
UA_MSR	UARTx_BA+0x14	R/W	UART Modem 状态寄存器.	0x0000_0110
UA_FSR	UARTx_BA+0x18	R/W	UART FIFO 状态寄存器.	0x1040_4000
UA_ISR	UARTx_BA+0x1C	R/W	UART 中断 状态寄存器.	0x0000_0002
UA_TOR	UARTx_BA+0x20	R/W	UART 超时寄存器	0x0000_0000
UA_BAUD	UARTx_BA+0x24	R/W	UART 波特率分频寄存器	0x0F00_0000
UA_IRCR	UARTx_BA+0x28	R/W	UART IrDA 控制寄存器	0x0000_0040
UA_ALT_CSR	UARTx_BA+0x2C	R/W	UART 控制/状态寄存器	0x0000_0000
UA_FUN_SEL	UARTx_BA+0x30	R/W	UART 功能选择寄存器	0x0000_0000

注: x = 0,1

### 6.14.7 寄存器描述

#### UART接收缓冲寄存器(UA\_RBR)

寄存器	偏移量	R/W	描述	复位后的值
UA_RBR	UARTx_BA+0x00	R	UART接收缓冲寄存器	Undefined



Bits	描述	
[31:8]	保留	保留
[7:0]	RBR	接收缓冲寄存器（只读） 通过读此寄存器, UART 将返回一个从 RX引脚接收到的 8-位数据 (LSB优先).

**UART发送保持寄存器(UA\_THR)**

寄存器	偏移量	R/W	描述	复位后的值
UA_THR	UARTx_BA+0x00	W	UART0发送保持寄存器	Undefined



Bits	描述	
[31:8]	保留	保留
[7:0]	THR	<b>发送保持寄存器</b> 通过写该寄存器， UART 将通过TX引脚 (LSB优先) 送出 8-位数据。

**UART中断使能寄存器(UA\_IER)**

寄存器	偏移量	R/W	描述	复位后的值
UA_IER	UARTx_BA+0x04	R/W	UART中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
保留							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
保留		AUTO_CTS_EN	AUTO_RTS_EN	TIME_OUT_EN	保留		LIN_RX_BRK_IEN
7	6	5	4	3	2	1	0
保留	WAKE_EN	BUF_ERR_IEN	RTO_IEN	MODEM_IEN	RLS_IEN	THRE_IEN	RDA_IEN

Bits	描述	
[31:14]	保留	保留
[13]	AUTO_CTS_EN	<p><b>CTS 自动流控制使能</b>            1 = 使能 CTS 自动流控制.            0 = 禁用 CTS 自动流控制.  <b>注:</b> 当 CTS 自动流控制使能, 当CTS输入有效时UART将向外部设备发送数据 (直到CTS有效UART 才会开始机发送数据).</p>
[12]	AUTO_RTS_EN	<p><b>RTS 自动流控制使能</b>            1 = 使能 RTS 自动流控制.            0 = 禁用 RTS自动流控制.  <b>注:</b> 当 RTS 自动流控使能时, 如果RX FIFO中接收的字节数和RTS_TRI_LEVEL (UA_FCR [19:16])相等, UART 将使RTS信号失效</p>
[11]	TIME_OUT_EN	<p><b>超时计数器使能控制</b>            1 = 使能超时计数器.            0 = 禁用超时计数器.</p>
[10:9]	保留	保留
[8]	LIN_RX_BRK_IEN	<p><b>LIN 检测到 Break 域中断使能控制</b>            1 = 使能 Lin 总线接收到break 域中断            0 = 禁止 Lin 总线接收break 域中断  <b>注:</b> 这个域只是给LIN功能使用.</p>
[7]	保留	保留
[6]	WAKE_EN	<b>UART唤醒 CPU 功能使能控制</b> 0 = 禁用 UART 唤醒 CPU 功能

		1 = 使能唤醒功能 注：当系统在深度睡眠模式下，外部 /CTS 信号改变将CPU从深度睡眠模式下唤醒。
[5]	<b>BUF_ERR_IEN</b>	缓冲错误中断使能控制 1 = 使能 BUF_ERR_INT 0 = 禁止 BUF_ERR_INT
[4]	<b>RTO_IEN</b>	接收超时中断使能控制 0 = 禁用TOUT_INT中断 1 = 使能TOUT_INT 中断
[3]	<b>MODEM_IEN</b>	调制解调器状态中断使能控制 0 = 禁用 MODEM_INT中断 1 = 使能MODEM_INT中断
[2]	<b>RLS_IEN</b>	接收线状态中断使能控制 0 = 禁用 RLS_INT中断 1 = 使能RLS_INT中断
[1]	<b>THRE_IEN</b>	发送保持寄存器空中断使能控制 0 = 禁用 THRE_INT中断 1 = 使能THRE_INT中断
[0]	<b>RDA_IEN</b>	接收数据可得中断使能控制 0 = 禁用RDA_INT中断 1 = 使能RDA_INT中断

**UART FIFO 控制寄存器 (UA\_FCR)**

寄存器	偏移量	R/W	描述	复位后的值
UA_FCR	UARTx_BA+0x08	R/W	UART FIFO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				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

Bits	描述											
[31:20]	保留	保留										
[19:16]	RTS_TRILEV	<p>自动流控下RTS触发级别 接收FIFO中的字节数</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>RTS_TRILEV</th> <th>触发级别(Bytes)</th> </tr> </thead> <tbody> <tr> <td>0000</td> <td>01</td> </tr> <tr> <td>0001</td> <td>04</td> </tr> <tr> <td>0010</td> <td>08</td> </tr> <tr> <td>0011</td> <td>14</td> </tr> </tbody> </table> <p>注: 该域用于RTS自动流控制..</p>	RTS_TRILEV	触发级别(Bytes)	0000	01	0001	04	0010	08	0011	14
RTS_TRILEV	触发级别(Bytes)											
0000	01											
0001	04											
0010	08											
0011	14											
[15:9]	保留	保留										
[8]	RX_DIS	<p>关闭接收寄存器. 接收器禁用或使能(置1禁止接收) 1 = 禁止接收 0 = 接收使能 注: 该位用于RS-485 普通多点模式。 必须在设置UA_ALT_CSR [RS-485_NMM]之前被设置好。</p>										
[7:4]	RFITL	<p>接收 FIFO 中断 (RDA_INT)触发级别 当接收FIFO中的字节数与RFITL匹配时， RDA_IF 将被置位 (如果UA_IER [RDA_IEN]使能，将发生中断)。</p> <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>RFITL</th> <th>中断触发级别(Bytes)</th> </tr> </thead> </table>	RFITL	中断触发级别(Bytes)								
RFITL	中断触发级别(Bytes)											

		<table border="1"> <tr><td>0000</td><td>01</td></tr> <tr><td>0001</td><td>04</td></tr> <tr><td>0010</td><td>08</td></tr> <tr><td>0011</td><td>14</td></tr> <tr><td>其它</td><td>预留</td></tr> </table>	0000	01	0001	04	0010	08	0011	14	其它	预留	
0000	01												
0001	04												
0010	08												
0011	14												
其它	预留												
[3]	保留	保留											
[2]	TFR	<p><b>发送域软件复位</b></p> <p>当 TX_RST 置位, 发送FIFO中的所有字节和发送内部状态机都会被清零            0 = 该位写 0 将无效.            1 = 该位置位将复位 TX 内部状态机和指针.  <b>注:</b> 该位自动清零需要至少3个UART时钟周期</p>											
[1]	RFR	<p><b>接收域软件复位</b></p> <p>当 RX_RST 置位, 接收FIFO中所有字节和RX内部状态机都将被清零            0 = 该位写 0 将无效.            1 = 该位置位将复位 RX 内部状态机和指针.  <b>注:</b> 该位自动清零需要至少3个UART时钟周期</p>											
[0]	保留	保留											

**UART 线控寄存器 (UA\_LCR)**

寄存器	偏移量	R/W	描述	复位后的值
UA_LCR	UARTx_BA+0x0C	R/W	UART线控制寄存器	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	BCB	SPE	EPE	PBE	NSB	WLS	

Bits	描述					
[31:7]	保留	保留				
[6]	<b>BCB</b>	<b>Break控制位</b> 该位置位，串行数据输出 (TX) 将被迫处于 Spacing State (逻辑0)。 该位仅作用于 TX 对传输逻辑不起作用。				
[5]	<b>SPE</b>	<b>Stick 校验使能控制</b> 0 = 禁用 stick 校验 1 = 当 PBE(UA_LCR[3])、EPE (UA_LCR[4]) 和 SPE 置位，校验位发送0；当 PBE(UA_LCR[3]) 和 SPE 置位 并且 EPE(UA_LCR[4]) 清除，校验位发送1				
[4]	<b>EPE</b>	<b>偶校验使能控制</b> 0 = 数据位和校验位中共有奇数个逻辑1被传输和检测 1 = 数据位和校验位中共有偶数个逻辑1被传输和检测 该位仅当PBE (UA_LCR[3]) (校验位使能位) 置位才有效..				
[3]	<b>PBE</b>	<b>校验使能位</b> 0 = 当传输时校验位不发送(只发送数据)或检测 (只接收数据). 1 = 在串行数据的最后一位和停止位之间生成 (发送) 和检测 (接收) 校验位				
[2]	<b>NSB</b>	<b>“STOP 位”个数</b> 0= 传递数据时产生 1个停止位 1= 传递数据时产生 1.5个停止位 (5位字长度被选择时); 6, 7-和 8位字长度被选择时，产生2个停止位.				
[1:0]	<b>WLS</b>	<b>字长度选择</b> <table border="1" style="width: 100%;"><tr><td style="width: 50%;">WLS[1:0]</td><td style="width: 50%;">字长度</td></tr><tr><td>00</td><td>5 bits</td></tr></table>	WLS[1:0]	字长度	00	5 bits
WLS[1:0]	字长度					
00	5 bits					

		01	6 bits	
		10	7 bits	
		11	8 bits	

**UART MODEM 控制寄存器 (UA\_MCR)**

寄存器	偏移量	R/W	描述	复位后的值
UA_MCR	UARTx_BA+0x10	R/W	UART调制解调器控制寄存器	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		RTS_ST	保留			LEV_RTS	保留
7	6	5	4	3	2	1	0
保留						RTS	保留

Bits	描述	
[31:14]	保留	保留
[13]	RTS_ST	<b>RTS 引脚的状态(只读)</b> 该位表示 RTS 引脚输出电压的状态。 0 = RTS引脚输出高电平 1 = RTS引脚输出低电平
[12:10]	保留	保留
[9]	LEV_RTS	<b>RTS 引脚有效电平</b> 该位设定 RTS 引脚的有效输出电平。 0= RTS引脚输出低电平有效 1= RTS引脚输出高电平有效 <b>注1:</b> 参考 图6-101 和 图6-102 UART 功能模式 <b>注2:</b> 参考图6-106 和 图6-107 RS-485 功能模式。
[8:2]	保留	保留
[1]	RTS	<b>RTS (Request-To-Send) 信号控制</b> 该位直接控制内部RTS信号是否有效，然后驱动RTS引脚输出LEV_RTS定义的电平 0 = RTS 信号无效 1 = RTS 信号有效。 <b>注1:</b> UART模式下如果RTS自动流控功能使能，该位无效 <b>注2:</b> RS485模式下如果自动方向模式(AUD)使能，该位无效
[0]	保留	保留

UART Modem状态寄存器 (UA\_MSR)

寄存器	偏移量	R/W	描述	复位后的值
UA_MSR	UARTx_BA+0x14	R/W	UART 调制解调器状态寄存器	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

Bits	描述	
[31:9]	保留	保留
[8]	LEV_CTS	<p><b>CTS 引脚有效电平</b>            该位决定 CTS 引脚输入的有效电平。            0 = CTS引脚输入高电平有效            1 = CTS引脚输入低电平有效  <b>注:</b> 请参考图6-100</p>
[7:5]	保留	保留
[4]	CTS_ST	<p><b>CTS 引脚状态(只读)</b>            该位指示了CTS引脚输入电平的状态            0 = CTS引脚输入低电平            1 = CTS引脚输入高电平  <b>注:</b> 当UART时钟使能，并且多功能引脚已经配置为CTS时，该位表示 CTS 引脚的状态.</p>
[3:1]	保留	保留
[0]	DCTSF	<p><b>检测 CTS 状态改变标志位(只读)</b>            只要 CTS 输入状态改变 该位置位，并且在UA_IER[MODEM_IEN]置位时，将发生调制解调器中断            0 = CTS输入状态没有发生改变            1 = CTS输入状态发生改变  <b>注:</b> 该位只读，可写‘1’清除.</p>

**UART FIFO 状态寄存器(UA\_FSR)**

寄存器	偏移量	R/W	描述				复位后的值
UA_FSR	UARTx_BA+0x18	R/W	UART FIFO状态寄存器.				0x1040_4000

31	30	29	28	27	26	25	24
保留		TE_FLAG		保留		TX_OVER_IF	
23	22	21	20	19	18	17	16
TX_FULL	TX_EMPTY	TX_POINTER					
15	14	13	12	11	10	9	8
RX_FULL	RX_EMPTY	RX_POINTER					
7	6	5	4	3	2	1	0
保留	BIF	FEF	PEF	RS485_ADD_DETF	保留		RX_OVER_IF

Bits	描述	
[31:29]	保留	保留
[28]	TE_FLAG	<b>发送器空闲标志位(只读)</b> 当 TX FIFO(UA_THR) 为空并且最后一个字节的停止位被发送之后，，该位由硬件自动置位. 0 = TX FIFO不为空 1 = TX FIFO为空 <b>注:</b> 当 TX FIFO(UA_THR) 不为空或最后一个字节传输未完成时，该位由硬件自动清为0
[27:24]	保留	保留
[24]	TX_OVER_IF	<b>发送缓冲溢出中断标志</b> 如果TX FIFO (UA_THR) 已满， 再写数据到UA_THR 将导致该位被置为1。 0 = TX FIFO没有溢出 1 = TX FIFO溢出 <b>注:</b> 该位写'1' 清0.
[23]	TX_FULL	<b>发送 FIFO 满(只读)</b> 该位表示TX FIFO是否已满. 0 = TX FIFO没有满 1 = TX FIFO已满 <b>注:</b> 当TX FIFO 中数据个数等于16时，该位将被置为1，否则硬件将自动清成0.
[22]	TX_EMPTY	<b>发送FIFO 为空(只读)</b> 该位表示 TX FIFO 是否为空. 0 = TX FIFO不为空 1 = TX FIFO为空 <b>注:</b> 当 TX FIFO的最后一个字节传输到发送移位寄存器时，硬件置位该位。 当写数据到

		THR (TX FIFO非空) 时硬件将自动清除该位.
[21:16]	<b>TX_POINTER</b>	<p><b>TX FIFO指针(只读)</b>            该位表示TX FIFO 缓冲指针。当CPU 写 1个字节到 UA_THR 时，TX_POINTER 增加 1。            当 TX FIFO 传输 1 个字节到发送移位寄存器时，TX_POINTER 减1。            TX_POINTER的最大值是15。当TX FIFO缓冲的使用级别等于16时，TX_FULL设为1并且TX_POINTER将显示0。当TX FIFO中1个字节传输到移位寄存器时，TX_FULL将被清为0，TX_POINTER将显示15</p>
[15]	<b>RX_FULL</b>	<p><b>接收 FIFO 满 (只读)</b>            该位表示RX FIFO 是否已满.            0 = RX FIFO没有满            1 = RX FIFO已满  <b>注:</b> 当RX FIFO中数据个数等于16时，这个bit将被置位，否则硬件自动清为0.</p>
[14]	<b>RX_EMPTY</b>	<p><b>接收FIFO 为空(只读)</b>            该位表示 RX FIFO是否为空.            0 = RX FIFO不为空            1 = RX FIFO为空  <b>注:</b> 当 RX FIFO 中最后一个字节 被CPU读出时， 硬件置位该位。当 UART 接收到新数据时 该位自动清除.</p>
[13:8]	<b>RX_POINTER</b>	<p><b>RX FIFO 指针(只读)</b>            该位表示 RX FIFO 缓冲指针。当UART 从外部设备接收到 1 个字节数据时，RX_POINTER 增加 1.; 当 RX FIFO 被 CPU 读走 1个字节数据时，RX_POINTER 减1。            RX_POINTER的最大值是15。当RX FIFO缓冲的使用级别等于16时，RX_FULL设为1并且RX_POINTER将显示0。当RX FIFO中1个字节被CPU读走时，RX_FULL将被清为0，RX_POINTER将显示15</p>
[7]	保留	保留
[6]	<b>BIF</b>	<p><b>Break中断标志位 (只读)</b>            当接收数据的输入(RX)保持在"Spacing State"(逻辑0)状态的时间大于整个字(即起始位 + 数据位 + 校验位 + 停止位的所有时间)的传输时间，该位置1。            0 = 没有Break中断产生            1 = 产生了Break中断  <b>注1:</b> 该位只读，但可以写1清除为零(M05xxDN/DE)  <b>注2:</b> 该位只读，但可以写1到RFR (UA_FCR [1])清零(M05xxBN/DN/DE)</p>
[5]	<b>FEF</b>	<p><b>帧错误 标志位 (只读)</b>            当接收的字符没有有效的停止位(即检测到跟在最后一个数据位或校验位后面的停止位为逻辑0)时，该位置位。            0 = 没有发生帧错误            1 = 帧错误发生  <b>注1:</b> 该位只读，但可以写1清除为零(M05xxDN/DE)  <b>注2:</b> 该位只读，但可以写1到RFR (UA_FCR [1])清零(M05xxBN/DN/DE)</p>
[4]	<b>PEF</b>	<p><b>奇偶校验错误 标志位(只读)</b>            当接收到的字符的校验位无效时，该位将置位。            0 = 没有发生校验错误            1 = 校验错误发生</p>

		<b>注1:</b> 该位只读, 但可以写1清除为零(M05xxDN/DE) <b>注2:</b> 该位只读, 但可以写1到RFR (UA_FCR [1])清零(M05xxBN/DN/DE)
[3]	<b>RS485_ADD_DET_F</b>	<b>RS-485地址字节检测标志</b> RS-485模式下, 如果RS485_ADD_EN (UA_ALT_CSR[15])为1使能地址检测模式并且接收器检测到地址字节 (bit 9=1), 该位将被设为1。 <b>注1:</b> 该位只用于RS-485 功能模式. <b>注2:</b> 该位写1清除为零.
[2:1]	保留	保留
[0]	<b>RX_OVER_IF</b>	<b>接收溢出中断标志</b> 当RX FIFO 溢出时, 这个位将被置位. 如果接收到的字节数大于RX_FIFO (UA_RBR) 的大小16, 该位 将被置位. 0 = RX FIFO没有溢出 1 = RX FIFO溢出 <b>注:</b> 该位写1清0.

**UART中断状态控制寄存器(UA\_ISR)**

寄存器	偏移量	R/W	描述	复位后的值
UA_ISR	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							
15	14	13	12	11	10	9	8
LIN_RX_BRE_AK_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_RX_BRE_AK_IF	保留	BUF_ERR_IF	TOUT_IF	MODEM_IF	RLS_IF	THRE_IF	RDA_IF

Bits	描述	
[31:16]	保留	保留
[15]	LIN_RX_BREAK_INT	<b>LIN 总线接收到Break 域中断标志(只读)</b> 如果 LIN_RX_BRK_IEN 和 LIN_RX_BREAK_IF 都被设为1，该位将被设为1. 1 = 发生了LIN 接收到Break 中断 0 = LIN接收到Break 中断没有发生
[14]	保留	保留
[13]	BUF_ERR_INT	<b>缓冲错误中断标志 (只读)</b> 如果 BUF_ERR_IEN 和 BUF_ERR_IF 都被设为 1，该位将被设为1. 1 = 发生了缓冲错误中断 0 = 缓冲错误中断没有发生
[12]	TOUT_INT	<b>超时中断标志(只读)</b> 如果RTO_IEN 和 TOUT_IF都被设为1，该位将被设为1 1 = 发生了超时中断 0 = 超时中断没有发生
[11]	MODEM_INT	<b>调制解调器状态中断标志(只读).</b> 如果MODEM_IEN 和 MODEM_IF都被设为1，该位将被设为1 1 = 发生了Modem中断 0 = Modem中断没有发生
[10]	RLS_INT	<b>接收线状态中断标志(只读).</b> 如果RLS_IEN 和 RLS_IF都被设为1，该位将被设为1 1 = 发生了RLS中断

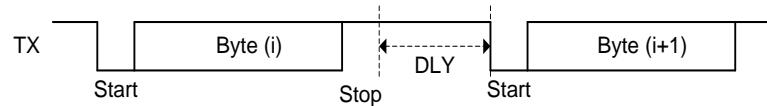
		0 = RLS中断没有发生
[9]	<b>THRE_INT</b>	<p><b>发送保持寄存器空中断标志 (只读).</b> 如果THRE_IEN 和 THRE_IF都被设为1, 该位将被设为1 1 = 发生了THRE中断 0 = THRE中断没有发生</p>
[8]	<b>RDA_INT</b>	<p><b>接收数据可得中断标志 (只读).</b> 如果RDA_IEN 和 RDA_IF都被设为1, 该位将被设为1 1 = 发生了RDA中断 0 = RDA中断没有发生</p>
[7]	<b>LIN_RX_BREAK_IF</b>	<p><b>LIN 总线接收到Break 域标志 (只读)</b> 当 RX 接收到 LIN Break 域时, 该位将被置位。 如果LIN_RX_BRK_IEN (UA_IER [8]) 被使能, LIN 接收到 Break 域中断将发生. 1 = 发生了LIN 接收到Break域中断 0 = LIN 接收到Break域中断没有发生 <b>注意:</b>该位写"1"清0.</p>
[6]	保留	保留
[5]	<b>BUF_ERR_IF</b>	<p><b>缓冲错误中断标志 (只读)</b> 当 TX 或者 RX FIFO缓冲溢出(TX_OVER_IF or RX_OVER_IF)标志被设时, 该位被设。 M05xxBN 中, Break 中断标志, 或者校验错误标志, 或者帧错误标志(BIF 或者 PEF 或者 FEF ) 被置位时, 该位也将被置位。 当 BUF_ERR_IF 被置位时, 代表传输不正确。如果UA_IER [BUF_ERR_IEN] 被使能, 缓冲错误中断将发生。 1 = 发生了缓冲错误中断 0 = 缓冲错误中断没有发生 <b>注1:</b> M05xxDN/DE中, 该位是只读的。当TX_OVER_IF 和 RX_OVER_IF都被清为0时, 该位才会被清为0 <b>注2:</b> M05xxBN中, 该位是只读的。当TX_OVER_IF 、 RX_OVER_IF、 BIF、 FEF和PEF都被清为0时, 该位才会被清为0</p>
[4]	<b>TOUT_IF</b>	<p><b>超时中断标志 (只读)</b> 当 RX FIFO非空且超时计数器计数到TOIC 寄存器的值没有再收到数据, 该位置位。 若RTO_IEN (UA_IER [4])使能, 超时中断将产生。 1 = 发生了超时中断 0 = 超时中断没有发生 <b>注:</b> 该位只读, 但是用户可以读UA_RBR (数据接收寄存器)来将该位清0.</p>
[3]	<b>MODEM_IF</b>	<p><b>调制解调器中断标志 (只读)</b> 当 CTS引脚状态(DCTS=1)改变 该位置位。若UA_IER [MODEM_IEN]使能, 调制解调器中断产生。 1 = 发生了Modem中断 0 = Modem中断没有发生 <b>注:</b> 该位是只读的, 当DCTS位写1清0时, 该位被清为0.</p>
[2]	<b>RLS_IF</b>	<p><b>接收线 中断标志 (只读).</b> 当 RX 接收数据有奇偶校验错误、帧错误、Break错误时 ,该位置位, (至少3个bit中的一个, BIF, FEF 和 PEF, 置位)。若RLS_IEN (UA_IER [2]) 使能, RLS 中断将产生.</p>

		<p>1 = 发生了RLS中断 0 = RLS中断没有发生</p> <p><b>注1:</b> 在RS-485功能模式下，“接收器检测到地址字节(第9位为1)“该位也会被置。同时RS485_ADD_DETF (UA_FSR[3])也会被置</p> <p><b>注2:</b> 该位是只读的，当所有位：BIF, FEF 和 PEF都被清为0时，该位将被清为0.</p>
[1]	<b>THRE_IF</b>	<p><b>发送保持寄存器空中断标志(只读).</b> 当TX FIFO 的最后一个数据发送到发送器移位寄存器，该位置位。如果THRE_IEN (UA_IER [1])使能， THRE中断将发生.</p> <p>1 = 发生了THRE中断 0 = THRE中断没有发生</p> <p><b>注:</b> 该位只读，写数据到THR 清零该位 (TX FIFO 非空).</p>
[0]	<b>RDA_IF</b>	<p><b>接收数据可得中断标志(只读).</b> 当RX FIFO 中的字节数等于RFITL ， RDA_IF 置位。如果使能RDA_IEN (UA_IER [0]), RDA 中断将产生.</p> <p>1 = 发生了RDA中断 0 = RDA中断没有发生</p> <p><b>注:</b> 该位只读，当RX FIFO 中未读取字节数少于阀值(RFITL)时该位清零</p>

UART超时寄存器 (UA\_TOR)

寄存器	偏移量	R/W	描述	复位后的值
UA_TOR	UARTx_BA + 0x20	R/W	UART 超时寄存器	0x0000_0000



Bits	描述	
[31:16]	保留	保留
[15:8]	DLY	<p><b>TX 延迟时间值</b> 该位用于编程上一停止位与下一开始位之间的延迟时间.</p> 
[7:0]	TOIC	<p><b>超时比较器</b> 当 RX FIFO 接收到新数据后超时计数器复位并开始计数(计数时钟频率=波特率). 一旦超时计数器 (TOUT_CNT) 的值和超时比较器 (TOIC) 相等, 且 RTO_IEN (UA_IER [4]) 使能, 接收超时中断将产生 (TOUT_INT). . 一个新的输入数据或 RX FIFO 为空将清除 TOUT_INT. 为了避免收到一个字节马上发生超时中断, TOIC 的值应该设为 40 到 255 之间. 例如: 如果 TOIC 设为 40, 没有校验位并且 1 个停止位的情况下, 四个字节接收时间之内没有收到一个字节, 超时中断发生</p>

**UART波特率分频寄存器(UA\_BAUD)**

寄存器	偏移量	R/W	描述	复位后的值
UA_BAUD	UARTx_BA+0x24	R/W	UART波特率分频寄存器	0x0F00_0000



Bits	描述	
[31:30]	保留	保留
[29]	DIV_X_EN	<p>分频器X使能控制 BRD = 波特率分频值， 波特率方程如下： 波特率= <math>\text{Clock} / [ M * (\text{BRD} + 2) ]</math>； M默认为 16.</p> <p>0 = 禁用分频器 X (M = 16) 1 = 使能分频器 X (M = X+1, 但是 DIVIDER_X[27:24] 必须 <math>\geq 8</math>). 注1: 详见6.14.5.1节 注2: 在IrDA 模式下 该位禁用.</p>
[28]	DIV_X_ONE	<p>分频系数X等于1 0 = 分频系数M = X+1 (M = X+1, 但DIVIDER_X[27:24]必须<math>\geq 8</math>) 1 = 分频系数M = 1 (M = 1, 但BRD [15:0] 必须<math>\geq 3</math>). 注: 详见6.14.5.1节.</p>
[27:24]	DIVIDER_X	分频器 X 波特率分频: M = X+1.
[23:16]	保留	保留
[15:0]	BRD	波特率分频器 这些位表示波特率分频值

**UART IrDA 控制器寄存器 (IRCR)**

寄存器	偏移量	R/W	描述	复位后的值
UA_IRCR	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

Bits	描述	
[31:7]	保留	保留
[6]	INV_RX	接收输入反转控制 1= RX输入信号反转 0= 无反转
[5]	INV_TX	发送输出反转控制 1= TX 输出信号反转 0=无反转
[4:2]	保留	保留
[1]	TX_SELECT	IrDA接收使能控制 1= 使能IrDA 发送器 0 = 使能 IrDA 接收器
[0]	保留	保留

**UART 控制/状态寄存器 (UA\_ALT\_CSR)**

寄存器	偏移量	R/W	描述	复位后的值
UA_ALT_CSR	UARTx_BA+0x2C	R/W	UART 控制/状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
ADDR_MATCH							
23	22	21	20	19	18	17	16
保留							
15	14	13	12	11	10	9	8
RS485_ADD_EN	保留				RS485_AUD	RS485_AAD	RS485_NMM
7	6	5	4	3	2	1	0
LIN_TX_EN	LIN_RX_EN	保留		UA_LIN_BKFL			

Bits	描述
[31:24]	<b>ADDR_MATCH</b> 地址匹配值寄存器 该域包含RS-485 的地址. 注: 该域用于RS-485自动地址识别模式.
[23:16]	保留
[15]	<b>RS485_ADD_EN</b> <b>RS485 地址识别使能使能</b> 该位用于使能RS485地址识别模式. 1 = 使能RS485地址识别模式 0 = 禁用RS485地址识别模式 注: 该域用于RS-485的所有模式.
[14:11]	保留
[10]	<b>RS485_AUD</b> <b>RS485 自动方向控制模式 (AUD)</b> 1 = 使能RS-485 自动方向控制模式 (AUO) 0 = 禁用 RS-485 自动方向控制模式(AUO) 注: RS485_AAD 或 RS485_NMM 操作模式下有效.
[9]	<b>RS485_AAD</b> <b>RS485 自动地址识别操作模式 (AAD)</b> 1 = 使能RS-485 自动地址识别操作模式(AAD) 0 = 禁用 RS-485 自动地址识别操作模式(AAD) 注: RS-485_NMM 操作模式下不要使能该位.
[8]	<b>RS-485_NMM</b> <b>RS-485 普通多点操作模式 (NMM)</b> 1 = 使能 RS-485 普通操作模式 (NMM) 0 = 禁用 RS-485 普通操作模式 (NMM)

		注: RS-485_AAD 操作模式下不要使能该位.
[7]	<b>LIN_TX_EN</b>	<p><b>LIN 使能发送 Break 域</b>            1 = 使能 LIN 发送 Break 域.            0 = 禁止 LIN 发送 Break 域.            注: 当发送 break 域完成时, 该位 将被自动清0.</p>
[6]	<b>LIN_RX_EN</b>	<p><b>LIN 接收使能</b>            1 = 使能 LIN 接收模式.            0 = 禁止 LIN 接收模式.</p>
[5:4]	保留	保留
[3:0]	<b>UA_LIN_BKFL</b>	<p><b>UART LIN Break 域长度</b>            这个4-bit的域用来设定LIN 发送 break 域的长度            注: break 域的长度为 UA_LIN_BKFL + 2</p>

**UART 功能选择寄存器 (UA\_FUN\_SEL)**

寄存器	偏移量	R/W	描述	复位后的值
UA_FUN_SEL	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						<b>FUN_SEL</b>	

Bits	描述	
[31:2]	保留	保留
[1:0]	<b>FUN_SEL</b>	<p>功能选择 UART控制器功能模式选择 00 = UART功能模式 01 = LIN功能模式 10 = IrDA功能模式 11 = RS-485功能模式</p>



## 6.15 看门狗定时器 (WDT)

### 6.15.1 概述

看门狗定时器用于在软件运行至未知状态时执行系统复位功能。可以防止系统无限制地挂机，除此之外，看门狗定时器还可将CPU由空闲/掉电模式唤醒。

### 6.15.2 特性

- 18-位自由运行的计数器用于看门狗超时间隔。
- 超时间隔可选( $2^4 \sim 2^{18}$ ) WDT\_CLK周期，超时时间范围在104 ms ~ 26.3168 s (如果WDT\_CLK = 10 kHz)。
- 系统维持在复位状态时间 =  $(1 / \text{WDT\_CLK}) * 63$ .
- 支持看门狗复位延迟
  - ◆ M05xxBN中固定的复位延迟时间为  $1024 * \text{WDT\_CLK}$
  - ◆ M05xxDN/DE中复位延迟时间可选 3/18/130/1026 \* WDT\_CLK
- 当CWDTCEN (CONFIG[31] 看门狗使能位) 位等于0时，支持上电使能看门狗. (M05xxDN/DE )
- 如果看门狗时钟源选择10 kHz，支持看门狗超时唤醒功能

### 6.15.3 WDT 框图

看门狗定时器时钟控制和框图如下所示。

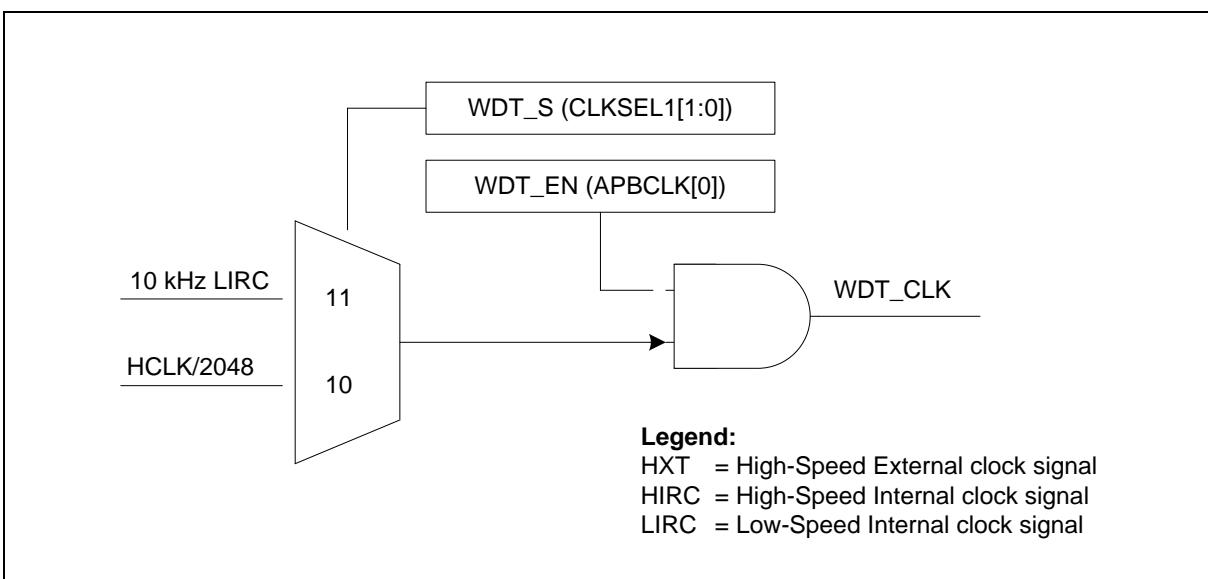


图 6-109 看门狗定时器时钟控制

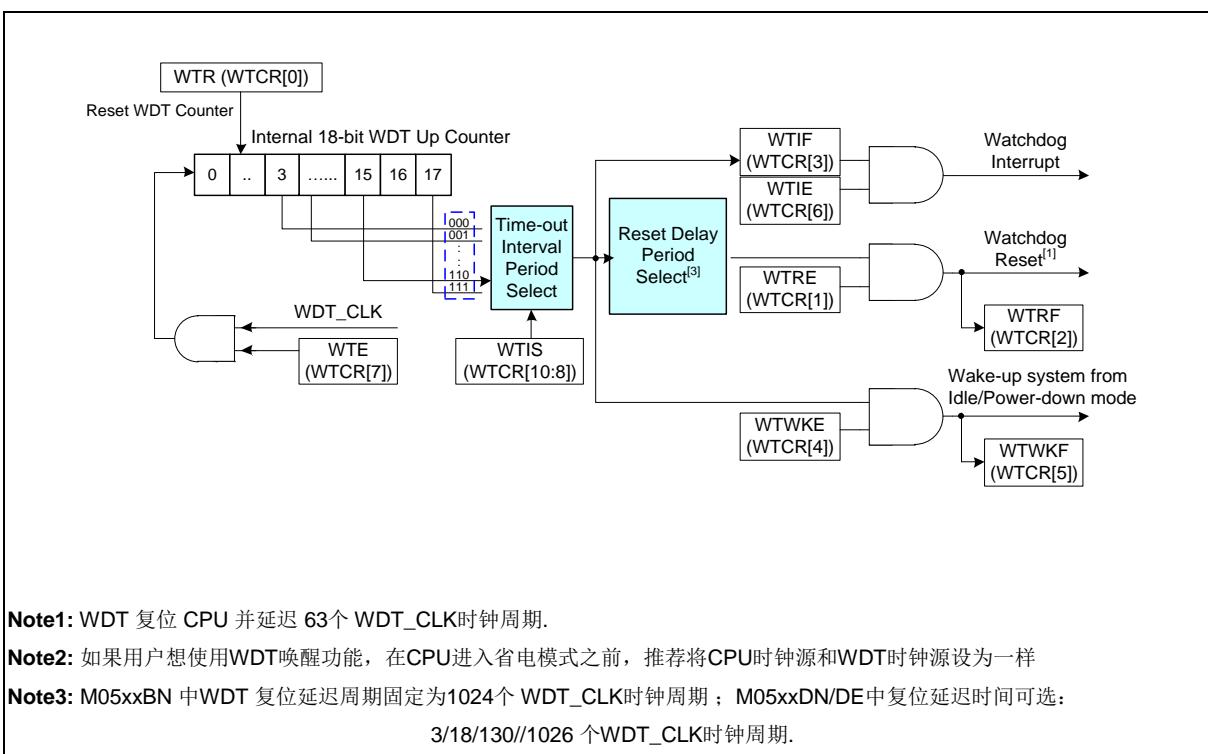


图 6-110 看门狗定时器框图

#### 6.15.4 基本配置

WDT 外设时钟在APBCLK[0]中使能，时钟源在CLKSEL1[1:0]中选择。或者M05xxDN/DE中，用户可以设定CONFIG[31]为0，迫使看门狗上电或者复位之后使能，并迫使使用10KHz作为时钟源。

#### 6.15.5 功能描述

看门狗定时器包含一个18位的自由运行的上数计数器，超时间隔可编程。表 6-21显示了WDT超时间隔周期选择，图6-111显示了WDT超时间隔和复位周期时序。

##### WDT超时中断

设置WTE(WDTCR[7])为1使能看门狗定时器，WDT计数器开始上数。通过设定WTIS有8种超时间隔周期可选。当上数计数器的值达到WTIS的设定值，如果看门狗定时器中断使能位WTIE置位，看门狗定时器中断标志WTIF被立即置位，并发生WDT中断。

##### WDT复位延迟周期和复位系统

跟随在时间溢出事件WTIF之后有一个指定延时T<sub>RSTD</sub> (例如：M05xxBN中是1024 \* T<sub>WDT</sub>个时钟周期)。用户必须在该延时时间结束前设置WTR(WDTCR[0]) (看门狗定时器复位)为高，复位18位WDT计数器，防止CPU复位。如果在指定T<sub>RSTD</sub>延迟时间超时后，WDT计数器没有被清零，看门狗定时器将置位看门狗定时器复位标志WTRF(如果WTRE使能)并使CPU复位。参考图6-111，T<sub>RST</sub> 复位周期将维持63个WDT时钟周期，然后WDT超时复位芯片，用户可以检查WTRF标志来识别系统是否由WDT导致复位。

##### WDT唤醒

如果WDT时钟源选择10kHz，当WDT超时中断发生并且WTWKE使能时，系统可以从掉电模式唤醒。同时WTWKF标志将被设为1，用户可以检查WTWKF标志来检查系统是否由WDT超时中断唤醒。

WTIS	Time-Out Interval Period T <sub>TIS</sub>	Reset Delay Period (M05xxBN) T <sub>RSTD</sub>	Reset Delay Period (M05xxDN/DE) T <sub>RSTD</sub>
000	$2^4 * T_{WDT}$	$1024 * T_{WDT}$	$(3/18/130/1026) * T_{WDT}$
001	$2^6 * T_{WDT}$	$1024 * T_{WDT}$	$(3/18/130/1026) * T_{WDT}$
010	$2^8 * T_{WDT}$	$1024 * T_{WDT}$	$(3/18/130/1026) * T_{WDT}$
011	$2^{10} * T_{WDT}$	$1024 * T_{WDT}$	$(3/18/130/1026) * T_{WDT}$
100	$2^{12} * T_{WDT}$	$1024 * T_{WDT}$	$(3/18/130/1026) * T_{WDT}$
101	$2^{14} * T_{WDT}$	$1024 * T_{WDT}$	$(3/18/130/1026) * T_{WDT}$
110	$2^{16} * T_{WDT}$	$1024 * T_{WDT}$	$(3/18/130/1026) * T_{WDT}$
111	$2^{18} * T_{WDT}$	$1024 * T_{WDT}$	$(3/18/130/1026) * T_{WDT}$

表6-22 看门狗超时间隔周期选择

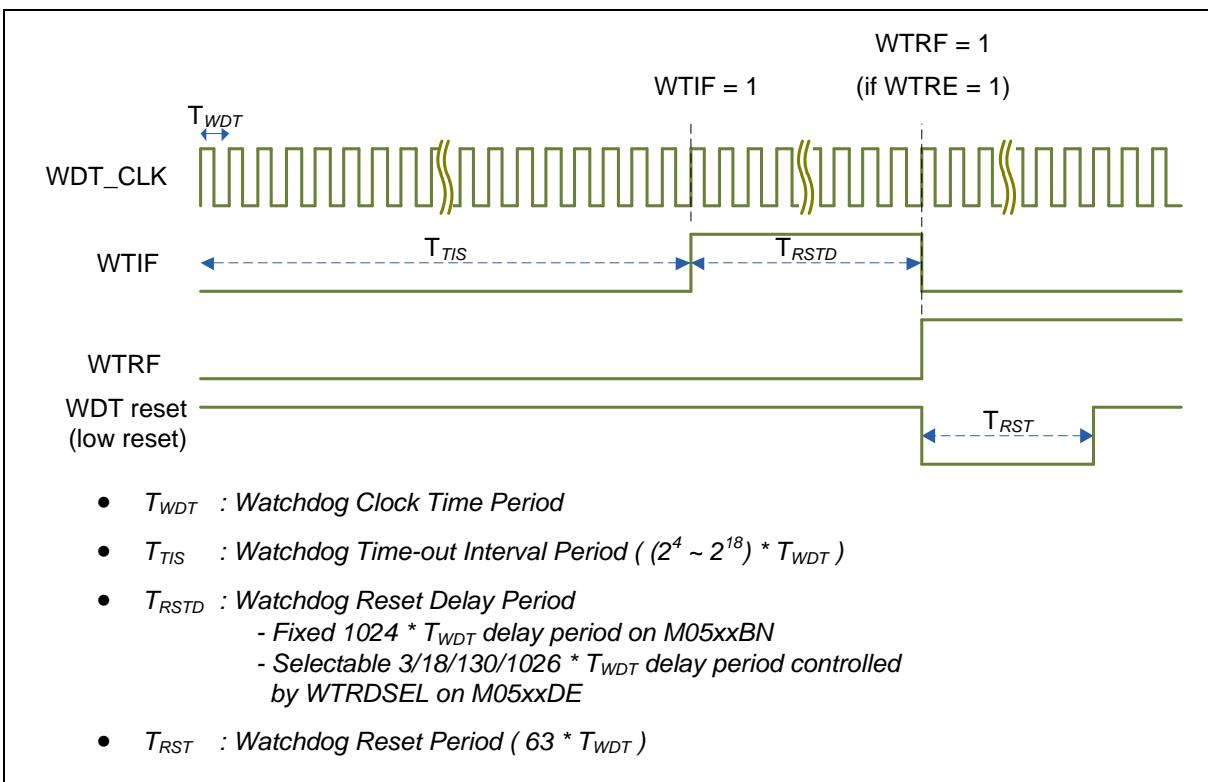


图 6-111 看门狗超时中断间隔与复位周期时序



### 6.15.6 寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>WDT基地址:</b>				
<b>WDT_BA = 0x4000_4000</b>				
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	看门狗定时器控制寄存器2(M05xxDN/DE)	0x0000_0000

### 6.15.7 寄存器描述

#### 看门狗定时器控制寄存器 (WTCR)

寄存器	偏移量	R/W	描述	复位后的值
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

Bits	描述	
[31]	DBGACK_WDT	<b>ICE调试模式应答禁止 (写保护)</b> 0 = ICE 调试模式应答影响看门狗定时器计数。 当ICE调试模式应答时，看门狗定时器计数器的值将维持不变。 1 = ICE 调试模式应答禁止。 无论CPU是否处于ICE调试模式，看门狗定时器计数器将继续正常计数。
[30:11]	保留	保留
[10:8]	WTIS	<b>看门狗定时器超时间隔选择 (写保护)</b> 选择看门狗定时器的超时间隔。 $000 = 2^4 * T_{WDT}$ . $001 = 2^6 * T_{WDT}$ . $010 = 2^8 * T_{WDT}$ . $011 = 2^{10} * T_{WDT}$ . $100 = 2^{12} * T_{WDT}$ . $101 = 2^{14} * T_{WDT}$ . $110 = 2^{16} * T_{WDT}$ . $111 = 2^{18} * T_{WDT}$ .
[7]	WTE	<b>看门狗定时器使能控制(写保护)</b> 0 = 禁用看门狗定时器功能(该动作复位内部计数器) 1 = 使能看门狗定时器
[6]	WTIE	<b>看门狗定时器超时中断使能控制(写保护)</b> 如果该位使能，WDT超时中断将产生。 0 = 禁用看门狗超时中断

		1 = 使能看门狗超时中断
[5]	<b>WTWKF</b>	<p><b>看门狗定时器超时唤醒标志</b>            如果看门狗定时器引起CPU从掉电模式下唤醒，该位将被置高。            0 = 看门狗没有导致CPU唤醒。            1 = CPU 由休眠或掉电模式下被看门狗唤醒  <b>注：</b>该位写1清除为零。</p>
[4]	<b>WTWKE</b>	<p><b>看门狗定时器超时唤醒功能控制(写保护)</b>            如果该位使能，当WTIF被置为1并且WTIE使能，WDT超时中断信号将导致系统唤醒            0 = 禁用看门狗唤醒CPU功能。            1 = 使能看门狗唤醒CPU功能。  <b>注：</b>如果WDT时钟源选择10kHz，芯片可以由WDT超时中断唤醒</p>
[3]	<b>WTIF</b>	<p><b>看门狗定时器超时中断标志</b>            如果看门狗定时器中断使能，该位由硬件置位表示看门狗定时器中断已发生。            0= 没有发生看门狗定时器中断            1= 发生看门狗定时器中断  <b>注：</b>该位写1清除为零。</p>
[2]	<b>WTRF</b>	<p><b>看门狗定时器超时复位标志</b>            该位指示是否看门狗定时器超时导致系统复位。            0 = 看门狗超时没有导致复位。            1 = 看门狗超时引发复位  <b>注：</b>该位写1清除为零。</p>
[1]	<b>WTRE</b>	<p><b>看门狗定时器超时复位使能(写保护)</b>            设定该位使能看门狗定时器复位功能。例如：M05xxBN中，如果WDT超时发生，  <math>(1024 * T_{WDT})</math>时间内没有复位WDT计数器，复位将发生            0 = 禁用看门狗定时器复位功能            1 = 使能看门狗定时器复位功能</p>
[0]	<b>WTR</b>	<p><b>复位看门狗定时器的计数器（写保护）</b>            0 = 写0无效            1 = 复位看门狗定时器内部18位上数计数器的值  <b>注：</b>该位由硬件自动清0。</p>

看门狗定时器控制寄存器2 (WTCRALT)

寄存器	偏移量	R/W	描述	复位后的值
WTCRALT	WDT_BA+0x04	R/W	看门狗定时器控制寄存器2(M05xxDN/DE)	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						<b>WTRDSEL</b>	

Bits	描述	
[31:2]	保留	保留.
[1:0]	<b>WTRDSEL</b>	<p><b>看门狗定时器复位延迟选择 (写保护)</b></p> <p>当WDT 超时发生，软件需要在有一个命名为WDT 复位延迟周期的时间内复位WDT计数器，避免WDT超时复位发生。M05xxDN/DE中软件可以选择一个合适的WDT复位延迟周期。</p> <p>00 = 看门狗复位延迟周期为 <math>1026 * WDT\_CLK</math>.      01 = 看门狗复位延迟周期为 <math>130 * WDT\_CLK</math>.      10 = 看门狗复位延迟周期为 <math>18 * WDT\_CLK</math>.      11 = 看门狗复位延迟周期为 <math>3 * WDT\_CLK</math>.</p> <p>注：如果WDT超时复位发生，该寄存器将被复位成0..</p>

## 6.16 窗看门狗 (WWDT) (M05xxDN/DE )

### 6.16.1 预览

窗看门狗定时器用来在一个指定的窗周期中实现系统复位，避免软件无限期进入不可控状态。

### 6.16.2 特性

- 6-bit 下数计数器当前值 (WWDTCVAL) 和 6-bit 比较窗口值 (WINCMP) 使 WWDT 超时窗周期更有弹性
- 支持 4-bit 值，编程WWDT计数器最大11-bit 预分频计数器周期

### 6.16.3 框图

窗看门狗定时器时钟控制和框图如下所示。

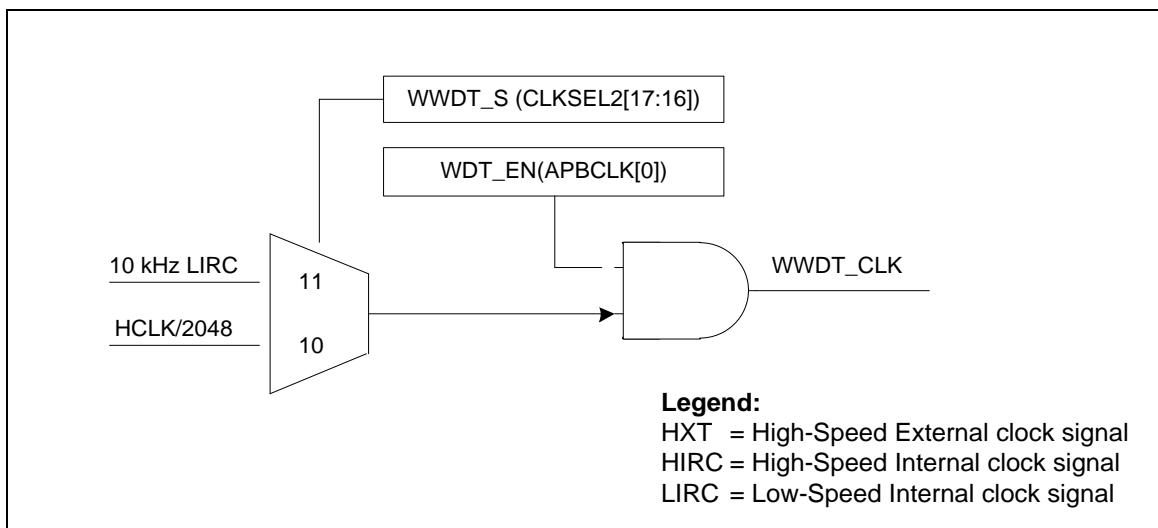


图 6-112 窗看门狗定时器时钟控制

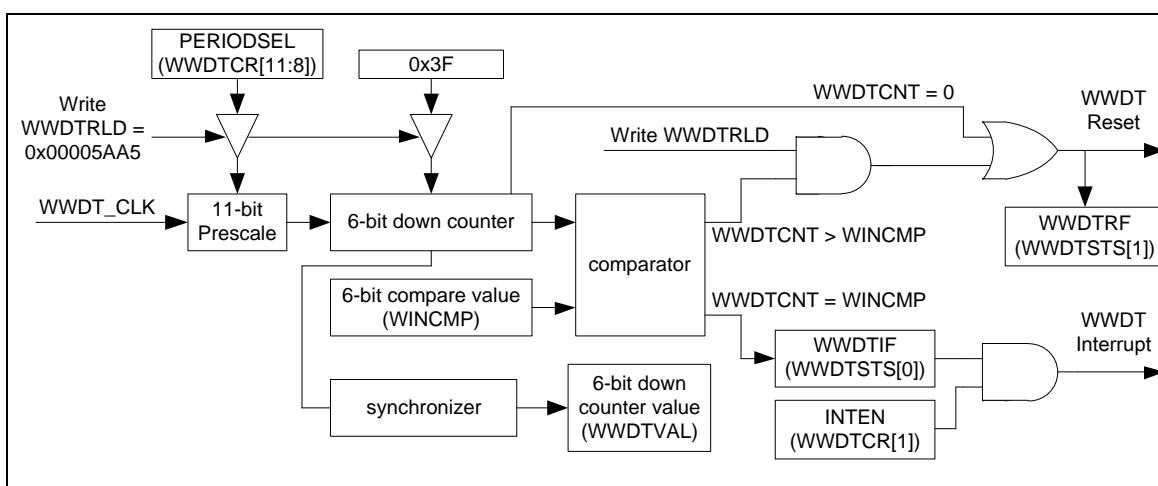


图 6-113 窗看门狗定时器框图

#### 6.16.4 基本配置

WWDT 外设时钟在 APBCLK[0] 中使能，时钟源在 CLKSEL2[17:16]中选择。

#### 6.16.5 功能描述

窗看门狗定时器(WWDT) 包含一个 6-bit 下数计数器，预分频值可编程定义不同的WWDT超时间隔。6-bit WWDT的时钟源为系统时钟除以 2048 (HCLK/2048) 或者内部 10 kHz 振荡器，支持可编程的11-bit预分频计数器，由PERIODSEL 的值控制。PERIODSEL 和预分频值的关系如下表所示。

PERIODSEL	Prescale Value	最大超时间隔	最大超时间隔 (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 6-23 窗看门狗定时器预分频值选择

#### WWDT 计数

当 WWDTEN 使能，WWDT 下数计数器开始从0x3F 到 0计数。为了避免程序无意间关闭WWDT计数器，在芯片上电或者复位之后，WWDT 控制寄存器WWDTCR 只能被写一次。WWDTEN位被软件使能后，除非复位，否则用户不能关闭WWDT 计数器 (WWDTEN)，改变计数器预分频值

(PERIODSEL) 或者改变窗比较值(WINCMR)。

### WWDT 比较匹配中断

中断源共享WDT，WWDT 计数器下数时，当WWDT计数器当前值等于WINCMR的值时，WWDTIF 被置为1。如果WWDTIE被使能，WWDT 比较匹配中断将发生。WWDTIF 可以被软件清除。

### WWDT 复位系统

当WWDTIF被置为1，用户必须写0x00005AA5到WWDTRLD重载WWDT 内部计数器的值为0x3F，同时避免 WWDT 计数器的值到达 0 产生 WWDT 复位系统信号导致系统复位。如果当前 WWDTVAL 的值大于WINCMR 的值并且用户写 0x00005AA5 到 WWDTRLD 寄存器，WWDT 将立即复位系统。

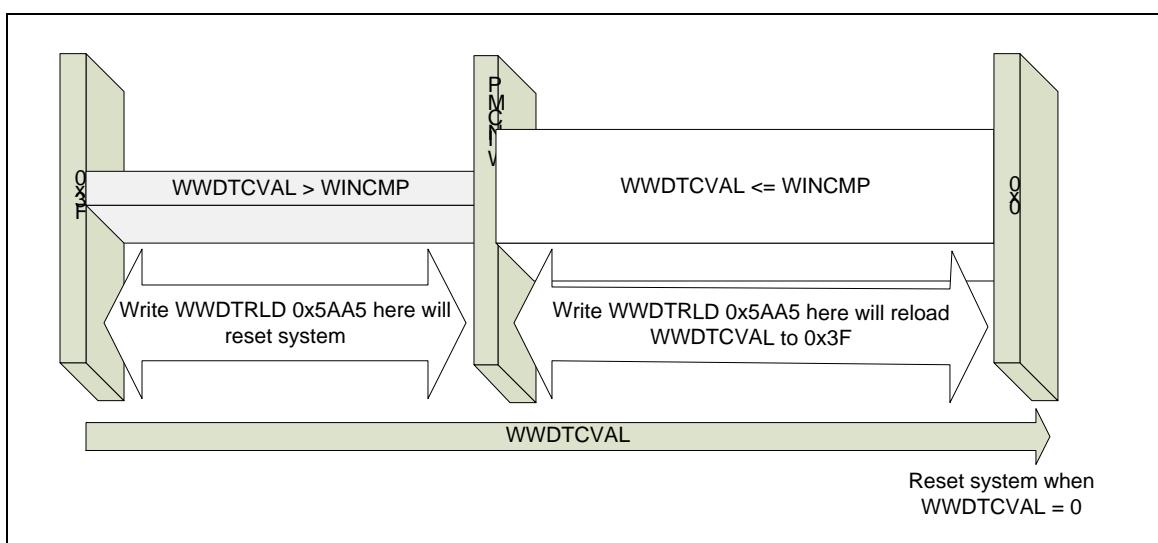


图 6-114 窗看门狗定时器复位和重载行为

**WWDT 窗设定限制**

当用户写 0x00005AA5 到 WWDTRLD 重载 WWDT 计数器的值为 0x3F 时，需要 3 个 WWDT 时钟来使重载命令和实际执行重载行为之间同步。这就意味着如果用户设定 PERIODSEL 等于 0000，计数器预分频值是 1，WINCMP 的值必须大于 2，否则，写 WWDTRLD 重载 WWDT 计数器的值为 0x3F 无法完成，同时 WWDTIF 置位并且 WWDT 复位系统将总是发生。

PERIODSEL	Prescale Value	Valid WINCMP Value
0000	1	0x3 ~ 0x3F
0001	2	0x2 ~ 0x3F
Others	Others	0x0 ~ 0x3F

表6-24 WINCMP 设定限制

### 6.16.6 寄存器映射

R: 只读, W: 只写, R/W: 可读写

寄存器	偏移量	R/W	描述	复位后的值
<b>WWDT 基地址:</b>				
<b>WWDT_BA = 0x4000_4100</b>				
WWDTRLD	WWDT_BA+0x00	W	窗看门狗定时器重载计数器寄存器	0x0000_0000
WWDTCSR	WWDT_BA+0x04	R/W	窗看门狗定时器控制寄存器	0x003F_0800
WWDTSR	WWDT_BA+0x08	R/W	窗看门狗定时器状态寄存器	0x0000_0000
WWDTCSR	WWDT_BA+0x0C	R	窗看门狗定时器计数值寄存器	0x0000_003F

### 6.16.7 寄存器描述

#### 窗看门狗定时器重载计数器寄存器(WWDTRLD)

寄存器	偏移量	R/W	描述	复位后的值
WWDTRLD	WWDT_BA+0x00	W	窗看门狗定时器 控制 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]							

Bits	描述	
[31:0]	WWDTRLD	<b>WWDT 重载计数值寄存器</b> 写 0x00005AA5 到该寄存器将重载 WWDT 计数器的值为 0x3F。 注：只有当WWDT当前计数器的值在0 到 WINCMP 之间时，软件才可以写 WWDTRLD 来重载 WWDT 计数器的值。如果当前WWDT计数器的值大于 WINCMP，软件写WWDTRLD 将导致WWDT立即发出复位信号。

**窗看门狗定时器 控制 寄存器(WWDTCR)**

寄存器	偏移量	R/W	描述	复位后的值
WWDTCR	WWDT_BA+0x04	R/W	窗看门狗定时器 控制 寄存器	0x003F_0800

注: 芯片上电或者复位以后该寄存器只能被写一次

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	

Bits	描述	
[31]	DBGACK_WWDT	<b>ICE调试模式应答禁止控制</b> 0 = ICE 调试模式应答影响窗看门狗定时器计数。 当ICE调试模式应答时，窗看门狗定时器计数器的值将维持不变。 1 = ICE 调试模式应答禁止。 无论CPU是否处于ICE调试模式，窗看门狗定时器计数器将继续正常计数。
[30:22]	保留	保留.
[21:16]	WINCMP	<b>WWDT 窗比较寄存器</b> 写该寄存器调整有效的重载窗。 注：只有当WWDT当前计数器的值在0 到 WINCMP之间时，软件才可以写WWDTRLRD 来重载 WWDT 计数器的值。如果当前WWDT计数器的值大于WINCMP，软件写WWDTRLRD 将导致WWDT立即发出复位信号。
[15:12]	保留	保留.
[11:8]	PERIODSEL	<b>WWDT 计数器预分频周期选择</b> 0000 = 预分频等于 1, 最大超时周期是 $1 * 64 * T_{WWDT}$ . 0001 = 预分频等于 2, 最大超时周期是 $2 * 64 * T_{WWDT}$ . 0010 = 预分频等于 4, 最大超时周期是 $4 * 64 * T_{WWDT}$ . 0011 = 预分频等于 8, 最大超时周期是 $8 * 64 * T_{WWDT}$ . 0100 = 预分频等于 16, 最大超时周期是 $16 * 64 * T_{WWDT}$ . 0101 = 预分频等于 32, 最大超时周期是 $32 * 64 * T_{WWDT}$ . 0110 = 预分频等于 64, 最大超时周期是 $64 * 64 * T_{WWDT}$ . 0111 = 预分频等于 128, 最大超时周期是 $128 * 64 * T_{WWDT}$ . 1000 = 预分频等于 192, 最大超时周期是 $192 * 64 * T_{WWDT}$ .

		1001 = 预分频等于256, 最大超时周期是 $256 * 64 * T_{WWDT}$ . 1010 = 预分频等于384, 最大超时周期是 $384 * 64 * T_{WWDT}$ . 1011 = 预分频等于512, 最大超时周期是 $512 * 64 * T_{WWDT}$ . 1100 = 预分频等于768, 最大超时周期是 $768 * 64 * T_{WWDT}$ . 1101 = 预分频等于1024, 最大超时周期是 $1024 * 64 * T_{WWDT}$ . 1110 = 预分频等于1536, 最大超时周期是 $1536 * 64 * T_{WWDT}$ . 1111 = 预分频等于2048, 最大超时周期是 $2048 * 64 * T_{WWDT}$ .
[7:2]	保留	保留.
[1]	<b>WWDTIE</b>	<b>WWDT 中断使能控制</b> 如果该位使能, WWDT 计数器比较匹配中断将发生。 0 = 禁止WWDT 计数器比较匹配中断. 1 = 使能WWDT 计数器比较匹配中断.
[0]	<b>WWDTEN</b>	<b>WWDT 使能控制</b> 0 = WWDT 计数器停止. 1 = WWDT 计数器开始计数.

窗看门狗定时器 状态 寄存器(WWDTSR)

寄存器	偏移量	R/W	描述	复位后的值
WWDTSR	WWDT_BA+0x08	R/W	窗看门狗定时器 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

Bits	描述	
[31:2]	保留	保留.
[1]	WWDTRF	<b>WWDT 超时复位标志</b> 该位指示系统是否已经被WWDT超时复位。 0 = 没有发生WWDT 超时复位。 1 = 发生WWDT超时复位。 注: 该位写1清0。
[0]	WWDTIF	<b>WWDT 比较匹配中断标志</b> 该位指示当WWDT计数器的值等于WINCMP的值时，中断标志的状态。 0 = 没有影响。 1 = WWDT 计数器的值匹配WINCMP 的值。 注: 该位写1清0



### 窗看门狗定时器 当前计数值 寄存器(WWDTCVR)

寄存器	偏移量	R/W	描述	复位后的值
WWDTCVR	WWDT_BA+0x0C	R	窗看门狗定时器 当前计数值 寄存器	0x0000_003F

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

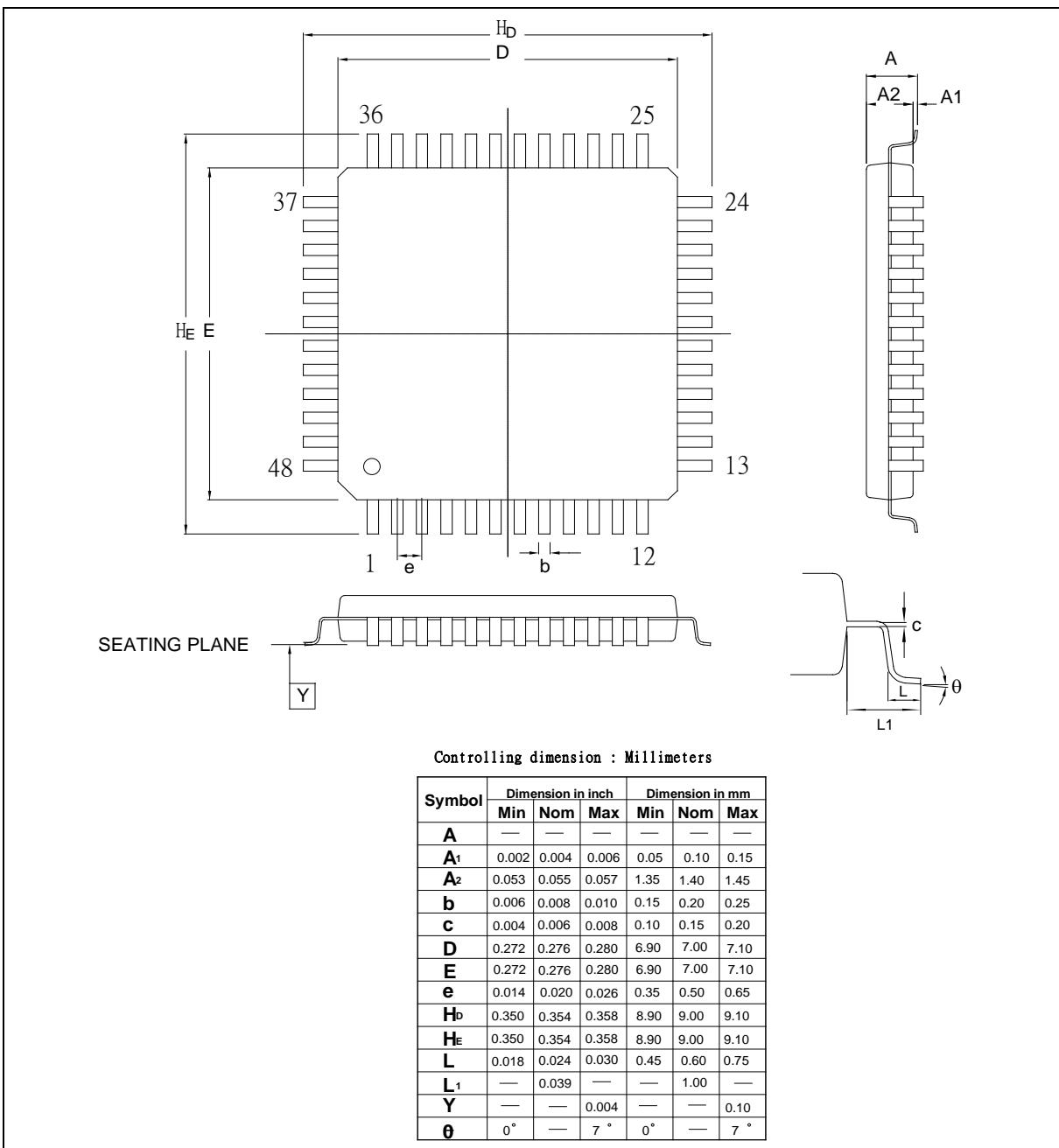
Bits	描述	
[31:6]	保留	保留.
[5:0]	WWDTCVAL	WWDT 计数器的值 WWDTCVAL 将被硬件连续更新直到反应6-bit下数计数器的值。

## 7 电器特性

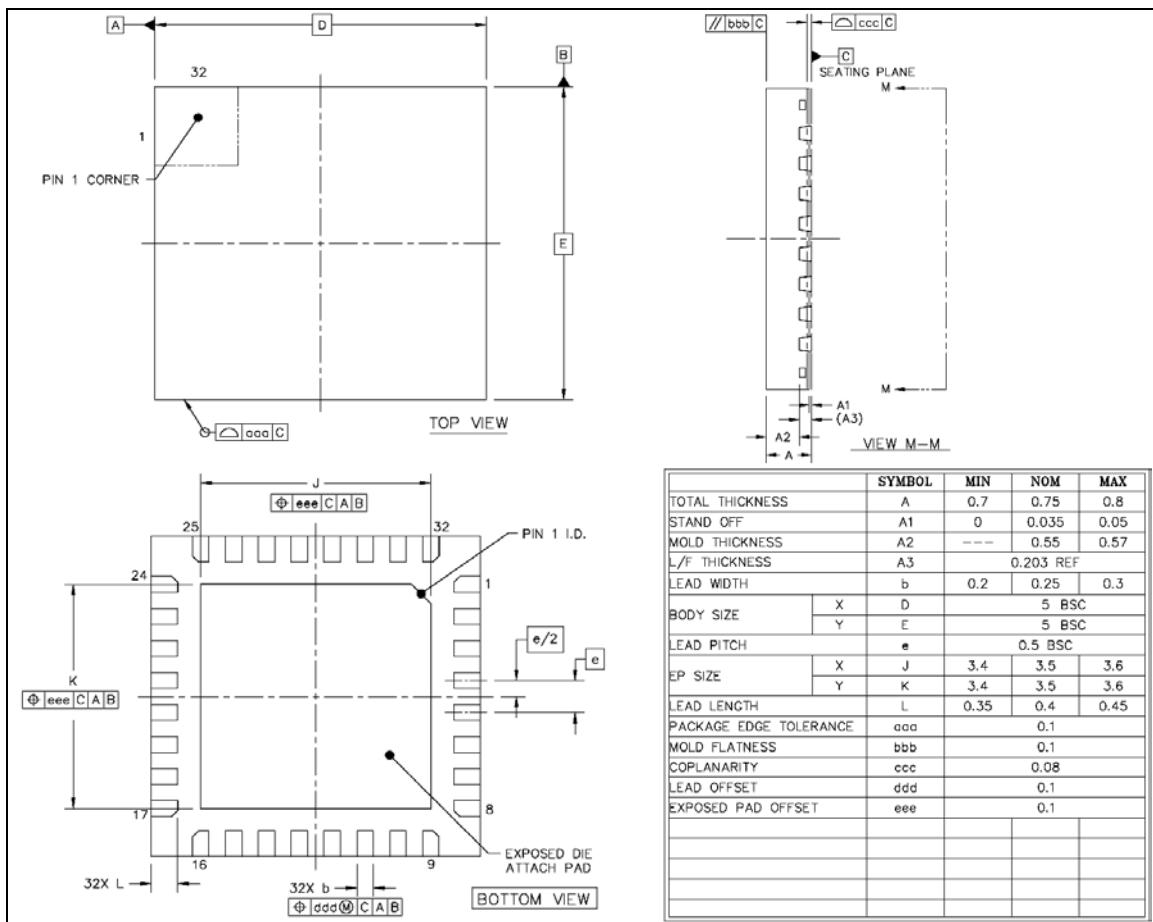
请参考M051系列DataSheet

## 8 封装尺寸

### 8.1 LQFP-48 (7x7x1.4mm<sup>2</sup> Footprint 2.0mm)



## 8.2 QFN-33 (5X5 mm<sup>2</sup>, Thickness 0.8mm, Pitch 0.5 mm)





## 9 版本历史

版本	日期	页	描述
V1.00	2014年6月5日,	-	初次发行中文版本



### 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.*