



ARM CORTEX™-M0

32-BIT 微控制器

**NuMicro™ NUC100 系列
NUC130/NUC140
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.

目录

目录	2
图	7
表	12
1	概述 13
2	特性 14
2.1	NuMicro™ NUC130 特征 – Automotive Line 14
2.2	NuMicro™ NUC140 特性 – Connectivity Line 18
3	编号信息列表及管脚名称定义 22
3.1	NuMicro™ NUC130 产品选型指南 22
3.1.1	NuMicro™ NUC130 Automotive Line 选型指南 22
3.2	NuMicro™ NUC140 产品选型指南 23
3.2.1	NuMicro™ NUC140 Connectivity Line 选型指南 23
3.3	管脚配置 24
3.3.1	NuMicro™ NUC130/NUC140 管脚图 24
3.4	管脚功能描述 30
3.4.1	NuMicro™ NUC130/NUC140 管脚定义 30
4	框图 46
4.1	NuMicro™ NUC130/NUC140 框图 46
4.1.1	NuMicro™ NUC130 框图 46
4.1.2	NuMicro™ NUC140 框图 47
5	功能描述 48
5.1	ARM® Cortex™-M0 内核 48
5.2	系统管理器 50
5.2.1	概述 50
5.2.2	系统复位 50
5.2.3	系统电源分配 51
5.2.4	系统内存映射 53
5.2.5	系统管理器控制寄存器 55
5.2.6	系统定时器 (SysTick) 92
5.2.7	嵌套向量中断控制器 (NVIC) 98
5.2.8	系统控制寄存器 122
5.3	时钟控制器 130
5.3.1	概述 130
5.3.2	时钟发生器 133
5.3.3	系统时钟 & SysTick 时钟 134
5.3.4	外围设备时钟 135

5.3.5	掉电模式时钟	135
5.3.6	分频器输出	136
5.3.7	寄存器映射	137
5.3.8	寄存器描述	138
5.4	USB 设备控制器 (USB)	157
5.4.1	概述	157
5.4.2	特征	157
5.4.3	框图	158
5.4.4	功能描述	159
5.4.5	寄存器与内存映射	163
5.4.6	寄存器描述	165
5.5	通用 I/O (GPIO)	183
5.5.1	概述	183
5.5.2	特征	183
5.5.3	功能描述	184
5.5.4	寄存器映射	186
5.5.5	寄存器描述	191
5.6	I ² C 串行接口控制器 (Master/Slave) (I ² C)	204
5.6.1	概述	204
5.6.2	特征	205
5.6.3	功能描述	206
5.6.4	协议寄存器	209
5.6.5	寄存器映射	212
5.6.6	寄存器描述	213
5.6.7	操作模式	222
5.6.8	五种操作模式中的数据传输	223
5.7	PWM 发生器和捕捉定时器 (PWM)	229
5.7.1	概述	229
5.7.2	特征	230
5.7.3	框图	231
5.7.4	功能描述	235
5.7.5	寄存器映射	242
5.7.6	寄存器描述	245
5.8	实时时钟 (RTC)	268
5.8.1	概述	268
5.8.2	特征	268
5.8.3	框图	269
5.8.4	功能描述	270
5.8.5	寄存器映射	272
5.8.6	寄存器描述	273
5.9	串行外围设备接口 (SPI)	288

5.9.1	概述	288
5.9.2	特征	288
5.9.3	框图	289
5.9.4	功能描述	290
5.9.5	时序图	297
5.9.6	编程例程	299
5.9.7	寄存器映射	302
5.9.8	寄存器描述	303
5.10	定时器控制器 (TMR)	316
5.10.1	概述	316
5.10.2	特征	316
5.10.3	框图	317
5.10.4	功能描述	318
5.10.5	寄存器映射	321
5.10.6	寄存器描述	323
5.11	看门狗定时器 (WDT)	333
5.11.1	概述	333
5.11.2	特征	335
5.11.3	框图	335
5.11.4	寄存器映射	336
5.11.5	寄存器描述	337
5.12	UART 接口控制器 (UART)	340
5.12.1	概述	340
5.12.2	特征	342
5.12.3	框图	343
5.12.4	IrDA 模式	346
5.12.5	LIN (Local Interconnection Network) 模式	348
5.12.6	RS-485 功能模式	349
5.12.7	寄存器映射	351
5.12.8	寄存器描述	353
5.13	控制器局域网 (CAN)	379
5.13.1	概述	379
5.13.2	特征	379
5.13.3	框图	380
5.13.4	功能描述	381
5.13.5	测试模式	382
5.13.6	CAN 通信	384
5.13.7	寄存器描述	403
5.13.8	寄存器映射	403
5.13.9	CAN 接口复位状态	404
5.14	PS/2 设备控制器 (PS2D)	443

5.14.1	概述	443
5.14.2	特征	443
5.14.3	框图	444
5.14.4	功能描述	445
5.14.5	寄存器映射	451
5.14.6	寄存器描述	452
5.15	I ² S 控制器 (I ² S)	460
5.15.1	概述	460
5.15.2	特征	460
5.15.3	框图	461
5.15.4	功能描述	462
5.15.5	寄存器映射	464
5.15.6	寄存器描述	465
5.16	模拟数字转换 (ADC)	478
5.16.1	概述	478
5.16.2	特征	478
5.16.3	框图	479
5.16.4	功能描述	480
5.16.5	寄存器映射	486
5.16.6	寄存器描述	487
5.17	模拟比较器 (CMP)	502
5.17.1	概述	502
5.17.2	特征	502
5.17.3	框图	503
5.17.4	功能描述	504
5.17.5	寄存器映射	505
5.17.6	寄存器描述	506
5.18	PDMA 控制器 (PDMA)	510
5.18.1	概述	510
5.18.2	特征	510
5.18.3	框图	511
5.18.4	功能描述	512
5.18.5	寄存器映射	513
5.18.6	寄存器描述	514
5.19	外部总线接口 (EBI)	537
5.19.1	概述	537
5.19.2	特征	537
5.19.3	框图	538
5.19.4	功能描述	538
5.19.5	寄存器映射	544
5.19.6	寄存器描述	544

6	FLASH 内存控制器 (FMC)	548
6.1	概述	548
6.2	特征	548
6.3	框图	549
6.4	Flash 内存结构	550
6.5	启动选择	552
6.6	数据 Flash	552
6.7	用户配置	553
6.8	在系统编程 (ISP)	557
6.8.1	ISP 程序	557
6.9	Flash 控制寄存器映射	560
6.10	Flash 控制寄存器描述	561
7	电气特性	571
7.1	绝对最大额定值	571
7.2	DC 电气特性	572
7.2.1	NuMicro™ NUC130/NUC140 DC 电气特性	572
7.3	AC 电气特性	577
7.3.1	外部 4~24 MHz 高速振荡器	577
7.3.2	外部 4~24 MHz 高速晶振	577
7.3.3	外部 32.768 kHz 低速晶振	579
7.3.4	内部 22.1184 MHz 高速振荡器	579
7.3.5	内部 10 kHz 低速振荡器	579
7.4	模拟量特性	580
7.4.1	12-位 SARADC 规格	580
7.4.2	LDO 规格和电源管理	581
7.4.3	低压复位说明	582
7.4.4	欠压检测说明	582
7.4.5	上电复位说明 (5 V)	582
7.4.6	温度传感器说明	583
7.4.7	比较器说明	583
7.4.8	USB PHY 说明	584
7.5	SPI 动态特性	586
8	封装定义	588
8.1	100L LQFP (14x14x1.4 mm footprint 2.0mm)	588
8.2	64L LQFP (10x10x1.4mm footprint 2.0 mm)	589
8.3	48L LQFP (7x7x1.4mm footprint 2.0mm)	590
9	版本历史	591

图

图 3-1 NuMicro™ NUC100 系列选型码	23
图 3-2 NuMicro™ NUC130 LQFP 100-pin 管脚图	24
图 3-3 NuMicro™ NUC130 LQFP 64-pin 管脚图	25
图 3-4 NuMicro™ NUC130 LQFP 48-pin 管脚图	26
图 3-5 NuMicro™ NUC140 LQFP 100-pin管脚图	27
图 3-6 NuMicro™ NUC140 LQFP 64-pin管脚图	28
图 3-7 NuMicro™ NUC140 LQFP 48-pin管脚图	29
图 4-1 NuMicro™ NUC130 框图	46
图 4-2 NuMicro™ NUC140 框图	47
图 5-1 功能框图	48
图 5-2 NuMicro™ NUC140 电源分配图	51
图 5-3 NuMicro™ NUC130 电源分配图	52
图 5-4 时钟发生器全局框图	131
图 5-5 时钟发生器框图	133
图 5-6 系统时钟框图	134
图 5-7 SysTick 时钟控制框图	134
图 5-8 分频器的时钟源	136
图 5-9 分频器的框图	136
图 5-10 USB 框图	158
图 5-11 唤醒中断的操作流程	160
图 5-12 端点 SRAM 的结构	161
图 5-13 数据传入事务紧跟Setup 事务	162
图 5-14 数据输出图	162
图 5-15 推挽输出	184
图 5-16 开漏输出	185
图 5-17 混双端 I/O 模式	185
图 5-18 I ² C 总线时序	204
图 5-19 I ² C 协议	206
图 5-20 主机向从机传输数据	206
图 5-21 主机从从机读取数据	206

图 5-22 起始 (START) 和停止 (STOP) 条件.....	207
图 5-23 I ² C 总线上的位传输	208
图 5-24 I ² C 总线上的应答信号.....	208
图 5-25 I ² C 数据移位方向	209
图 5-26: I ² C 超时计数器框图	211
图 5-27 传输流程图	223
图 5-28 主机传送模式	224
图 5-29 主机接收模式	225
图 5-30 从机传送模式	226
图 5-31 从机接收模式	227
图 5-32 广播呼叫模式	228
图 5-33 PWM 发生器 0 时钟源控制	231
图 5-34 PWM 发生器 0 结构框图	231
图 5-35 PWM 发生器2 时钟源控制	232
图 5-36 PWM 发生器 2 结构框图	232
图 5-37 PWM 发生器 4 时钟源控制	233
图 5-38 PWM 发生器 4 结构框图	233
图 5-39 PWM 发生器 6 时钟源控制	234
图 5-40 PWM 发生器 6 结构框图	234
图 5-41 PWM-定时器内部比较器输出	235
图 5-42 PWM-定时器操作时序	236
图 5-43 PWM 双缓存图解	236
图 5-44 PWM 控制输出占空比	237
图 5-45 PWM-对输出带死区发生器操作	237
图 5-46 捕捉操作时序	238
图 5-47 PWM A 组PWM-定时器中断结构图	239
图 5-48 PWM B 组PWM-定时器中断结构图	239
图 5-49 RTC 框图	269
图 5-50 SPI 框图	289
图 5-51 SPI 主机模式应用框图	290
图 5-52 SPI 从机模式应用框图	290
图 5-53 可调串行时钟频率	292
图 5-54 一次传输报文的 32-位	292

图 5-55 一次传输两个报文 (Burst 模式)	293
图 5-56 字节重排序	294
图 5-57 字节休眠的时序波形	295
图 5-58 两位传输模式 (从机模式)	296
图 5-59 SPI 主机模式下的时序	297
图 5-60 SPI 主机模式下的时序 (Alternate Phase of SPICLK)	298
图 5-61 SPI 从机模式下的时序	298
图 5-62 SPI 从机模式下的时序 (Alternate Phase of SPICLK)	299
图 5-63 定时器控制器框图	317
图 5-64 定时器控制器的时钟源	317
图 5-65 Continuous Counting 模式	319
图 5-66 中断和复位信号时序	334
图 5-67 看门狗定时器时钟控制	335
图 5-68 看门狗定时器框图	335
图 5-69 UART 时钟控制图	343
图 5-70 UART 框图	343
图 5-71 自动流控制框图	345
图 5-72 IrDA 框图	346
图 5-73 IrDA TX/RX 时序图	347
图 5-74 LIN 帧结构	348
图 5-75 RS-485 帧结构	350
图 5-76 CAN 外设框图	380
图 5-77 Silent 模式下的 CAN 内核	382
图 5-78 Loop Back 模式下的 CAN 内核	383
图 5-79 环回模式和静默模式整合下的 CAN 内核	383
图 5-80 IFn 寄存器和报文间的数据传输	386
图 5-81 应用软件处理 FIFO 缓存	391
图 5-82 位时间	393
图 5-83 传播时间段	394
图 5-84 “late” 和 “early” 边沿的同步	396
图 5-85 过滤短显性尖峰脉冲	397
图 5-86 CAN 内核的协议控制器结构	399
图 5-87 PS/2 设备框图	444

图 5-88 设备向主机传输的数据格式	446
图 5-89 主机向设备传输的数据格式	446
图 5-90 PS/2 位数据格式	447
图 5-91 PS/2 总线时序	447
图 5-92 PS/2 数据格式	450
图 5-93 I ² S 时钟控制框图	461
图 5-94 I ² S 控制器框图	461
图 5-95 I ² S 总线时序图 (Format =0)	462
图 5-96 MSB 校正 (MSB Justified) 时序图 (Format=1)	462
图 5-97 不同 I ² S 模式的 FIFO 内容	463
图 5-98 ADC 控制器框图	479
图 5-99 ADC 转换器自校正时序图	480
图 5-100 ADC 时钟控制	481
图 5-101 单一模式转换时序图	481
图 5-102 使能通道上的单周期扫描模式时序图	482
图 5-103 使能通道上的连续扫描模式时序图	483
图 5-104 A/D转换结果监控逻辑图	484
图 5-105 A/D 控制器中断	485
图 5-106 ADC单端输入转换电压和转换结果图	490
图 5-107 ADC差分输入转换电压和转换结果图	490
图 5-108 模拟比较器框图	503
图 5-109 模拟比较器控制器中断源	504
图 5-110 PDMA 控制器框图	511
图 5-111 EBI 框图	538
图 5-112 16-位 EBI 数据宽度与 16-位设备的连接	539
图 5-113 8-位 EBI 数据宽度和 8-位设备的连接	539
图 5-114 16-位数据宽度的时序控制波形	541
图 5-115 8-位数据宽度时序控制波形	542
图 5-116 插入空闲周期的时序控制波形	543
图 6-1 Flash 内存控制器框图	549
图 6-2 Flash 内存组织结构	551
图 6-3 Flash 内存结构	552
图 7-1 典型晶振应用电路	578



图 7-2 SPI 主机动态特性时序图	587
图 7-3 SPI 从机动态特性时序图	587

表

表 1-1 所支持的接口列表.....	13
表 5-1 片上控制器的地址空间分配	54
表 5-2 异常模式	99
表 5-3 系统中断映射	100
表 5-4 向量表格式	101
表 5-5 掉电模式控制表	141
表 5-6 字节排序和字节休眠条件.....	295
表 5-7 看门狗定时溢出间隔选择.....	333
表 5-8 UART 波特率公式.....	340
表 5-9 UART 波特率设置表	341
表 5-10 DMA 模式下UART 的中断源和标志位表	371
表 5-11 软件模式下UART 的中断源和标志位表	371
表 5-12 波特率方程式表	374
表 5-13 一个发送对象的初始化	388
表 5-14 接收对象的初始化.....	389
表 5-15 CAN 位时间参数	393
表 5-16 CAN 寄存器每位的位功能映射	407
表 5-17 错误码.....	411
表 5-18 中断源.....	414
表 5-19 IF1 和 IF2 报文接口寄存器	418
表 5-20 报文内存中的报文对象的结构	432
表 6-1 内存地址映射	550
表 6-2 ISP 模式.....	559

1 概述

NuMicro™ NUC100系列是32位的内嵌ARM® Cortex™-M0核的微控制器，适用于工业控制和需要丰富的通信接口的应用领域，Cortex™-M0是ARM最新的32位嵌入式处理器，拥有与传统8051单片机之匹敌的价格优势。NuMicro™ NUC100 系列包括NUC100, NUC120, NUC130和NUC140。

NuMicro™ NUC130 Automotive Line带CAN功能，内嵌Cortex™-M0内核，最高可运行至50 MHz，内建32K/64K/128K字节的Flash存储器，以及4K/8K/16K字节SRAM，4K字节用于存储ISP引导代码的ROM，和4K字节的数据 Flash 存储器。另外还有丰富的外设，如定时器，看门狗定时器，RTC，PDMA，UART，SPI，I²C，I²S，PWM 定时器，GPIO，LIN，CAN，PS/2，12位ADC，模拟比较器，低电压复位控制和欠压检测功能。

NuMicro™ NUC140 Connectivity Line 带全速USB 2.0 和 CAN功能，内嵌Cortex™-M0内核，最高可运行至50 MHz，内建32K/64K/128K字节的Flash存储器，以及4K/8K/16K字节SRAM，4K字节用于存储ISP引导代码的ROM，和4K字节的数据 Flash 存储器。另外还有丰富的外设，如定时器，看门狗定时器，RTC，PDMA，UART，SPI，I²C，I²S，PWM 定时器，GPIO，LIN，CAN，PS/2，USB 2.0 FS 设备，12位ADC，模拟比较器，低电压复位控制和欠压检测功能。

Product Line	UART	SPI	I ² C	USB	LIN	CAN	PS/2	I ² S
NUC100	•	•	•				•	•
NUC120	•	•	•	•			•	•
NUC130	•	•	•		•	•	•	•
NUC140	•	•	•	•	•	•	•	•

表 1-1 所支持的接口列表



2 特性

该器件的功能依赖于产品线和他们的子系统产品。

2.1 NuMicro™ NUC130 特征 – Automotive Line

- 内核
 - ARM® Cortex™-M0 内核最高允许 50 MHz
 - 一个 24-位系统定时器
 - 支持低功耗睡眠模式
 - 单周期32位硬件乘法器
 - 嵌套向量中断控制器 (NVIC) 用于控制32个中断源，每个中断源可设置为4个优先级
 - 支持串行线调试 (SWD) 带2个观察点/4个断点
- 内建 LDO，宽电压工作范围 2.5 V 到 5.5 V
- Flash 存储器
 - 32K/64K/128K 字节 Flash 用于存储程序代码
 - 4KB flash 用于存储ISP引导代码
 - 支持在系统编程 (ISP) 方式更新应用程序
 - 支持512 字节单页擦除
 - 在128K字节系统中可配置数据FLASH地址和大小，在32K字节和64K字节系统中固定为4K字节数据
 - 通过SWD/ICE接口，支持2 线 ICP升级方式
 - 支持外部编程器并行高速编程模式
- SRAM 存储器
 - 4K/8K/16K 字节内建 SRAM
 - 支持 PDMA 模式
- PDMA (Peripheral DMA)
 - 支持9通道 PDMA 用于SRAM和周边设备的自动数据传输
- 时钟控制
 - 针对不同应用可灵活选择时钟
 - 内部 22.1184 MHz 高速振荡器可用于系统运行
 - ◆ 在+25 °C, VDD = 5.0 V时, 精度校正到± 1 %
 - ◆ 在-40 °C ~ +85 °C 和 VDD = 2.5 V ~ 5.5 V范围内, 精度为± 3 %
 - 内部低功耗 10 KHz 低速振荡器用于看门狗及掉电模式唤醒等功能
 - 支持一组PLL, 高至 50 MHz, 用于高性能的系统运行
 - 外部 4~24 MHz 晶振输入用于精准的定时操作
 - 外部 32.768 kHz 晶振输入用于 RTC 及低功耗模式操作
- GPIO
 - 四种 I/O 模式:
 - ◆ 准双向模式
 - ◆ 推挽输出模式
 - ◆ 开漏输出模式
 - ◆ 高阻输入模式
 - TTL/Schmitt 触发输入可选
 - I/O 引脚可被配置为边沿/电平触发模式的中断源



- 支持大电流驱动和灌入 I/O 模式
- Timer
 - 支持4组32位定时器，每个定时器有一个24位向上计数定时器和一个8位预分频计数器
 - 每个定时器有独立的时钟源
 - 提供 one-shot, periodic, toggle and continuous 计数操作模式
 - 支持事件计数功能
 - 支持输入捕捉功能
- Watchdog Timer
 - 多路时钟源
 - 从1.6ms 到26.0sec 有8个可选的定时溢出周期(取决于所选的时钟源)
 - WDT 可用作掉电模式/睡眠模式的唤醒
 - 看门狗定时溢出的中断/复位选择
- RTC
 - 通过频率补偿寄存器(FCR) 支持软件频率补偿功能
 - 支持RTC计数(秒, 分, 小时) 及万年历功能 (日, 月, 年)
 - 支持闹铃寄存器 (秒, 分, 小时, 日, 月, 年)
 - 可选择为12小时制或24小时
 - 闰年自动识别
 - 支持周期时间滴答中断, 包括8个可选周期1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 和 1 秒
 - 支持唤醒功能
- PWM/Capture
 - 内建四个16位PWM产生器, 可输出8路PWM或4组互补配对PWM
 - 每个PWM产生器配有一个时钟源选择器, 一个时钟分频器, 一个8位时钟预分频和一个用于互补配对PWM的死区发生器
 - 8路16位捕捉定时器 (共享PWM定时器) 提供8路输入的上升/下降沿的捕捉功能
 - 支持捕捉(Capture)中断
- UART
 - 最多三组UART控制器
 - 支持流控 (TXD, RXD, CTS 和 RTS)
 - UART0 带 64-字节 FIFO 用于高速模式
 - UART1/2 (可选) 带16-字节 FIFO 用于标准模式
 - 支持 IrDA (SIR) 和 LIN 功能
 - 支持 RS-485 9 位模式和方向控制
 - 可编程波特率发生器频率高至1/16系统时钟
 - 支持 PDMA 模式
- SPI
 - 最高支持4组 SPI 控制器
 - 主机速率高至 32 MHz, 从机高至 10 MHz (芯片工作在 5V 状态时)
 - 支持 SPI 主机/从机模式
 - 全双工同步串行数据传输
 - 可变数据长度 (从1位到 32 位) 传输模式
 - 可设置MSB 或LSB 在前的传输模式
 - 在时钟上升沿或下降沿接收还是发送是独立配置的
 - 当作为主机时2条从机片选线, 作为从机时1条从机片选线

- 支持 32-bit 传输模式下的字节睡眠模式
- 支持 PDMA 模式
- 支持三线无从机选择信号的双向接口
- I²C
 - 最多支持2组 I²C 设备
 - 主机/从机模式
 - 主从机之间双向数据传输
 - 多主机总线支持（无中心主机）
 - 多主机间同时传输数据仲裁，避免总线上串行数据损坏
 - 总线采用串行同步时钟，可实现设备之间以不同的速率传输
 - 串行同步时钟可作为握手方式控制总线上数据暂停及恢复传送
 - 可编程的时钟适用于不同速率控制
 - I²C总线上支持多地址识别（4个从机地址带mask选项）
- I²S
 - 外部音频 CODEC 接口
 - 可作主机也可作从机模式
 - 能处理 8, 16, 24 和 32 位字
 - 支持单声道和立体声的音频数据
 - 支持I²S 和 最高有效位数据格式
 - 提供两组 8 字的FIFO数据缓存，一组用于发送，一组用于接收
 - 缓冲区超过可编程边界时，产生中断请求
 - 支持两组DMA请求，一组用于发送，另一组用于接收
- PS/2 设备控制器
 - 禁止 Host 通信和请求发送检测
 - 接收帧错误检测
 - 可编程的 1 到 16 字节的发送缓冲以减少CPU的负担
 - 数据接收的双缓冲
 - 软件可控总线
- CAN 2.0
 - 支持 CAN 2.0A 和 2.0B 协议
 - 位传输速率最高至1M bit/s
 - 32个报文对象
 - 每个报文对象有其自己的标识符掩码
 - 可编程的 FIFO 模式（链接报文对象）
 - 可屏蔽中断
 - 时间触发的CAN应用中禁用自动重传模式
 - 支持掉电模式唤醒功能
- 支持EBI（外部总线接口）（仅100-pin and 64-pin 封装系列）
 - 可访问的空间：8位模式为64KB 或16位模式为 128KB
 - 支持8bit/16bit 数据宽度
 - 在16位数据宽度模式下支持字节写入
- ADC
 - 12位ADC，转换速率达 700K SPS
 - 最多 8 通道单端模式输入或4 通道差分模式输入



- 单一扫描模式/单周期扫描模式/连续扫描模式
- 每个通道有独立的结果寄存器
- 扫描使能通道
- 阈电压侦测
- 软件编程或外部管脚触发开始转换
- 支持 PDMA 模式
- 模拟比较器(Analog Comparator)
 - 2组模拟比较器模块
 - 负端电位可选择外部输入或内部频带间隙电压
 - 比较结果改变可作为中断触发条件
 - 支持掉电模式唤醒功能
- 内建温度传感器, 1°C 分辨率
- 欠压检测(Brown-Out detector)
 - 支持四级检测电压: 4.5 V/3.8 V/2.7 V/2.2 V
 - 支持欠压中断和复位选择
- 低压复位
 - 阈电压: 2.0 V
- 工作温度: -40°C~85°C
- 封装:
 - 无铅封装(RoHS)
 - LQFP 100-pin / 64-pin / 48-pin



2.2 NuMicro™ NUC140 特征 – Connectivity Line

- 内核
 - ARM® Cortex™-M0 内核最高运行 50 MHz
 - 一个 24-位系统定时器
 - 支持低功耗睡眠模式
 - 单周期32位硬件乘法器
 - 嵌套向量中断控制器 (NVIC) 用于控制32个中断源，每个中断源可设置为4个优先级
 - 支持串行线调试 (SWD) 带2个观察点/4个断点
- 内建 LDO，宽电压工作范围 2.5 V 到 5.5 V
- Flash 存储器
 - 32K/64K/128K 字节 Flash 用于存储程序代码
 - 4KB flash 用于存储ISP引导代码
 - 支持在系统编程 (ISP) 方式更新应用程序
 - 支持512 字节单页擦除
 - 在128K字节系统中可配置数据FLASH地址和大小，在32K字节和64K字节系统中固定为4K字节数据
 - 通过SWD/ICE接口，支持2 线 ICP升级方式
 - 支持外部编程器并行高速编程模式
- SRAM 存储器
 - 4K/8K/16K 字节内建 SRAM
 - 支持 PDMA 模式
- PDMA (Peripheral DMA)
 - 支持9通道 PDMA 用于SRAM和周边设备的自动数据传输
- 时钟控制
 - 针对不同应用可灵活选择时钟
 - 内部 22.1184 MHz 高速振荡器可用于系统运行
 - ◆ 在+25 °C, VDD = 5.0 V 时, 精度校正到± 1 %
 - ◆ 在-40 °C ~ +85 °C 和 VDD = 2.5 V ~ 5.5 V 范围内, 精度为± 3 %
 - 内部低功耗 10 KHz 低速振荡器用于看门狗及掉电模式唤醒等功能
 - 支持一组PLL, 高至 50 MHz, 用于高性能的系统运行
 - 外部 4~24 MHz 晶振输入用于 USB 和精准的定时操作
 - 外部 32.768 kHz 晶振输入用于 RTC 及低功耗模式操作
- GPIO
 - 四种 I/O 模式:
 - ◆ 准双向模式
 - ◆ 推挽输出模式
 - ◆ 开漏输出模式
 - ◆ 高阻输入模式
 - TTL/Schmitt 触发输入可选
 - I/O 引脚可被配置为边沿/电平触发模式的中断源
 - 支持大电流驱动和灌入 I/O 模式
- Timer



- 支持4组32位定时器，每个定时器有一个24位向上计数定时器和一个8位预分频计数器
- 每个定时器有独立的时钟源
- 提供 one-shot, periodic, toggle and continuous 计数操作模式
- 支持事件计数功能
- 支持输入捕捉功能
- Watchdog Timer
 - 多路时钟源
 - 从1.6ms 到26.0sec 有8个可选的定时溢出周期(取决于所选的时钟源)
 - WDT 可用作掉电模式/睡眠模式的唤醒
 - 看门狗定时溢出的中断/复位选择
- RTC
 - 通过频率补偿寄存器(FCR) 支持软件频率补偿功能
 - 支持RTC计数(秒, 分, 小时) 及万年历功能 (日, 月, 年)
 - 支持闹铃寄存器 (秒, 分, 小时, 日, 月, 年)
 - 可选择为12小时制或24小时
 - 闰年自动识别
 - 支持周期时间滴答中断, 包括8个可选周期1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 和 1秒
 - 支持唤醒功能
- PWM/Capture
 - 内建四个16位PWM产生器, 可输出8路PWM或4组互补配对PWM
 - 每个PWM产生器配有一个时钟源选择器, 一个时钟分频器, 一个8位时钟预分频和一个用于互补配对PWM的死区发生器
 - 8路16位捕捉定时器 (共享PWM定时器) 提供8路输入的上升/下降沿的捕捉功能
 - 支持捕捉(Capture)中断
- UART
 - 最多三组UART控制器
 - 支持流控 (TXD, RXD, CTS 和 RTS)
 - UART0 带 64-字节 FIFO 用于高速模式
 - UART1/2 (可选) 带16-字节 FIFO 用于标准模式
 - 支持 IrDA (SIR) 和 LIN 功能
 - 支持 RS-485 9 位模式和方向控制
 - 可编程波特率发生器频率高至1/16系统时钟
 - 支持 PDMA 模式
- SPI
 - 最高支持4组 SPI 控制器
 - 主机速率高至 32 MHz, 从机高至 10 MHz (芯片工作在 5V 状态时)
 - 支持 SPI 主机/从机模式
 - 全双工同步串行数据传输
 - 可变数据长度 (从1位到 32 位) 传输模式
 - 可设置MSB 或LSB 在前的传输模式
 - 在时钟上升沿或下降沿接收还是发送是独立配置的
 - 当作为主机时2条从机片选线, 作为从机时1条从机片选线
 - 支持 32-bit 传输模式下的字节睡眠模式
 - 支持 PDMA 模式
 - 支持三线无从机选择信号的双向接口

- I²C
 - 最多支持2组 I²C 设备
 - 主机/从机模式
 - 主从机之间双向数据传输
 - 多主机总线支持（无中心主机）
 - 多主机间同时传输数据仲裁，避免总线上串行数据损坏
 - 总线采用串行同步时钟，可实现设备之间以不同的速率传输
 - 串行同步时钟可作为握手方式控制总线上数据暂停及恢复传送
 - 可编程的时钟适用于不同速率控制
 - I²C总线上支持多地址识别（4个从机地址带mask选项）
- I²S
 - 外部音频 CODEC 接口
 - 可作主机也可作从机模式
 - 能处理8, 16, 24 和 32 位word
 - 支持单声道和立体声的音频数据
 - 支持I²S 和 最高有效位数据格式
 - 提供两组8字的FIFO数据缓存，一组用于发送，一组用于接收
 - 缓冲区超过可编程边界时，产生中断请求
 - 支持两组DMA请求，一组用于发送，另一组用于接收
- CAN 2.0
 - 支持 CAN 2.0A 和 2.0B 协议
 - 位传输速率最高至1M bit/s
 - 32个报文对象
 - 每个报文对象有其自己的标识符掩码
 - 可编程的 FIFO 模式（链接报文对象）
 - 可屏蔽中断
 - 时间触发的CAN应用中禁用自动重传模式
 - 支持掉电模式唤醒功能
- PS/2 设备控制器
 - 禁止 Host 通信和请求发送检测
 - 接收帧错误检测
 - 可编程的 1 到 16 字节的发送缓冲以减少CPU的负担
 - 数据接收的双缓冲
 - 软件可控总线
- USB 2.0 Full-Speed Device
 - 一组12Mbps的USB 2.0 FS 设备
 - 片内集成USB收发模块
 - 提供1组中断源，提供四个中断事件
 - 支持控制传输(Control)，批量传输(Bulk In/Out)，中断传输(Interrupt)及同步传输
 - 当总线上无信号达到3ms时，具有自动暂停的功能
 - 支持6组可编程端点(endpoints)
 - 512 字节内部 SRAM 作为 USB 的缓存区
 - 支持远程唤醒功能
- 支持EBI（外部总线接口）(100-pin and 64-pin Package Only)



- 可访问的空间：8位模式为64KB 或16位模式为128KB
- 支持8-位/16-位数据宽度
- 在16位数据宽度模式下支持字节写入
- ADC
 - 12位ADC，转换速率达 700K SPS
 - 最多8通道单端模式输入或4通道差分模式输入
 - 单一扫描模式/单周期扫描模式/连续扫描模式
 - 每个通道有独立的结果寄存器
 - 扫描使能通道
 - 阈电压侦测
 - 软件编程或外部管脚触发开始转换
 - 支持 PDMA 模式
- 模拟比较器(Analog Comparator)
 - 2组模拟比较器模块
 - 负端电位可选择外部输入或内部频带间隙电压
 - 比较结果改变可作为中断触发条件
 - 支持掉电模式唤醒功能
- 内建温度传感器， 1°C 分辨率
- 欠压检测(Brown-Out detector)
 - 支持四级检测电压：4.5 V/3.8 V/2.7 V/2.2 V
 - 支持欠压中断和复位选择
- 低压复位
 - 阈电压：2.0 V
- 工作温度：-40°C~85°C
- 封装：
 - 无铅封装(RoHS)
 - LQFP 100-pin / 64-pin / 48-pin

3 编号信息列表及管脚名称定义

3.1 NuMicro™ NUC130 产品选型指南

3.1.1 NuMicro™ NUC130 Automotive Line 选型指南

编号	APROM	RAM	Data Flash	ISP Loader ROM	I/O	Timer	各总线界面						I ² S	Comp.	PWM	ADC	RTC	EBI	ISP ICP	封装
							UART	SPI	I ² C	USB	LIN	CAN								
NUC130LC1CN	32 KB	4 KB	4 KB	4 KB	up to 35	4x32-bit	3	1	2	-	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC130LD2CN	64 KB	8 KB	4 KB	4 KB	up to 35	4x32-bit	3	1	2	-	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC130LE3CN	128 KB	16 KB	Definable	4 KB	up to 35	4x32-bit	3	1	2	-	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC130RC1CN	32 KB	4 KB	4 KB	4 KB	up to 49	4x32-bit	3	2	2	-	2	1	1	2	6	8x12-bit	v	v	v	LQFP64
NUC130RD2CN	64 KB	8 KB	4 KB	4 KB	up to 49	4x32-bit	3	2	2	-	2	1	1	2	6	8x12-bit	v	v	v	LQFP64
NUC130RD3CN	128 KB	16 KB	Definable	4 KB	up to 49	4x32-bit	3	2	2	-	2	1	1	2	6	8x12-bit	v	v	v	LQFP64
NUC130VE3CN	128 KB	16 KB	Definable	4 KB	up to 80	4x32-bit	3	4	2	-	2	1	1	2	8	8x12-bit	v	v	v	LQFP100

3.2 NuMicro™ NUC140 产品选型指南

3.2.1 NuMicro™ NUC140 Connectivity Line 选型指南

编号	APROM	RAM	Data Flash	ISP Loader ROM	I/O	Timer	各总线界面						I ² S	Comp.	PWM	ADC	RTC	EBI	ISP ICP	封装	
							UART	SPI	I ² C	USB	LIN	CAN									
NUC140LC1CN	32 KB	4 KB	4 KB	4 KB	up to 31	4x32-bit	2	1	2	1	2	1	1	1	4	8x12-bit	v	-	v	LQFP48	
NUC140LD2CN	64 KB	8 KB	4 KB	4 KB	up to 31	4x32-bit	2	1	2	1	2	1	1	1	4	8x12-bit	v	-	v	LQFP48	
NUC140LE3CN	128 KB	16 KB	Definable	4 KB	up to 31	4x32-bit	2	1	2	1	2	1	1	1	4	8x12-bit	v	-	v	LQFP48	
NUC140RC1CN	32 KB	4 KB	4 KB	4 KB	up to 45	4x32-bit	3	2	2	1	2	1	1	1	2	4	8x12-bit	v	v	v	LQFP64
NUC140RD2CN	64 KB	8 KB	4 KB	4 KB	up to 45	4x32-bit	3	2	2	1	2	1	1	1	2	4	8x12-bit	v	v	v	LQFP64
NUC140RE3CN	128 KB	16 KB	Definable	4 KB	up to 45	4x32-bit	3	2	2	1	2	1	1	1	2	4	8x12-bit	v	v	v	LQFP64
NUC140VE3CN	128 KB	16 KB	Definable	4 KB	up to 76	4x32-bit	3	4	2	1	2	1	1	1	2	8	8x12-bit	v	v	v	LQFP100

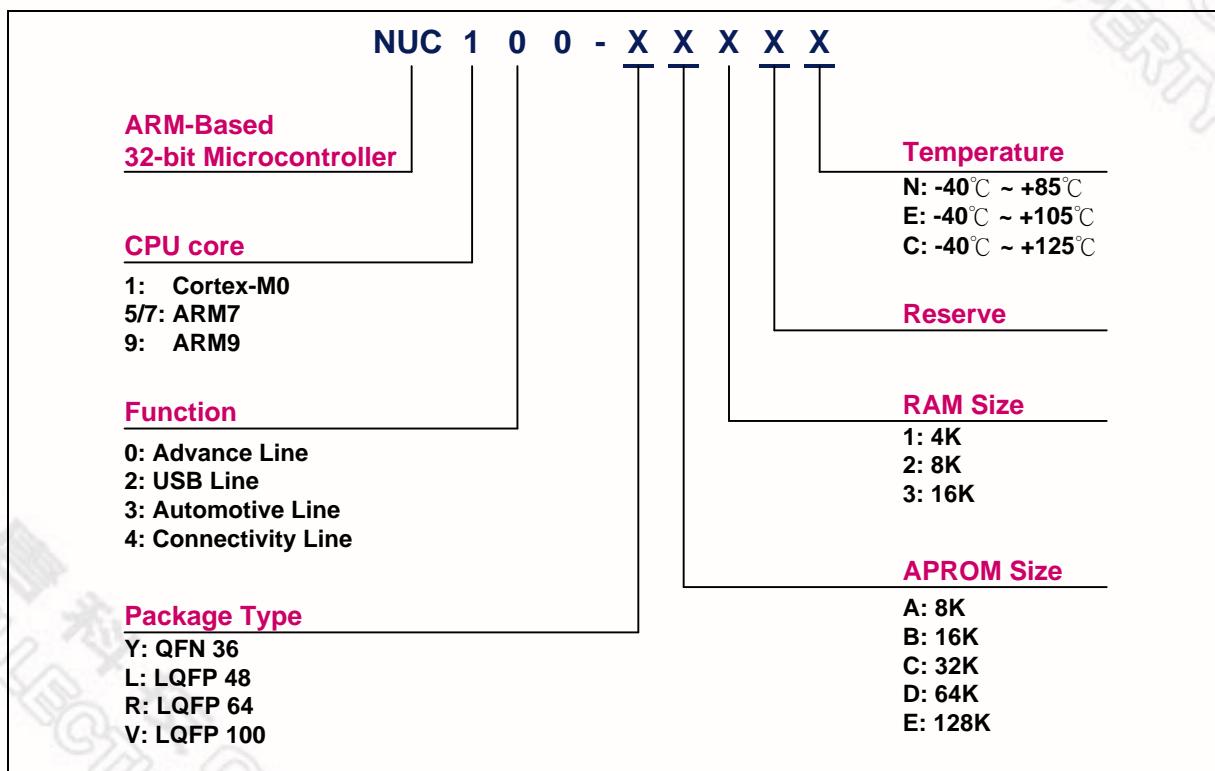


图 3-1 NuMicro™ NUC100 系列选型码

3.3 管脚配置

3.3.1 NuMicro™ NUC130/NUC140 管脚图

3.3.1.1 NuMicro™ NUC130 LQFP 100 pin

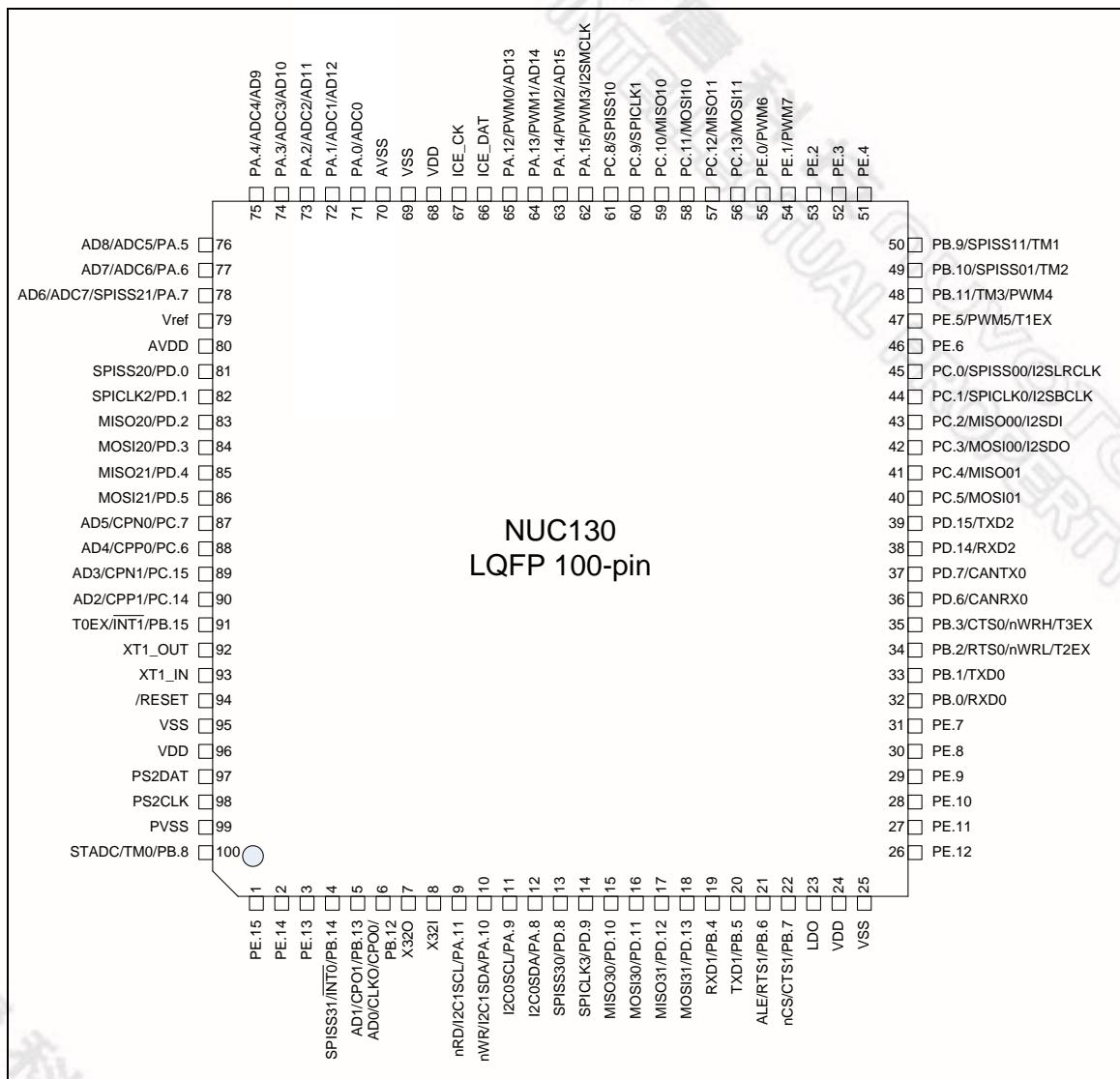


图 3-2 NuMicro™ NUC130 LQFP 100-pin 管脚图

3.3.1.2 NuMicro™ NUC130 LQFP 64 pin

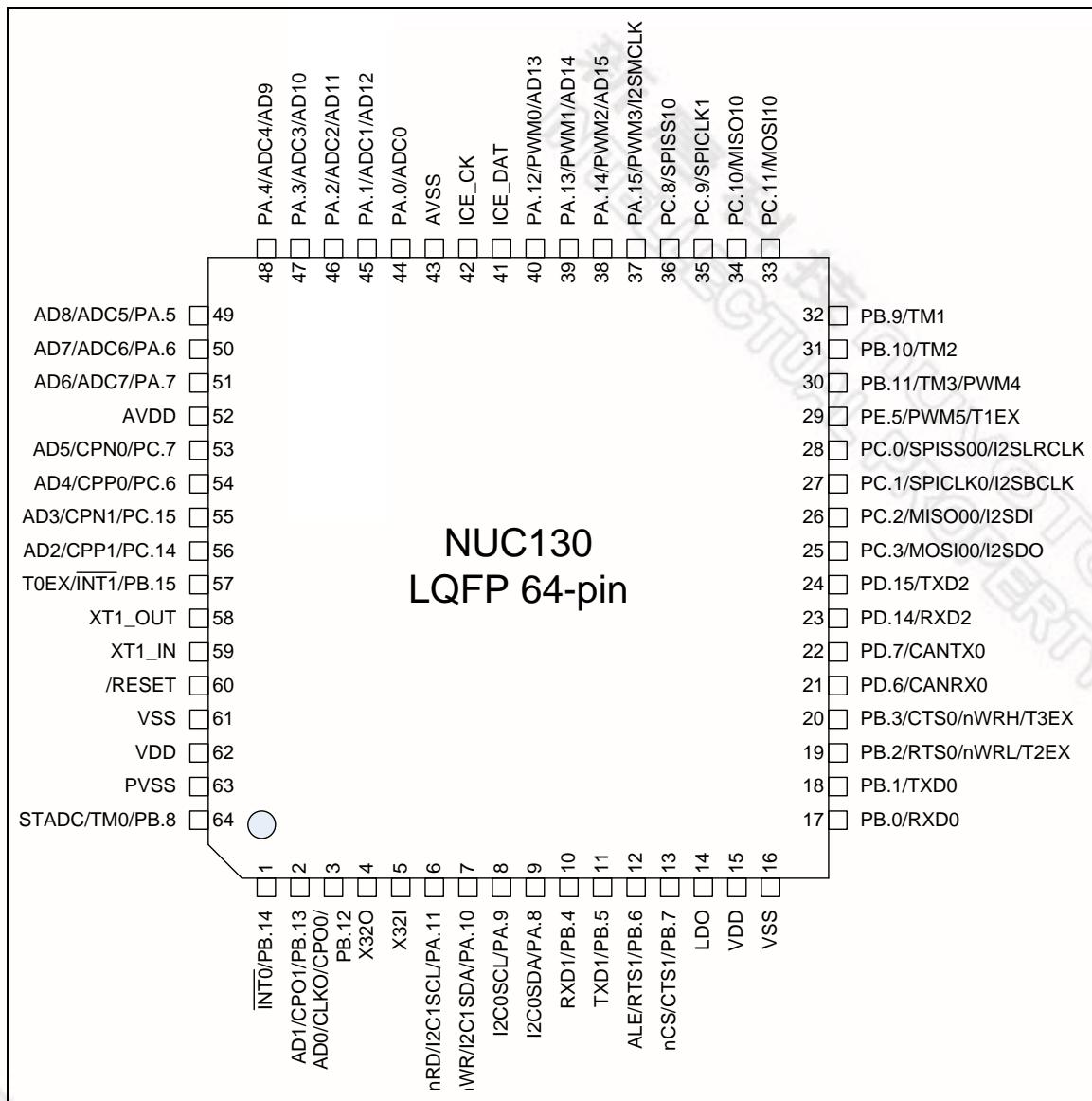


图 3-3 NuMicro™ NUC130 LQFP 64-pin 管脚图

3.3.1.3 NuMicro™ NUC130 LQFP 48 pin

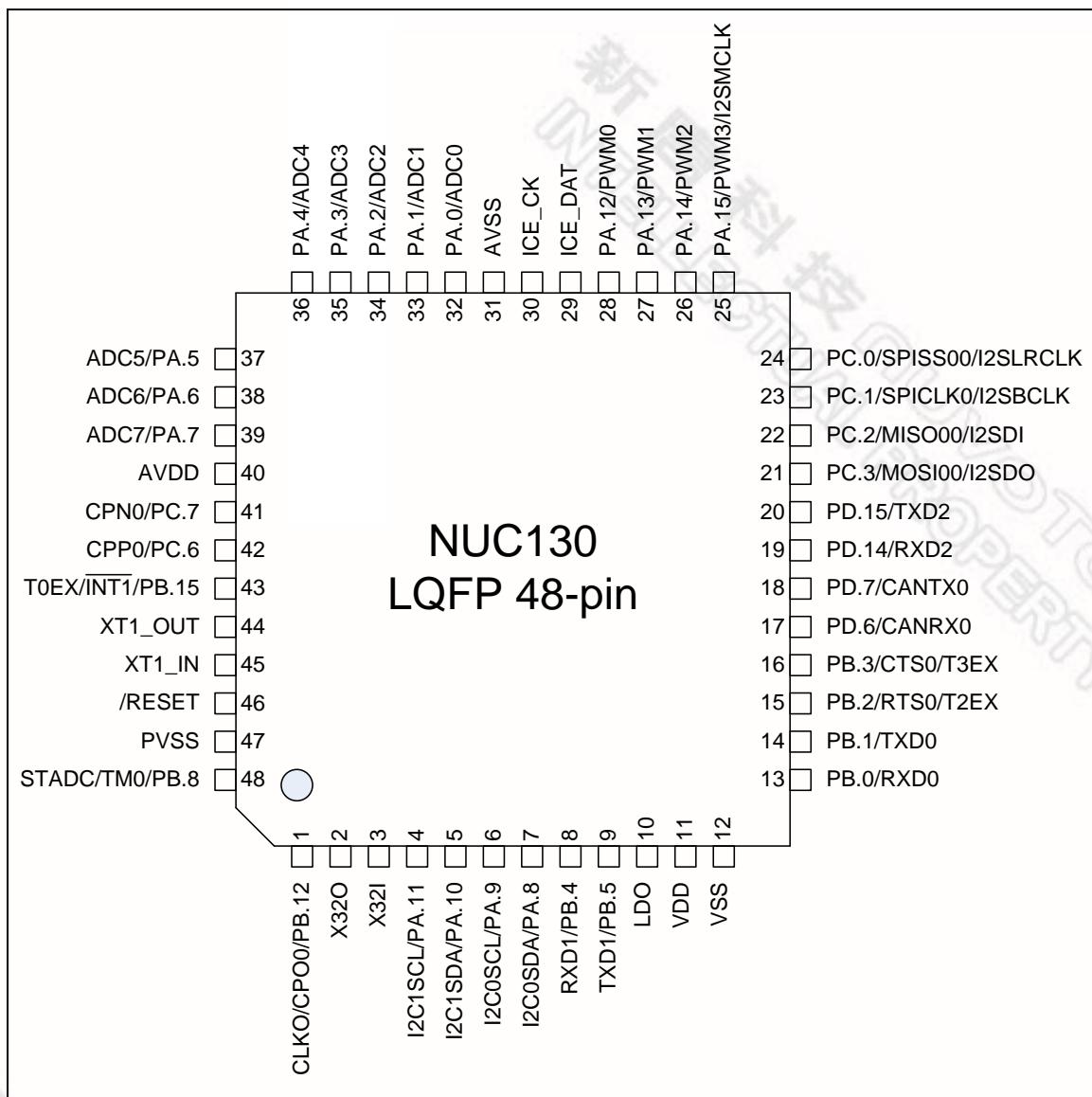


图 3-4 NuMicro™ NUC130 LQFP 48-pin 管脚图

3.3.1.4 NuMicro™ NUC140 LQFP 100 pin

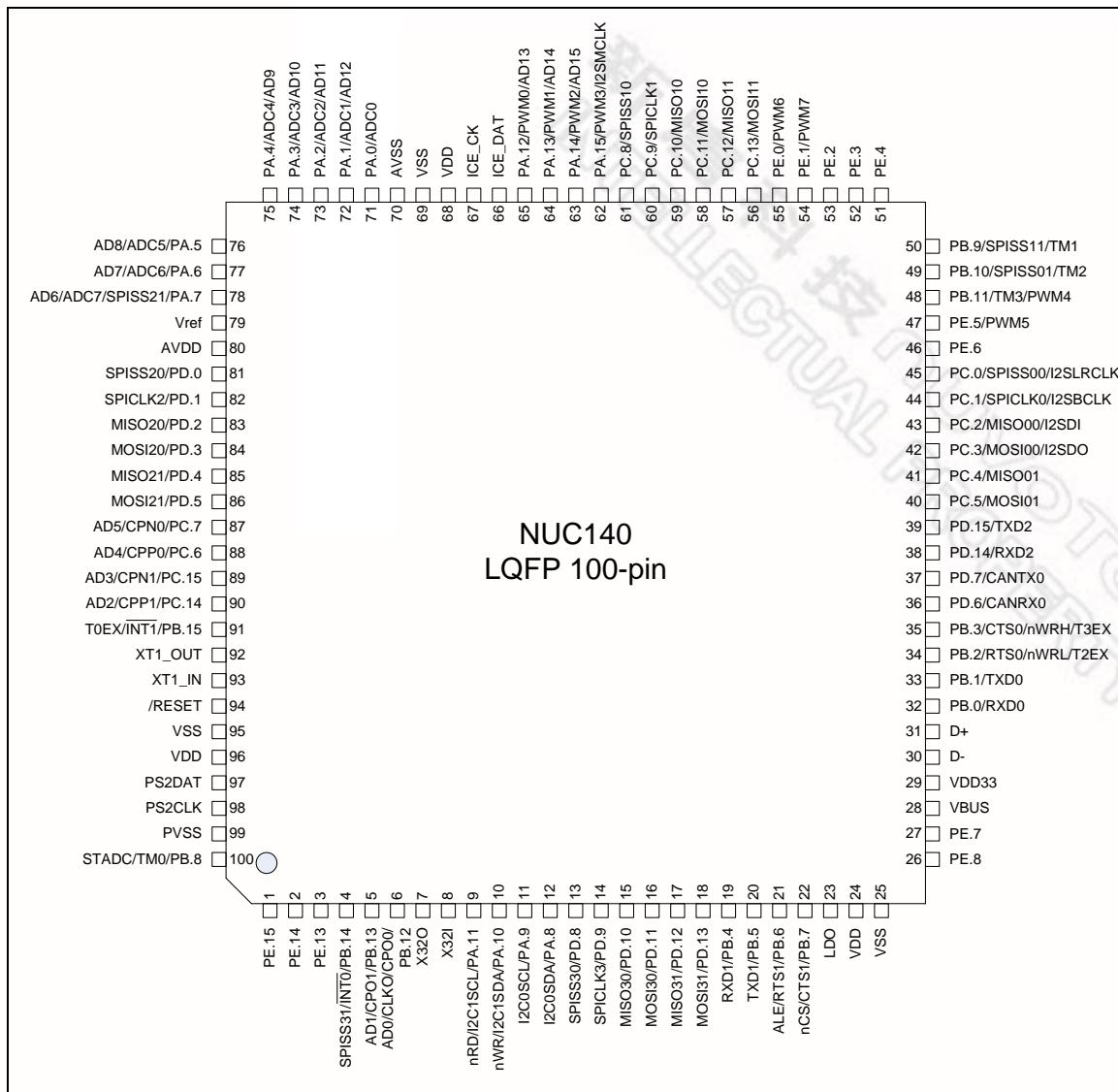


图 3-5 NuMicro™ NUC140 LQFP 100-pin 管脚图

3.3.1.5 NuMicro™ NUC140 LQFP 64 pin

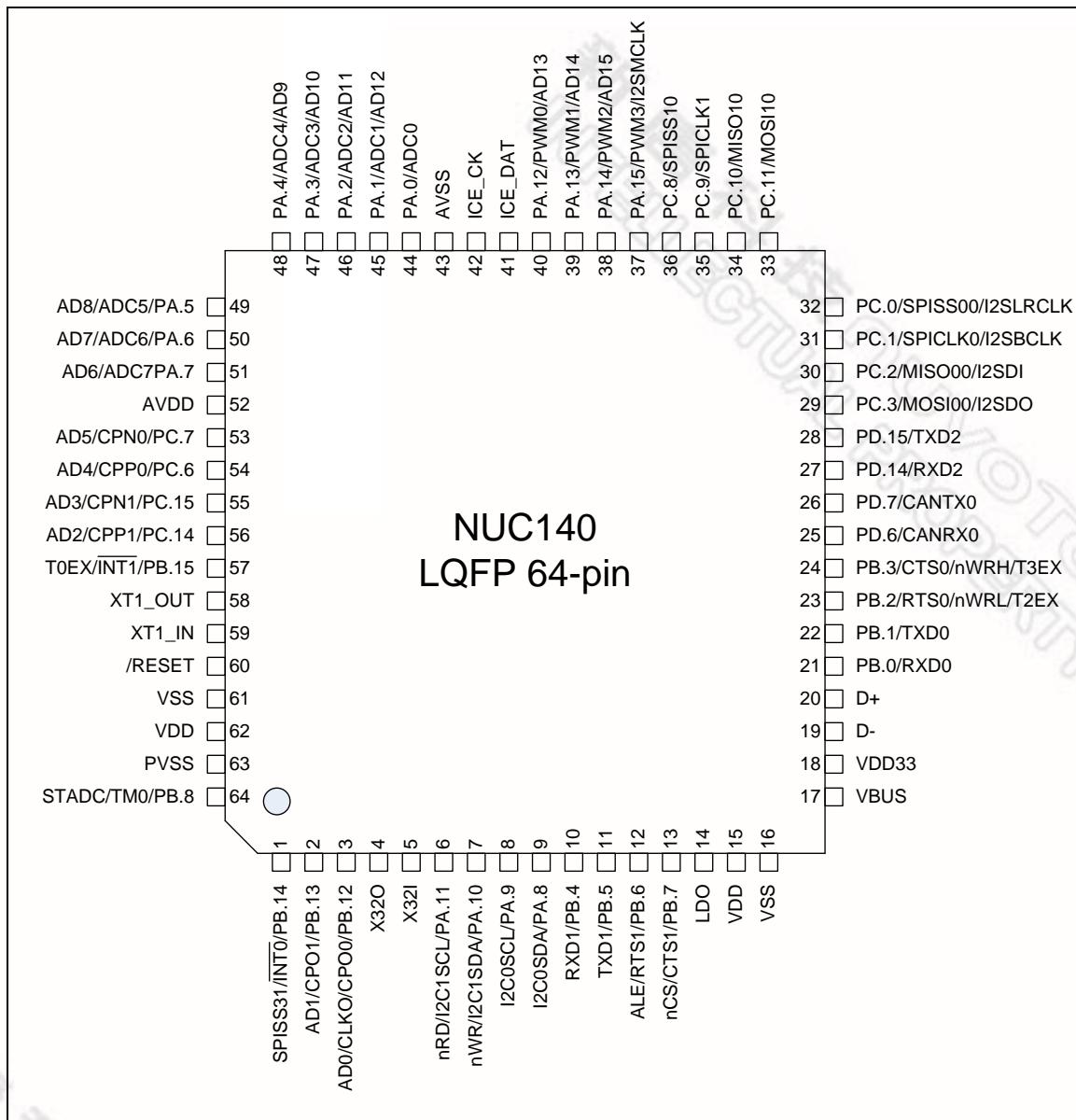


图 3-6 NuMicro™ NUC140 LQFP 64-pin 管脚图

3.3.1.6 NuMicro™ NUC140 LQFP 48 pin

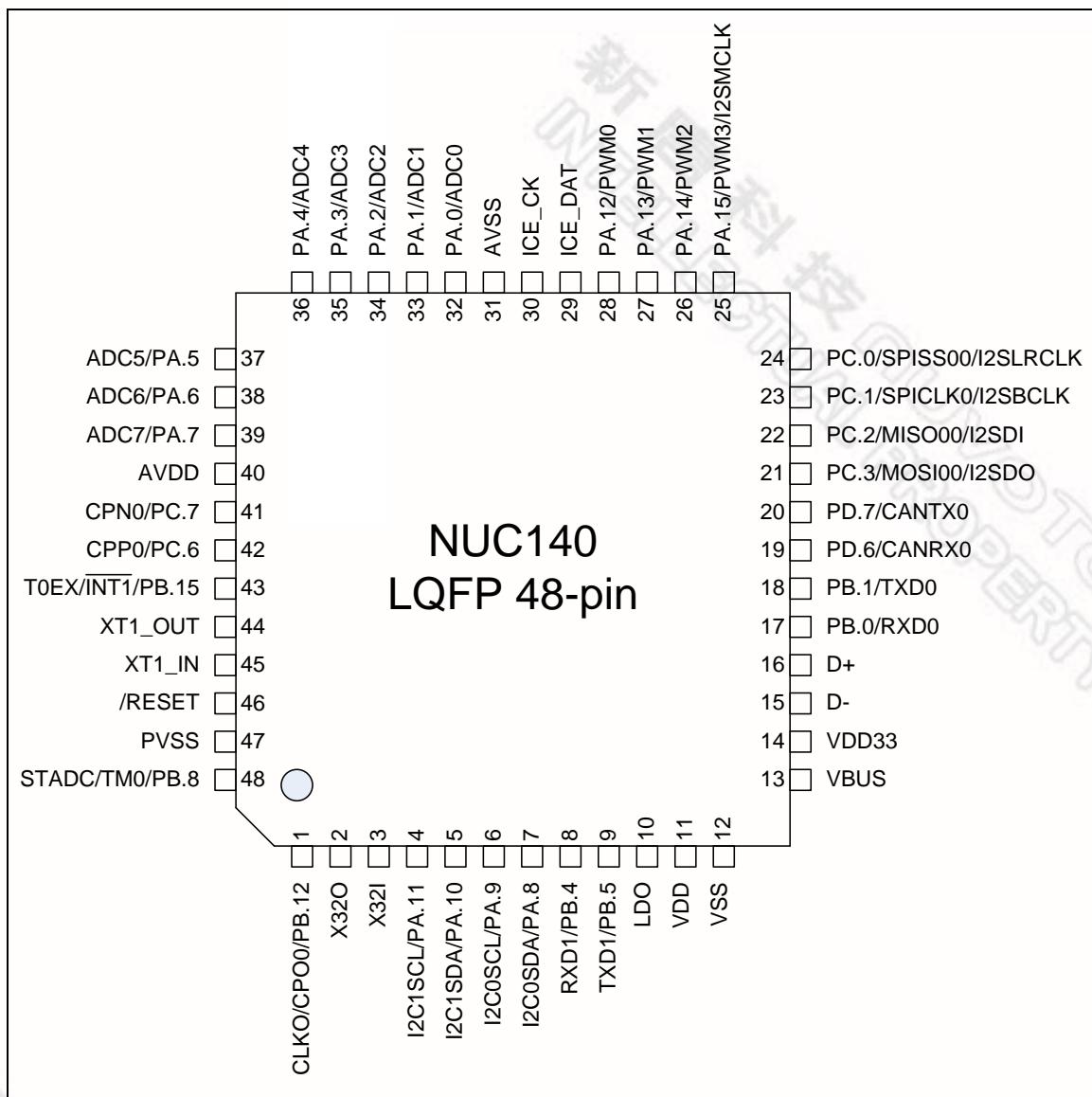


图 3-7 NuMicro™ NUC140 LQFP 48-pin 管脚图

3.4 管脚功能描述

3.4.1 NuMicro™ NUC130/NUC140 管脚定义

3.4.1.1 NuMicro™ NUC130 管脚定义

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
1			PE.15	I/O		通用数字输入/输出管脚
2			PE.14	I/O		通用数字输入/输出管脚
3			PE.13	I/O		通用数字输入/输出管脚
4	1		PB.14	I/O		通用数字输入/输出管脚
			/INT0	I		/INT0: 外部中断1输入管脚
			SPISS31	I/O		SPISS31: SPI3 2 nd 从机选择管脚
5	2		PB.13	I/O		通用数字输入/输出管脚
			CPO1	O		Comparator1 输出管脚
			AD1	I/O		EBI 地址/数据总线 bit1
6	3	1	PB.12	I/O		通用数字输入/输出管脚
			CPO0	O		Comparator0 输出管脚
			CLKO	O		分频器输出管脚
			AD0	I/O		EBI 地址/数据总线 bit0
7	4	2	X32O	O		外部 32.768 kHz 晶振输出管脚
8	5	3	X32I	I		外部 32.768 kHz 晶振输入管脚
9	6	4	PA.11	I/O		通用数字输入/输出管脚
			I2C1SCL	I/O		I2C1SCL: I ² C1 时钟管脚
			nRD	O		EBI 读使能输出管脚
10	7	5	PA.10	I/O		通用数字输入/输出管脚
			I2C1SDA	I/O		I2C1SDA: I ² C1 数据输入/输出管脚
			nWR	O		EBI 写使能输出管脚
11	8	6	PA.9	I/O		通用数字输入/输出管脚
			I2C0SCL	I/O		I2C0SCL: I ² C0 时钟管脚
12	9	7	PA.8	I/O		通用数字输入/输出管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
			I2C0SDA	I/O		I2C0SDA: I ² C0 数据输入/输出管脚
13			PD.8	I/O		通用数字输入/输出管脚
			SPISS30	I/O		SPISS30: SPI3 从机选择管脚
14			PD.9	I/O		通用数字输入/输出管脚
			SPICLK3	I/O		SPICLK3: SPI3 串行时钟管脚
15			PD.10	I/O		通用数字输入/输出管脚
			MISO30	I/O		MISO30: SPI3 MISO (主机输入, 从机输出) 脚
16			PD.11	I/O		通用数字输入/输出管脚
			MOSI30	I/O		MOSI30: SPI3 MOSI (主机输出, 从机输入) 脚
17			PD.12	I/O		通用数字输入/输出管脚
			MISO31	I/O		MISO31: SPI3 2 nd MISO (主机输入, 从机输出) 脚
18			PD.13	I/O		通用数字输入/输出管脚
			MOSI31	I/O		MOSI31: SPI3 2 nd MOSI (主机输出, 从机输入) 脚
19	10	8	PB.4	I/O		通用数字输入/输出管脚
			RXD1	I		RXD1: UART1 数据接收输入管脚
20	11	9	PB.5	I/O		通用数字输入/输出管脚
			TXD1	O		TXD1: UART1 数据发送输出管脚
21	12		PB.6	I/O		通用数字输入/输出管脚
			RTS1	O		RTS1: UART1 请求发送输出管脚
			ALE	O		EBI 地址锁存使能输出管脚
22	13		PB.7	I/O		通用数字输入/输出管脚
			CTS1	I		CTS1: UART1 清发送输出管脚
			nCS	O		EBI 片选使能输出管脚
23	14	10	LDO	P		LDO 输出管脚
24	15	11	VDD	P		电源供应管脚, 为IO端口、内部PLL电路LDO源和数字功能提供电源
25	16	12	VSS	P		地
26			PE.12	I/O		通用数字输入/输出管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
27			PE.11	I/O		通用数字输入/输出管脚
28			PE.10	I/O		通用数字输入/输出管脚
29			PE.9	I/O		通用数字输入/输出管脚
30			PE.8	I/O		通用数字输入/输出管脚
31			PE.7	I/O		通用数字输入/输出管脚
32	17	13	PB.0	I/O		通用数字输入/输出管脚
			RXD0	I		RXD0: UART0 数据接收输入管脚
33	18	14	PB.1	I/O		通用数字输入/输出管脚
			TXD0	O		TXD0: UART0 数据发送输出管脚
34	19	15	PB.2	I/O		通用数字输入/输出管脚
			RTS0	O		RTS0: UART0 请求发送输出管脚
			nWR _L	O		EBI 低字节写使能输出管脚
			T2EX	I		Timer2 外部捕捉输入管脚
35	20	16	PB.3	I/O		通用数字输入/输出管脚
			CTS0	I		CTS0: UART0 清发送输入管脚
			nWR _H	O		EBI 高字节写使能输出管脚
			T3EX	I		Timer3 外部捕捉输入管脚
36	21	17	PD.6	I/O		通用数字输入/输出管脚
			CANRX0	I		CAN Bus0 RX 输入
37	22	18	PD.7	I/O		通用数字输入/输出管脚
			CANTX0	O		CAN Bus0 TX 输出
38	23	19	PD.14	I/O		通用数字输入/输出管脚
			RXD2	I		RXD2: UART2 数据接收输入管脚
39	24	20	PD.15	I/O		通用数字输入/输出管脚
			TXD2	O		TXD2: UART2 数据发送输出管脚
40			PC.5	I/O		通用数字输入/输出管脚
			MOSI01	I/O		MOSI01: SPI0 2 nd MOSI (主机输出, 从机输入) 脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
41			PC.4	I/O		通用数字输入/输出管脚
			MISO01	I/O		MISO01: SPI0 2 nd MISO (主机输入, 从机输出) 脚
42	25	21	PC.3	I/O		通用数字输入/输出管脚
			MOSI00	I/O		MOSI00: SPI0 MOSI (主机输出, 从机输入) 脚
			I2SDO	O		I2SDO: I ² S 数据输出
43	26	22	PC.2	I/O		通用数字输入/输出管脚
			MISO00	I/O		MISO00: SPI0 MISO (主机输入, 从机输出) 脚
			I2SDI	I		I2SDI: I ² S 数据输入
44	27	23	PC.1	I/O		通用数字输入/输出管脚
			SPICLK0	I/O		SPICLK0: SPI0 串行时钟管脚
			I2SBCLK	I/O		I2SBCLK: I ² S bit 时钟管脚
45	28	24	PC.0	I/O		通用数字输入/输出管脚
			SPISS00	I/O		SPISS00: SPI0 从机选择管脚
			I2SLRCLK	I/O		I2SLRCLK: I ² S 左右声道时钟
46			PE.6	I/O		通用数字输入/输出管脚
47	29		PE.5	I/O		通用数字输入/输出管脚
			PWM5	I/O		PWM5: PWM 输出/Capture 输入
			T1EX	I		Timer1 外部捕捉输入管脚
48	30		PB.11	I/O		通用数字输入/输出管脚
			TM3	I/O		TM3: Timer3 事件计数输入/切换输出
			PWM4	I/O		PWM4: PWM 输出/Capture 输入
49	31		PB.10	I/O		通用数字输入/输出管脚
			TM2	I/O		TM2: Timer2 事件计数输入/切换输出
			SPISS01	I/O		SPISS01: SPI0 2 nd 从机选择管脚
50	32		PB.9	I/O		通用数字输入/输出管脚
			TM1	I/O		TM1: Timer1 事件计数输入/切换输出
			SPISS11	I/O		SPISS11: SPI1 2 nd 从机选择管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
51			PE.4	I/O		通用数字输入/输出管脚
52			PE.3	I/O		通用数字输入/输出管脚
53			PE.2	I/O		通用数字输入/输出管脚
54			PE.1	I/O		通用数字输入/输出管脚
			PWM7	I/O		PWM7: PWM 输出/Capture 输入
55			PE.0	I/O		通用数字输入/输出管脚
			PWM6	I/O		PWM6: PWM 输出/Capture 输入
56			PC.13	I/O		通用数字输入/输出管脚
			MOSI11	I/O		MOSI11: SPI1 2 nd MOSI (主机输出, 从机输入) 脚
57			PC.12	I/O		通用数字输入/输出管脚
			MISO11	I/O		MISO11: SPI1 2 nd MISO (主机输入, 从机输出) 脚
58	33		PC.11	I/O		通用数字输入/输出管脚
			MOSI10	I/O		MOSI10: SPI1 MOSI (主机输出, 从机输入) 脚
59	34		PC.10	I/O		通用数字输入/输出管脚
			MISO10	I/O		MISO10: SPI1 MISO (主机输入, 从机输出) 脚
60	35		PC.9	I/O		通用数字输入/输出管脚
			SPICLK1	I/O		SPICLK1: SPI1 串行时钟管脚
61	36		PC.8	I/O		通用数字输入/输出管脚
			SPISS10	I/O		SPISS10: SPI1 从机选择管脚
			MCLK	O		EBI 时钟输出
62	37	25	PA.15	I/O		通用数字输入/输出管脚
			PWM3	I/O		PWM3: PWM 输出/Capture 输入
			I2SMCLK	O		I2SMCLK: I ² S 主机时钟输出管脚
63	38	26	PA.14	I/O		通用数字输入/输出管脚
			PWM2	I/O		PWM2: PWM 输出/Capture 输入
			AD15	I/O		EBI 地址/数据总线 bit15
64	39	27	PA.13	I/O		通用数字输入/输出管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
			PWM1	I/O		PWM1: PWM 输出/Capture 输入
			AD14	I/O		EBI 地址/数据总线 bit14
65	40	28	PA.12	I/O		通用数字输入/输出管脚
			PWM0	I/O		PWM0: PWM 输出/Capture 输入
			AD13	I/O		EBI 地址/数据总线 bit13
66	41	29	ICE_DAT	I/O		调试器的串行数据管脚
67	42	30	ICE_CK	I		调试器的串行时钟管脚
68			VDD	P		电源供应管脚，为IO端口、内部PLL电路LDO源和数字电路提供电源
69			VSS	P		地
70	43	31	AVSS	AP		模拟电路地
71	44	32	PA.0	I/O		通用数字输入/输出管脚
			ADC0	AI		ADC0: ADC 模拟输入
72	45	33	PA.1	I/O		通用数字输入/输出管脚
			ADC1	AI		ADC1: ADC 模拟输入
			AD12	I/O		EBI 地址/数据总线 bit12
73	46	34	PA.2	I/O		通用数字输入/输出管脚
			ADC2	AI		ADC2: ADC 模拟输入
			AD11	I/O		EBI 地址/数据总线 bit11
74	47	35	PA.3	I/O		通用数字输入/输出管脚
			ADC3	AI		ADC3: ADC 模拟输入
			AD10	I/O		EBI 地址/数据总线 bit10
75	48	36	PA.4	I/O		通用数字输入/输出管脚
			ADC4	AI		ADC4: ADC 模拟输入
			AD9	I/O		EBI 地址/数据总线 bit9
76	49	37	PA.5	I/O		通用数字输入/输出管脚
			ADC5	AI		ADC5: ADC 模拟输入
			AD8	I/O		EBI 地址/数据总线 bit8

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
77	50	38	PA.6	I/O		通用数字输入/输出管脚
			ADC6	AI		ADC6: ADC 模拟输入
			AD7	I/O		EBI 地址/数据总线 bit7
78	51	39	PA.7	I/O		通用数字输入/输出管脚
			ADC7	AI		ADC7: ADC 模拟输入
			SPISS21	I/O		SPISS21: SPI2 2 nd 从机选择管脚
			AD6	I/O		EBI 地址/数据总线 bit6
79			VREF	AP		ADC 参考电压输入
80	52	40	AVDD	AP		内部模拟电路电源
81			PD.0	I/O		通用数字输入/输出管脚
			SPISS20	I/O		SPISS20: SPI2 从机选择管脚
82			PD.1	I/O		通用数字输入/输出管脚
			SPICLK2	I/O		SPICLK2: SPI2 串行时钟管脚
83			PD.2	I/O		通用数字输入/输出管脚
			MISO20	I/O		MISO20: SPI2 MISO (主机输入, 从机输出) 脚
84			PD.3	I/O		通用数字输入/输出管脚
			MOSI20	I/O		MOSI20: SPI2 MOSI (主机输出, 从机输入) 脚
85			PD.4	I/O		通用数字输入/输出管脚
			MISO21	I/O		MISO21: SPI2 2 nd MISO (主机输入, 从机输出) 脚
86			PD.5	I/O		通用数字输入/输出管脚
			MOSI21	I/O		MOSI21: SPI2 2 nd MOSI (主机输出, 从机输入) 脚
87	53	41	PC.7	I/O		通用数字输入/输出管脚
			CPN0	AI		CPN0: Comparator0 负端输入管脚
			AD5	I/O		EBI 地址/数据总线 bit5
88	54	42	PC.6	I/O		通用数字输入/输出管脚
			CPP0	AI		CPP0: Comparator0 正端输入管脚
			AD4	I/O		EBI 地址/数据总线 bit4
89	55		PC.15	I/O		通用数字输入/输出管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
			CPN1	AI		CPN1: Comparator1 负端输入管脚
			AD3	I/O		EBI 地址/数据总线 bit3
90	56		PC.14	I/O		通用数字输入/输出管脚
			CPP1	AI		CPP1: Comparator1 正端输入管脚
			AD2	I/O		EBI 地址/数据总线 bit2
91	57	43	PB.15	I/O		通用数字输入/输出管脚
			/INT1	I		/INT1: 外部中断0输入管脚
			TOEX	I		Timer0 外部捕捉输入
92	58	44	XT1_OUT	O		外部 4~24 MHz 晶振输出管脚
93	59	45	XT1_IN	I		外部 4~24 MHz 晶振输入管脚
94	60	46	/RESET	I		外部复位输入: 低有效, 置低复位MCU为初始状态, 带内部上拉。
95	61		VSS	P		地
96	62		VDD	P		电源供应引脚, 为IO端口、内部PLL电路LDO源和数字电路提供电源
97			PS2DAT	I/O		PS/2 数据管脚
98			PS2CLK	I/O		PS/2 时钟管脚
99	63	47	PVSS	P		PLL 地
100	64	48	PB.8	I/O		通用数字输入/输出管脚
			STADC	I		STADC: ADC 外部触发输入.
			TM0	I/O		TM0: Timer0 事件计数输入/切换输出

注: 管脚类型 I = 数字输入 (Digital Input), O = 数字输出 (Digital Output); AI= 模拟输入 (Analog Input); P=电源管脚 (Power Pin); AP= 模拟电源 (Analog Power)

3.4.1.2 NuMicro™ NUC140 管脚定义

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
1			PE.15	I/O		通用数字输入/输出管脚
2			PE.14	I/O		通用数字输入/输出管脚
3			PE.13	I/O		通用数字输入/输出管脚
4	1		PB.14	I/O		通用数字输入/输出管脚
			/INT0	I		/INT0: 外部中断1 输入管脚
			SPISS31	I/O		SPISS31: SPI3 2 nd 从机选择管脚
5	2		PB.13	I/O		通用数字输入/输出管脚
			CPO1	O		Comparator1 输出管脚
			AD1	IO		EBI 地址/数据总线 bit1
6	3	1	PB.12	I/O		通用数字输入/输出管脚
			CPO0	O		Comparator0 输出管脚
			CLK0	O		分频器输出管脚
			AD0	I/O		EBI 地址/数据总线 bit0
7	4	2	X32O	O		外部 32.768 kHz 晶振输出管脚
8	5	3	X32I	I		外部 32.768 kHz 晶振输入管脚
9	6	4	PA.11	I/O		通用数字输入/输出管脚
			I2C1SCL	I/O		I2C1SCL: I ² C1 时钟管脚
			nRD	O		EBI 读使能输出管脚
10	7	5	PA.10	I/O		通用数字输入/输出管脚
			I2C1SDA	I/O		I2C1SDA: I ² C1 数据输入/输出管脚
			nWR	O		EBI 写使能输出管脚
11	8	6	PA.9	I/O		通用数字输入/输出管脚
			I2C0SCL	I/O		I2C0SCL: I ² C0 时钟管脚
12	9	7	PA.8	I/O		通用数字输入/输出管脚
			I2C0SDA	I/O		I2C0SDA: I ² C0 数据输入/输出管脚
13			PD.8	I/O		通用数字输入/输出管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
			SPISS30	I/O		SPISS30: SPI3 从机选择管脚
14			PD.9	I/O		通用数字输入/输出管脚
			SPICLK3	I/O		SPICLK3: SPI3 串行时钟管脚
15			PD.10	I/O		通用数字输入/输出管脚
			MISO30	I/O		MISO30: SPI3 MISO (主机输入, 从机输出) 脚
16			PD.11	I/O		通用数字输入/输出管脚
			MOSI30	I/O		MOSI30: SPI3 MOSI (主机输出, 从机输入) 脚
17			PD.12	I/O		通用数字输入/输出管脚
			MISO31	I/O		MISO31: SPI3 2 nd MISO (主机输入, 从机输出) 脚
18			PD.13	I/O		通用数字输入/输出管脚
			MOSI31	I/O		MOSI31: SPI3 2 nd MOSI (主机输出, 从机输入) 脚
19	10	8	PB.4	I/O		通用数字输入/输出管脚
			RXD1	I		RXD1: UART1 数据接收输入管脚
20	11	9	PB.5	I/O		通用数字输入/输出管脚
			TXD1	O		TXD1: UART1 数据发送输出管脚
21	12		PB.6	I/O		通用数字输入/输出管脚
			RTS1	O		RTS1: UART1 请求发送输出管脚
			ALE	O		EBI 地址锁存使能输出管脚
22	13		PB.7	I/O		通用数字输入/输出管脚
			CTS1	I		CTS1: UART1 清发送输入管脚
			nCS	O		EBI 片选使能输出管脚
23	14	10	LDO	P		LDO 输出管脚
24	15	11	VDD	P		电源供应管脚, 为IO端口、内部PLL电路LDO源和数字功能提供电源
25	16	12	VSS	P		地
26			PE.8	I/O		通用数字输入/输出管脚
27			PE.7	I/O		通用数字输入/输出管脚
28	17	13	VBUS	USB		USB HOST 或 HUB提供电源管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
29	18	14	VDD33	USB		内部3.3V电压输出管脚
30	19	15	D-	USB		USB 差分信号 D-
31	20	16	D+	USB		USB 差分信号 D+
32	21	17	PB.0	I/O		通用数字输入/输出管脚
			RXD0	I		RXD0: UART0 数据接收输入管脚
33	22	18	PB.1	I/O		通用数字输入/输出管脚
			TXD0	O		TXD0: UART0 数据发送输出管脚
34	23		PB.2	I/O		通用数字输入/输出管脚
			RTS0	O		RTS0: UART0 请求发送输出管脚
			nWRL	O		EBI 低字节写使能输出管脚
			T2EX	I		Timer2 外部捕捉输入管脚
35	24		PB.3	I/O		通用数字输入/输出管脚
			CTS0	I		CTS0: UART0 清发送输入管脚
			nWRH	O		EBI 高字节写使能输出管脚
			T3EX	I		Timer3 外部捕捉输入管脚
36	25	19	PD.6	I/O		通用数字输入/输出管脚
			CANRX0	I		CAN Bus0 RX 输入
37	26	20	PD.7	I/O		通用数字输入/输出管脚
			CANTX0	O		CAN Bus0 TX 输出
38	27		PD.14	I/O		通用数字输入/输出管脚
			RXD2	I		RXD2: UART2 数据接收输入管脚
39	28		PD.15	I/O		通用数字输入/输出管脚
			TXD2	O		TXD2: UART2 数据发送输出管脚
40			PC.5	I/O		通用数字输入/输出管脚
			MOSI01	I/O		MOSI01: SPI0 2 nd MOSI (主机输出, 从机输入) 脚
41			PC.4	I/O		通用数字输入/输出管脚
			MISO01	I/O		MISO01: SPI0 2 nd MISO (主机输入, 从机输出) 脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
42	29	21	PC.3	I/O		通用数字输入/输出管脚
			MOSI00	I/O		MOSI00: SPI0 MOSI (主机输出, 从机输入) 脚
			I2SDO	O		I2SDO: I ² S 数据输出
43	30	22	PC.2	I/O		通用数字输入/输出管脚
			MISO00	I/O		MISO00: SPI0 MISO (主机输入, 从机输出) 脚
			I2SDI	I		I2SDI: I ² S 数据输入
44	31	23	PC.1	I/O		通用数字输入/输出管脚
			SPICLK0	I/O		SPICLK0: SPI0 串行时钟管脚
			I2SBCLK	I/O		I2SBCLK: I ² S bit 时钟管脚
45	32	24	PC.0	I/O		通用数字输入/输出管脚
			SPISS00	I/O		SPISS00: SPI0 从机选择管脚
			I2SLRCLK	I/O		I2SLRCLK: I ² S 左右声道时钟
46			PE.6	I/O		通用数字输入/输出管脚
47			PE.5	I/O		通用数字输入/输出管脚
			PWM5	I/O		PWM5: PWM 输出/Capture 输入
			T1EX	I		Timer1 外部捕捉输入管脚
48			PB.11	I/O		通用数字输入/输出管脚
			TM3	I/O		TM3: Timer3 事件计数输入/切换输出
			PWM4	I/O		PWM4: PWM 输出/Capture 输入
49			PB.10	I/O		通用数字输入/输出管脚
			TM2	I/O		TM2: Timer2 事件计数输入/切换输出
			SPISS01	I/O		SPISS01: SPI0 2 nd 从机选择管脚
50			PB.9	I/O		通用数字输入/输出管脚
			TM1	I/O		TM1: Timer1 事件计数输入/切换输出
			SPISS11	I/O		SPISS11: SPI1 2 nd 从机选择管脚
51			PE.4	I/O		通用数字输入/输出管脚
52			PE.3	I/O		通用数字输入/输出管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
53			PE.2	I/O		通用数字输入/输出管脚
54			PE.1	I/O		通用数字输入/输出管脚
			PWM7	I/O		PWM7: PWM 输出/Capture 输入
55			PE.0	I/O		通用数字输入/输出管脚
			PWM6	I/O		PWM6: PWM 输出/Capture 输入
56			PC.13	I/O		通用数字输入/输出管脚
			MOSI11	I/O		MOSI11: SPI1 2 nd MOSI (主机输出, 从机输入) 脚
57			PC.12	I/O		通用数字输入/输出管脚
			MISO11	I/O		MISO11: SPI1 2 nd MISO (主机输入, 从机输出) 脚
58	33		PC.11	I/O		通用数字输入/输出管脚
			MOSI10	I/O		MOSI10: SPI1 MOSI (主机输出, 从机输入) 脚
59	34		PC.10	I/O		通用数字输入/输出管脚
			MISO10	I/O		MISO10: SPI1 MISO (主机输入, 从机输出) 脚
60	35		PC.9	I/O		通用数字输入/输出管脚
			SPICLK1	I/O		SPICLK1: SPI1 串行时钟管脚
61	36		PC.8	I/O		通用数字输入/输出管脚
			SPISS10	I/O		SPISS10: SPI1 从机选择管脚
			MCLK	O		EBI 时钟输出
62	37	25	PA.15	I/O		通用数字输入/输出管脚
			PWM3	I/O		PWM3: PWM 输出/Capture 输入
			I2SMCLK	O		I2SMCLK: I ² S 主机时钟输出管脚
63	38	26	PA.14	I/O		通用数字输入/输出管脚
			PWM2	I/O		PWM2: PWM 输出/Capture 输入
			AD15	I/O		EBI 地址/数据总线 bit15
64	39	27	PA.13	I/O		通用数字输入/输出管脚
			PWM1	I/O		PWM1: PWM 输出/Capture 输入
			AD14	I/O		EBI 地址/数据总线 bit14
65	40	28	PA.12	I/O		通用数字输入/输出管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
			PWM0	I/O		PWM0: PWM 输出/Capture 输入
			AD13	I/O		EBI 地址/数据总线 bit13
66	41	29	ICE_DAT	I/O		调试器的串行数据管脚
67	42	30	ICE_CK	I		调试器的串行时钟管脚
68			VDD	P		电源供应管脚，为IO端口、内部PLL电路LDO源和数字功能提供电源
69			VSS	P		地
70	43	31	AVSS	AP		模拟电路地
71	44	32	PA.0	I/O		通用数字输入/输出管脚
			ADC0	AI		ADC0: ADC 模拟输入
72	45	33	PA.1	I/O		通用数字输入/输出管脚
			ADC1	AI		ADC1: ADC 模拟输入
			AD12	I/O		EBI 地址/数据总线 bit12
73	46	34	PA.2	I/O		通用数字输入/输出管脚
			ADC2	AI		ADC2: ADC 模拟输入
			AD11	I/O		EBI 地址/数据总线 bit11
74	47	35	PA.3	I/O		通用数字输入/输出管脚
			ADC3	AI		ADC3: ADC 模拟输入
			AD10	I/O		EBI 地址/数据总线 bit10
75	48	36	PA.4	I/O		通用数字输入/输出管脚
			ADC4	AI		ADC4: ADC 模拟输入
			AD9	I/O		EBI 地址/数据总线 bit9
76	49	37	PA.5	I/O		通用数字输入/输出管脚
			ADC5	AI		ADC5: ADC 模拟输入
			AD8	I/O		EBI 地址/数据总线 bit8
77	50	38	PA.6	I/O		通用数字输入/输出管脚
			ADC6	AI		ADC6: ADC 模拟输入
			AD7	I/O		EBI 地址/数据总线 bit7

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
78	51	39	PA.7	I/O		通用数字输入/输出管脚
			ADC7	AI		ADC7: ADC 模拟输入
			SPISS21	I/O		SPISS21: SPI2 2 nd 从机选择管脚
			AD6	I/O		EBI 地址/数据总线 bit6
79			VREF	AP		ADC 参考电压输入
80	52	40	AVDD	AP		内部模拟电路电源
81			PD.0	I/O		通用数字输入/输出管脚
			SPISS20	I/O		SPISS20: SPI2 从机选择管脚
82			PD.1	I/O		通用数字输入/输出管脚
			SPICLK2	I/O		SPICLK2: SPI2 串行时钟管脚
83			PD.2	I/O		通用数字输入/输出管脚
			MISO20	I/O		MISO20: SPI2 MISO (主机输入, 从机输出) 脚
84			PD.3	I/O		通用数字输入/输出管脚
			MOSI20	I/O		MOSI20: SPI2 MOSI (主机输出, 从机输入) 脚
85			PD.4	I/O		通用数字输入/输出管脚
			MISO21	I/O		MISO21: SPI2 2 nd MISO (主机输入, 从机输出) 脚
86			PD.5	I/O		通用数字输入/输出管脚
			MOSI21	I/O		MOSI21: SPI2 2 nd MOSI (主机输出, 从机输入) 脚
87	53	41	PC.7	I/O		通用数字输入/输出管脚
			CPN0	AI		CPN0: Comparator0 负端输入管脚
			AD5	I/O		EBI 地址/数据总线 bit 5
88	54	42	PC.6	I/O		通用数字输入/输出管脚
			CPP0	AI		CPP0: Comparator0 正端输入管脚
			AD4	I/O		EBI 地址/数据总线 bit 4
89	55		PC.15	I/O		通用数字输入/输出管脚
			CPN1	AI		CPN1: Comparator1 负端输入管脚
			AD3	I/O		EBI 地址/数据总线 bit 3
90	56		PC.14	I/O		通用数字输入/输出管脚

管脚号			管脚名称	管脚类型		描述
LQFP 100	LQFP 64	LQFP 48				
			CPP1	AI		CPP1: Comparator1 正端输入管脚
			AD2	I/O		EBI 地址/数据总线 bit 2
91	57	43	PB.15	I/O		通用数字输入/输出管脚
			/INT1	I		/INT1: 外部中断0 输入管脚
			T0EX	I		Timer 0 外部捕捉输入管脚
92	58	44	XT1_OUT	O		外部 4~24 MHz 晶振输出管脚
93	59	45	XT1_IN	I		外部 4~24 MHz 晶振输出管脚
94	60	46	/RESET	I		外部复位输入: 低有效, 置低复位MCU为初始状态, 带内部上拉。
95	61		VSS	P		地
96	62		VDD	P		电源供应管脚, 为IO端口、内部PLL电路LDO源和数字功能提供电源
97			PS2DAT	I/O		PS/2 数据管脚
98			PS2CLK	I/O		PS/2 时钟管脚
99	63	47	PVSS	P		PLL 地
100	64	48	PB.8	I/O		通用数字输入/输出管脚
			STADC	I		STADC: ADC 外部触发输入
			TMO	I/O		TMO: Timer0 事件计数输入/切换输出

注: 管脚类型 I = 数字输入 (Digital Input), O = 数字输出 (Digital Output); AI= 模拟输入 (Analog Input); P=电源管脚 (Power Pin); AP= 模拟电源 (Analog Power)

4 框图

4.1 NuMicro™ NUC130/NUC140 框图

4.1.1 NuMicro™ NUC130 框图

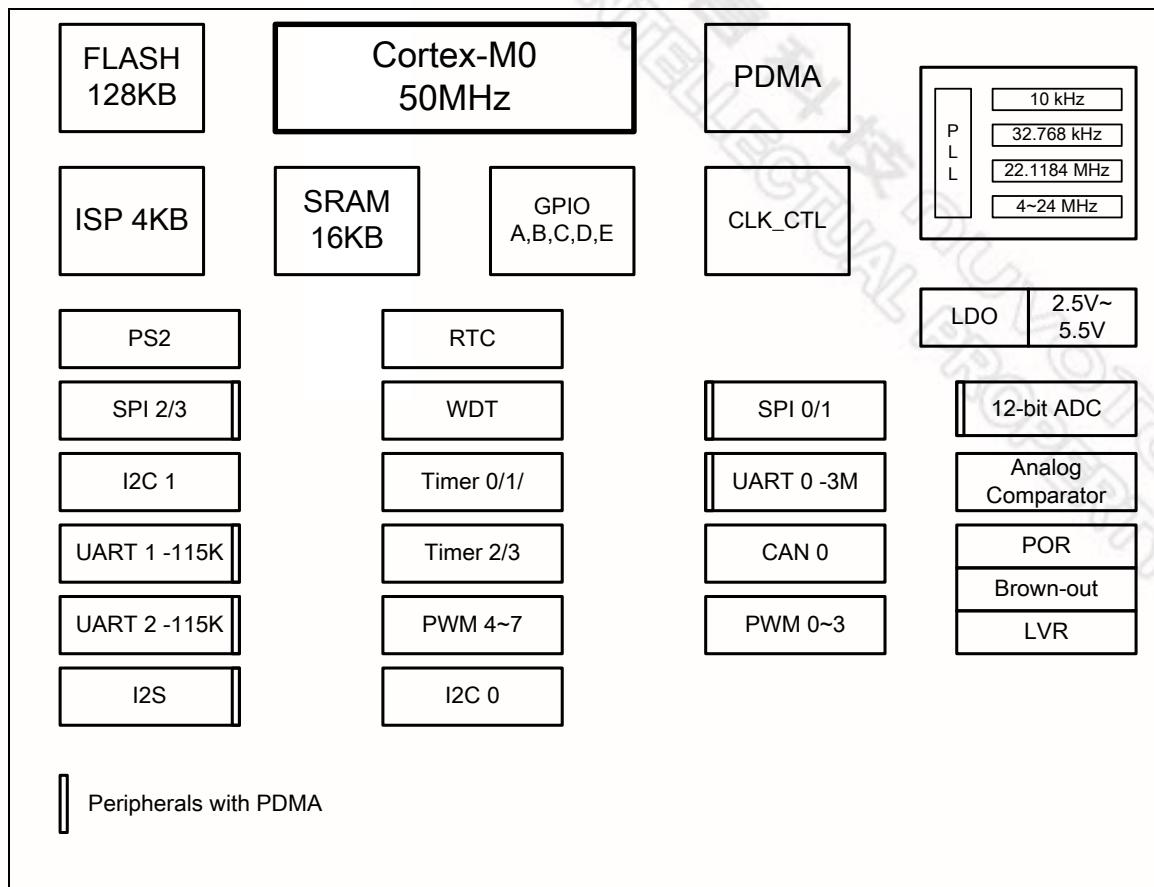


图 4-1 NuMicro™ NUC130 框图

4.1.2 NuMicro™ NUC140 框图

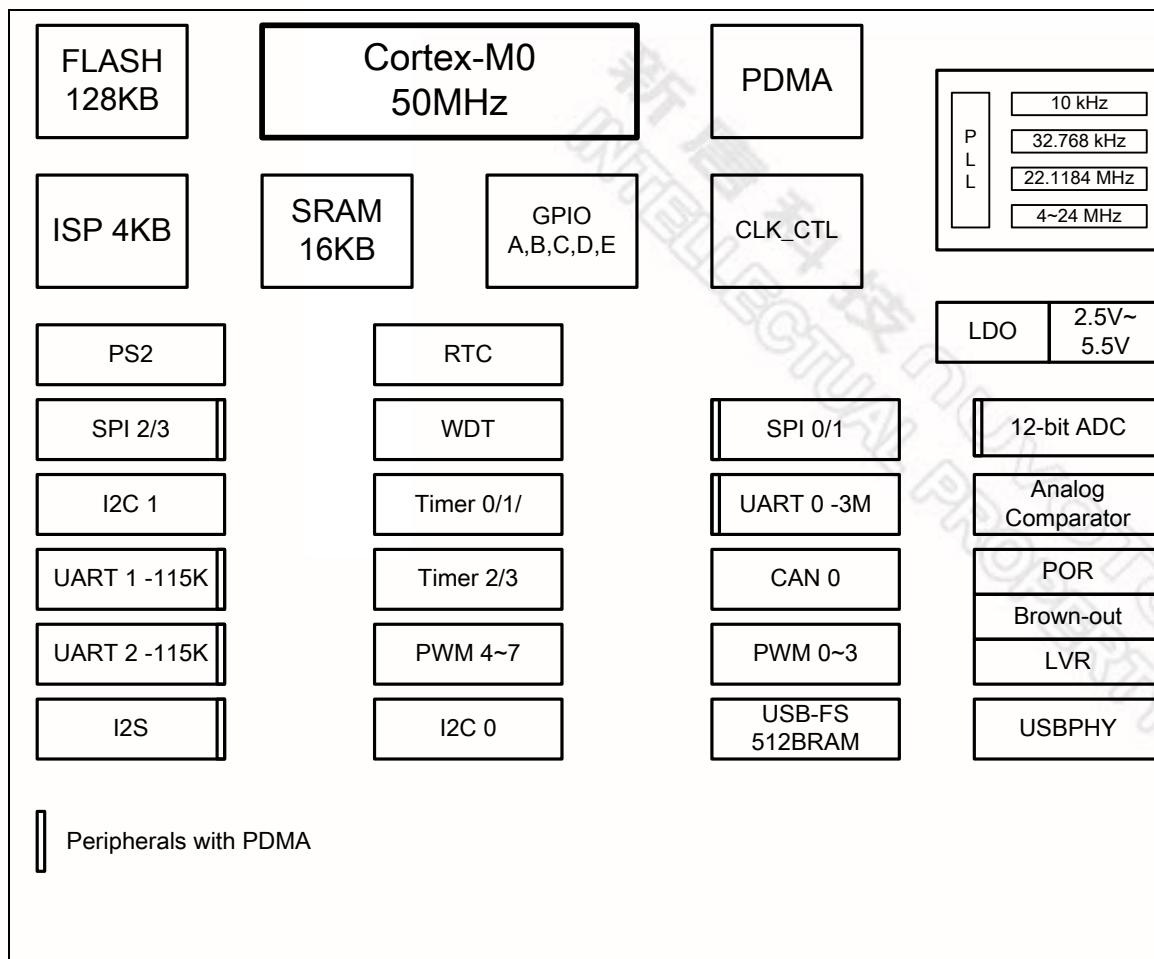


图 4-2 NuMicro™ NUC140 框图

5 功能描述

5.1 ARM® Cortex™-M0 内核

The Cortex™-M0处理器是32位可配置的多级流水线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。支持两种模式-Thread 模式与 Handler 模式。异常时系统进入 Handler 模式。从Handler 模式返回时，执行异常返回。复位时系统进入Thread 模式。Thread 模式也可由异常返回时进入。图 5-1为处理器的功能图。

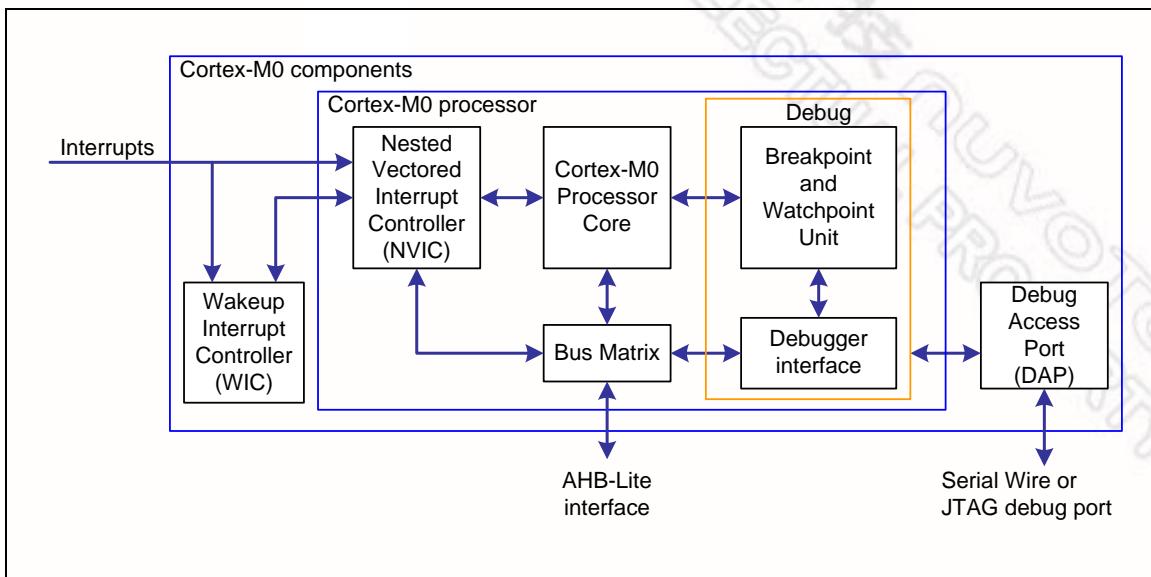


图 5-1 功能框图

设备提供：

- 低门数处理器特征：
 - ◆ 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
 - ◆ 准确而及时的中断处理能力
 - ◆ 加载、存储多个数据和多周期乘法指令可被终止然后重新开始从而实现快速中断处理
 - ◆ C 应用程序二进制接口的异常兼容模式（C-ABI）。这个 ARMv6-M 的模式允许用户使用纯C函数实现中断处理。
 - ◆ 使用中断唤醒（WFI）与事件唤醒（WFE）指令进入低功耗的休眠模式，或者从中断退出休眠模式



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



5.2 系统管理器

5.2.1 概述

系统管理器包括如下功能：

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

5.2.2 系统复位

下列任一情况发生时，系统复位，复位标志由寄存器 RSTSRC 读出。

- 上电复位
- 复位脚 (/RESET) 上有低电平
- 看门狗复位
- 低压复位
- 欠压检测器复位
- CPU 复位
- 系统复位

系统复位和上电复位使整个芯片复位，包括所有外设。系统复位与上电复位的区别在于外部晶振电路与 ISPCON.BS 位。系统复位不复位外部晶振电路和 ISPCON.BS 位，上电复位可以。

5.2.3 系统电源分配

该器件的电源分为三个部分：

- 由 AVDD 和 AVSS 提供的模拟电源，为模拟部分工作提供电压。
- 由 VDD 和 VSS 提供的数字电源，提供一个固定的 2.5V 的数字电源，用于数字操作和 I/O 引脚的内部稳压电源。
- VBUS 提供给 USB 的电源，用于 USB 模块传输操作。（仅用于 NuMicro™ NUC140）

内部电影调节器输出，LDO 和 VDD33，需要在相应的引脚上外接电容。模拟电源（AVDD）的电压电平必须和数字电源（VDD）的一样。图 5-2 为 NuMicro™ NUC140 的电源分配图，图 5-3 为 NuMicro™ NUC130 的电源分配图。

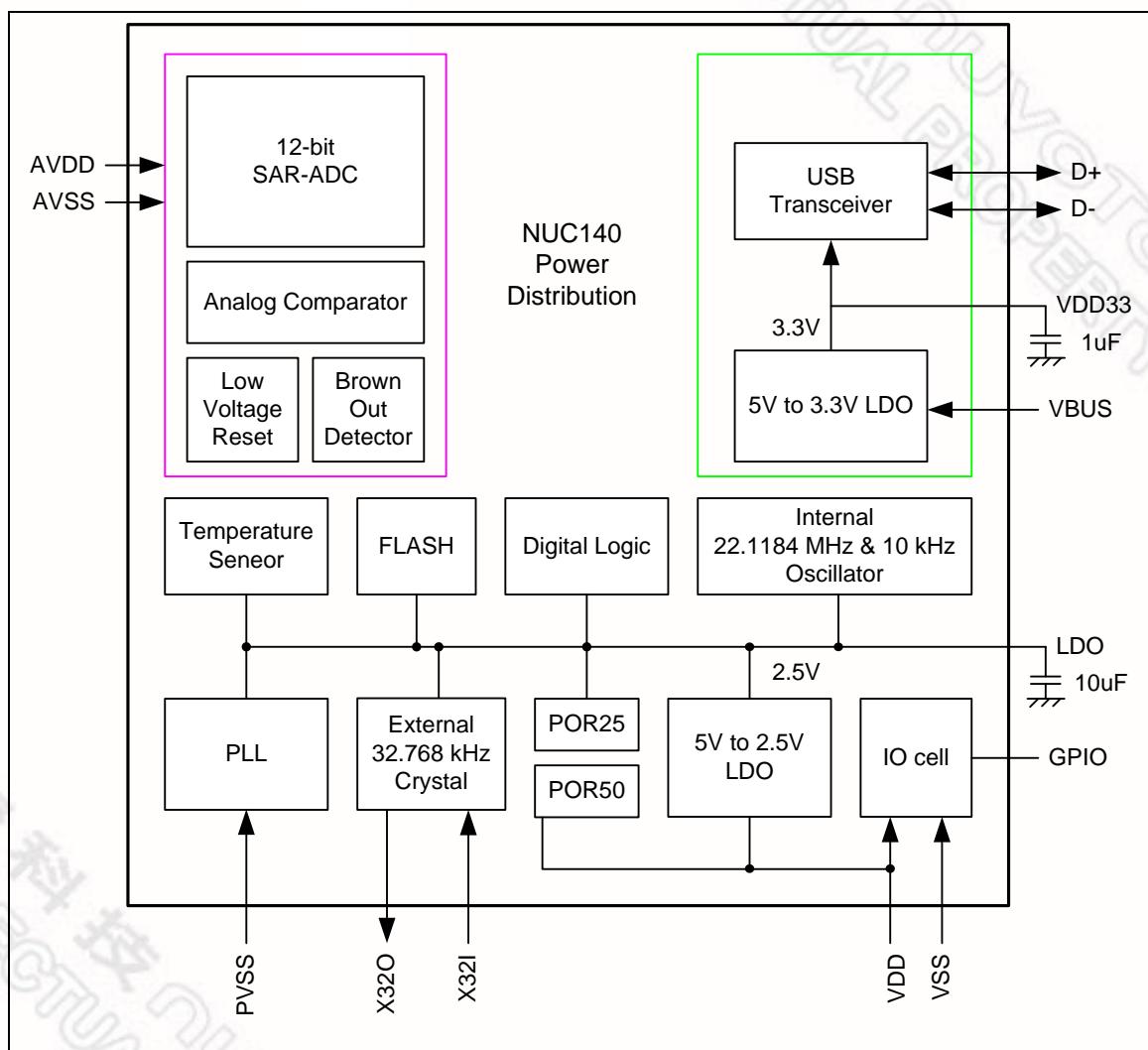


图 5-2 NuMicro™ NUC140 电源分配图

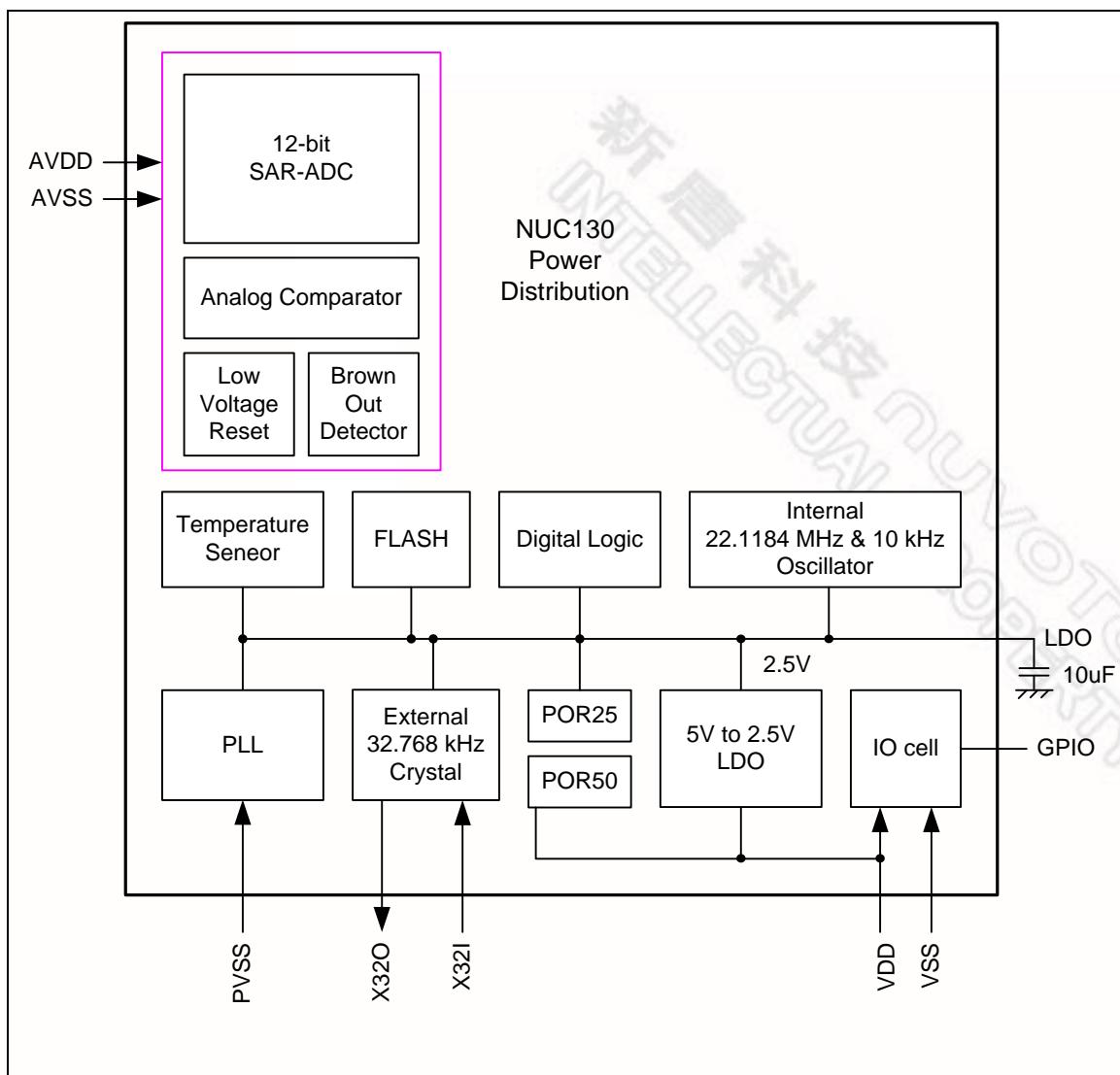


图 5-3 NuMicro™ NUC130 电源分配图

5.2.4 系统内存映射

NuMicro™ NUC100 系列提供 4G-字节的寻址空间。内存地址分配情况见下表。对各片上外设的详细的寄存器描述，内存空间，和编程指南，稍后章节将有详细描述。NuMicro™ NUC100 系列仅支持小端数据格式。

地址空间	标志	控制器
Flash & SRAM 内存空间		
0x0000_0000 – 0x0001_FFFF	FLASH_BA	FLASH 内存空间 (128KB)
0x2000_0000 – 0x2000_3FFF	SRAM_BA	SRAM 内存空间 (16KB)
0x6000_0000 – 0x6001_FFFF	EXTMEM_BA	外部存储空间 (128KB)
AHB Controllers Space (0x5000_0000 – 0x501F_FFFF)		
0x5000_0000 – 0x5000_01FF	GCR_BA	系统全局控制寄存器
0x5000_0200 – 0x5000_02FF	CLK_BA	时钟控制寄存器
0x5000_0300 – 0x5000_03FF	INT_BA	多路中断控制寄存器
0x5000_4000 – 0x5000_7FFF	GPIO_BA	GPIO 控制寄存器
0x5000_8000 – 0x5000_BFFF	PDMA_BA	外设 DMA 控制寄存器
0x5000_C000 – 0x5000_FFFF	FMC_BA	Flash 内存控制寄存器
0x5001_0000 – 0x5001_03FF	EBI_BA	外部总线接口控制寄存器
APB1 控制器空间 (0x4000_0000 ~ 0x400F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	看门狗控制寄存器
0x4000_8000 – 0x4000_BFFF	RTC_BA	RTC 控制寄存器
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 控制寄存器
0x4002_0000 – 0x4002_3FFF	I2C0_BA	I ² C0 接口控制寄存器
0x4003_0000 – 0x4003_3FFF	SPI0_BA	带主/从功能的SPI0 控制寄存器
0x4003_4000 – 0x4003_7FFF	SPI1_BA	带主/从功能的SPI1 控制寄存器
0x4004_0000 – 0x4004_3FFF	PWMA_BA	PWM0/1/2/3 控制寄存器
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0 控制寄存器
0x4006_0000 – 0x4006_3FFF	USBD_BA	USB 2.0 FS 设备控制寄存器

0x400D_0000 – 0x400D_3FFF	ACMP_BA	模拟比较控制寄存器
0x400E_0000 – 0x400E_FFFF	ADC_BA	ADC 控制寄存器
APB2 控制器空间 (0x4010_0000 ~ 0x401F_FFFF)		
0x4010_0000 – 0x4010_3FFF	PS2_BA	PS/2 接口控制寄存器
0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3 控制寄存器
0x4012_0000 – 0x4012_3FFF	I2C1_BA	I ² C1 接口控制寄存器
0x4013_0000 – 0x4013_3FFF	SPI2_BA	带主/从功能的SPI2 控制寄存器
0x4013_4000 – 0x4013_7FFF	SPI3_BA	带主/从功能的SPI3 控制寄存器
0x4014_0000 – 0x4014_3FFF	PWMB_BA	PWM4/5/6/7 控制寄存器
0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 控制寄存器
0x4015_4000 – 0x4015_7FFF	UART2_BA	UART2 控制寄存器
0x4018_0000 – 0x4018_3FFF	CAN0_BA	CAN0 总线控制寄存器
0x401A_0000 – 0x401A_3FFF	I2S_BA	I ² S 接口控制寄存器
系统控制器空间 (0xE000_E000 ~ 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SCS_BA	System 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	System 控制寄存器

表 5-1 片上控制器的地址空间分配

5.2.5 系统管理器控制寄存器

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

寄存器	偏移量	R/W	描述	复位后的值
GCR_BA = 0x5000_0000				
PDID	GCR_BA+0x00	R	器件 ID 寄存器	0x0014_0018 ^[1]
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX
IPRSTC1	GCR_BA+0x08	R/W	外设复位控制寄存器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	温度传感器控制寄存器	0x0000_0000
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_00XX
GPA_MFP	GCR_BA+0x30	R/W	GPIOA 多功能和输入类型控制寄存器	0x0000_0000
GPB_MFP	GCR_BA+0x34	R/W	GPIOB 多功能和输入类型控制寄存器	0x0000_0000
GPC_MFP	GCR_BA+0x38	R/W	GPIOC 多功能和输入类型控制寄存器	0x0000_0000
GPD_MFP	GCR_BA+0x3C	R/W	GPIOD 多功能和输入类型控制寄存器	0x0000_0000
GPE_MFP	GCR_BA+0x40	R/W	GPIOE 多功能和输入类型控制寄存器	0x0000_0000
ALT_MFP	GCR_BA+0x50	R/W	复用多功能管脚控制寄存器	0x0000_0000
REGWRPROT	GCR_BA+0x100	R/W	寄存器写保护寄存器	0x0000_0000

注: [1] 依赖于零件登记号。



器件 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	器件 ID 寄存器	0x0014_0018 ^[1]

[1] 每个器件独有一个独一无二的默认复位值。

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

Bits	描述	
[31:0]	PDID	产品器件识别码 该寄存器反映器件的器件号码。软件可以读该寄存器来识别所使用的器件。



NuMicro NUC130/NUC140™ Series	Part Device Identification Number
NUC130LC1CN	0x20013008
NUC130LD2CN	0x20013004
NUC130LE3CN	0x20013000
NUC130RC1CN	0x20013017
NUC130RD2CN	0x20013013
NUC130RE3CN	0x20013009
NUC130VE3CN	0x20013018
NUC140LC1CN	0x20014008
NUC140LD2CN	0x20014004
NUC140LE3CN	0x20014000
NUC140RC1CN	0x20014017
NUC140RD2CN	0x20014013
NUC140RE3CN	0x20014009
NUC140VE3CN	0x20014018

系统复位源寄存器 (RSTSRC)

该寄存器提供一些信息用于识别引起芯片上次复位操作的复位源。

寄存器	偏移量	R/W	描述	复位后的值
RSTSRC	GCR_BA+0x04	R/W	系统复位源寄存器	0x0000_00XX

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

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

		1 = 低压 LVR 模块发出复位信号使系统复位。 0 = LVR 无复位 向该位写 1 清零。
[2]	RSTS_WDT	RSTS_WDT 标志位由看门狗模块的“复位信号”置位，用于表示导致之前复位的复位源。 1 = 看门狗模块发出复位信号是系统复位。 0 = 看门狗无复位 向该位写 1 清零。
[1]	RSTS_RESET	RSTS_RESET 标志位由 /RESET 管脚的“复位信号”置位，用于表示导致之前复位的复位源。 1 = /RESET 管脚发出复位信号使系统复位 0 = /RESET 管脚无复位 向该位写 1 清零。
[0]	RSTS_POR	RSTS_POR 标志位由 上电复位（POR）的“复位信号”或 CHIP_RST (IPRSTC1[0]) 位置位，用于表示导致之前复位的复位源。 1 = 上电复位（POR）或 CHIP_RST 发出复位信号是系统复位 0 = 上电复位（POR）或 CHIP_RST 无复位 向该位写 1 清零。

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+0x08	R/W	外设复位控制寄存器1	0x0000_0000

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

Bits	描述	
[31:4]	Reserved	保留
[3]	EBI_RST	EBI 控制器复位 (在 NUC130/NUC140 100-pin and 64-pin 的封装芯片中该位写保护) 该位置 1，产生复位信号到 EBI。用户需要置 0 才能释放复位状态。 该位受保护，编程该位时，需要依次向地址 0x5000_0100 写入 “59h”, “16h”, “88h”，该操作参考寄存器 REGWRPROT (地址 GCR_BA+0x100) 1 = EBI 控制器复位 0 = EBI 控制器正常工作
[2]	PDMA_RST	PDMA 控制器复位 (在 NUC130/NUC140 中该位写保护) 该位置 1，产生复位信号到 PDMA，用户需要置 0 才能释放复位状态。 该位受保护，编程该位时，需要依次向地址 0x5000_0100 写入 “59h”, “16h”, “88h”，该操作参考寄存器 REGWRPROT (地址 GCR_BA+0x100) 1 = PDMA 控制器复位 0 = PDMA 控制器正常工作
[1]	CPU_RST	CPU 内核复位 (写保护位) 设置该位仅复位 CPU 内核和Flash 存储控制器 (FMC)，该位将在2个时钟周期后自动清零 该位受保护，编程该位时，需要依次向地址 0x5000_0100 写入 “59h”, “16h”, “88h”，该操作参考寄存器 REGWRPROT (地址 GCR_BA+0x100) 1 = CPU 复位 0 = CPU 正常工作
[0]	CHIP_RST	CHIP 复位 (写保护) 设置该位复位整个芯片，包括CPU 内核和所有外设，该位将在2个时钟周期后自动清

		<p>零。</p> <p>CHIP_RST 与上电复位 (POR) 一样，所有芯片控制器都复位，芯片设置从 flash 重新加载。</p> <p>CHIP_RST 和 SYSRESETREQ 的区别，请参考章节 5.2.2</p> <p>该位受保护，编程该位时，需要依次向地址 0x5000_0100 写入 “59h”，“16h”，“88h”，该操作参考寄存器 REGWRPROT（地址 GCR_BA+0x100）</p> <p>1 = CHIP 复位</p> <p>0 = CHIP 正常工作</p>
--	--	---

外设复位控制寄存器2 (IPRSTC2)

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

寄存器	偏移量	R/W	描述	复位后的值
IPRSTC2	GCR_BA+0x0C	R/W	外设复位控制寄存器2	0x0000_0000

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

Bits	Descriptions	
[31:30]	Reserved	保留
[29]	I2S_RST	I²S 控制器复位 1 = I ² S 控制器复位 0 = I ² S 控制器正常工作
[28]	ADC_RST	ADC 控制器复位 1 = ADC 控制器复位 0 = ADC 控制器正常工作
[27]	USBD_RST	USB 设备控制器复位 1 = USB 设备控制器复位 0 = USB 设备控制器正常工作
[26:25]	Reserved	保留
[24]	CAN0_RST	CAN0 控制器复位 1 = CAN0 控制器复位 0 = CAN0 控制器正常工作
[23]	PS2_RST	PS/2 控制器复位 1 = PS/2 控制器复位 0 = PS/2 控制器正常工作
[22]	ACMP_RST	模拟比较控制器复位

		1 = 模拟比较控制器复位 0 = 模拟比较控制器正常工作
[21]	PWM47_RST	PWM47 控制器复位 1 = PWM47 控制器复位 0 = PWM47 控制器正常工作
[20]	PWM03_RST	PWM03 控制器复位 1 = PWM03 控制器复位 0 = PWM03 控制器正常工作
[19]	Reserved	保留
[18]	UART2_RST	UART2 控制器复位 1 = UART2 控制器复位 0 = UART2 控制器正常工作
[17]	UART1_RST	UART1 控制器复位 1 = UART1 控制器复位 0 = UART1 控制器正常工作
[16]	UART0_RST	UART0 控制器复位 1 = UART0 控制器复位 0 = UART0 控制器正常工作
[15]	SPI3_RST	SPI3 控制器复位 1 = SPI3 控制器复位 0 = SPI3 控制器正常工作
[14]	SPI2_RST	SPI2 控制器复位 1 = SPI2 控制器复位 0 = SPI2 控制器正常工作
[13]	SPI1_RST	SPI1 控制器复位 1 = SPI1 控制器复位 0 = SPI1 控制器正常工作
[12]	SPI0_RST	SPI0 控制器复位 1 = SPI0 控制器复位 0 = SPI0 控制器正常工作
[11:10]	Reserved	保留
[9]	I2C1_RST	I²C1 控制器复位 1 = I ² C1 控制器复位 0 = I ² C1 控制器正常工作
[8]	I2C0_RST	I²C0 控制器复位

		1 = I ² C0 控制器复位 0 = I ² C0 控制器正常工作
[7:6]	Reserved	保留
[5]	TMR3_RST	Timer3 控制器复位 1 = Timer3 控制器复位 0 = Timer3 控制器正常工作
[4]	TMR2_RST	Timer2 控制器复位 1 = Timer2 控制器复位 0 = Timer2 控制器正常工作
[3]	TMR1_RST	Timer1 控制器复位 1 = Timer1 控制器复位 0 = Timer1 控制器正常工作
[2]	TMR0_RST	Timer0 控制器复位 1 = Timer0 控制器复位 0 = Timer0 控制器正常工作
[1]	GPIO_RST	GPIO 控制器复位 1 = GPIO 控制器复位 0 = GPIO 控制器正常工作
[0]	Reserved	保留

欠压检测控制寄存器 (BODCR)

BODCR 控制寄存器的部分位在 flash 配置时 (config0 寄存器) 已经被初始化，部分位是受保护的位。编程这些写保护的位时，需要向地址 0x5000_0100 依次写入“59h”，“16h”，“88h”。该位操作请参考寄存器 REGWRPROT (地址 GCR_BA+0x100)

寄存器	偏移量	R/W	描述	复位后的值
BODCR	GCR_BA+0x18	R/W	欠压检测控制寄存器	0x0000_008X

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

Bits	描述	
[31:8]	Reserved	保留
[7]	LVR_EN	低压复位使能 (写保护位) 当输入电源电压低于 LVR 电路设置时，LVR 复位芯片。默认使能低电压复位功能。 1 = 使能低电压复位功能，使能该位100us后，低电压复位输出稳定，LVR功能生效（默认） 0 = 禁用低电压复位功能 该位受保护，编程该位时，需要依次向地址 0x5000_0100 写入 “59h”，“16h”，“88h”，该操作参考寄存器 REGWRPROT (地址 GCR_BA+0x100)
[6]	BOD_OUT	欠压检测输出状态位 1 = 欠压检测输出状态为 1。表示检测到的电压低于 BOD_VL 的设置。若 BOD_EN 是 '0'，禁用 BOD 功能，该位通常响应必定为 '0'。 0 = 欠压检测输出状态为 0。表示检测到的电压高于 BOD_VL 的设置或者 BOD_EN 为 0。
[5]	BOD_LPM	低压模式下的欠压检测 (写保护位) 1 = 使能 BOD 的低功耗模式 0 = BOD 工作在正常模式（默认） BOD 在正常模式下消耗电流约为100 uA，低功耗模式下能减小到当前的约1/10，但 BOD 响应速度变慢。 该位受保护，编程该位时，需要依次向地址 0x5000_0100 写入 “59h”，“16h”，“88h”，该操作参考寄存器 REGWRPROT (地址 GCR_BA+0x100)

[4]	BOD_INTF	<p>欠压检测中断标志</p> <p>1 = 当欠压检测到 VDD 下降到 BOD_VL 的设定电压或者 VDD 上升到 BOD_VL 的设定电压, 该位设为1, 如果欠压电压中断被使能, 则发生欠压中断。</p> <p>0 = 欠压检测没有检测到任何电压由 VDD 下降或者上升到 BOD_VL 设定值。向该位写1清零。</p>															
[3]	BOD_RSTEN	<p>欠压复位使能 (写保护位)</p> <p>1 = 使能欠压复位功能 当同时使能欠压检测功能 (BOD_EN 为高) 和 BOD 复位功能 (BOD_RSTEN 为高) 时, 如果检测到电压低于阈电压 (BOD_OUT 为高), BOD 将发送信号复位芯片。</p> <p>0 = 使能欠压中断功能 当同时使能 BOD 功能 (BOD_EN 为高) 和 BOD 中断功能 (BOD_RSTEN 为低), 如果 BOD_OUT 为高, 则 BOD 将产生中断。BOD 中断将保持直到 BOD_EN 被设置为0。可以通过禁用 NVIC BOD 中断或者禁用 BOD 功能 (设置 BOD_EN 为低) 封锁 BOD 中断。</p> <p>默认值由用户在配置 flash 控制寄存器 (config0 bit[20]) 时设置。 该位受保护, 编程该位时, 需要依次向地址 0x5000_0100 写入 “59h”, “16h”, “88h”, 该操作参考寄存器 REGWRPROT (地址 GCR_BA+0x100)</p>															
[2:1]	BOD_VL	<p>欠压检测阈电压选择 (写保护位)</p> <p>默认值由用户在配置 flash 控制寄存器 (config0 bit[22:21]) 时设置。 该位受保护, 编程该位时, 需要依次向地址 0x5000_0100 写入 “59h”, “16h”, “88h”, 该操作参考寄存器 REGWRPROT (地址 GCR_BA+0x100)</p> <table border="1"> <thead> <tr> <th>BOV_VL[1]</th> <th>BOV_VL[0]</th> <th>Brown-Out voltage</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>4.5 V</td> </tr> <tr> <td>1</td> <td>0</td> <td>3.8 V</td> </tr> <tr> <td>0</td> <td>1</td> <td>2.7 V</td> </tr> <tr> <td>0</td> <td>0</td> <td>2.2 V</td> </tr> </tbody> </table>	BOV_VL[1]	BOV_VL[0]	Brown-Out voltage	1	1	4.5 V	1	0	3.8 V	0	1	2.7 V	0	0	2.2 V
BOV_VL[1]	BOV_VL[0]	Brown-Out voltage															
1	1	4.5 V															
1	0	3.8 V															
0	1	2.7 V															
0	0	2.2 V															
[0]	BOD_EN	<p>欠压检测使能 (写保护位)</p> <p>默认值由用户在配置 flash 控制寄存器 (config0 bit[23]) 时设置。</p> <p>1 = 使能欠压检测功能 0 = 禁用欠压检测功能 该位受保护, 编程该位时, 需要依次向地址 0x5000_0100 写入 “59h”, “16h”, “88h”, 该操作参考寄存器 REGWRPROT (地址 GCR_BA+0x100)</p>															

温度传感器控制寄存器 (TEMPCR)

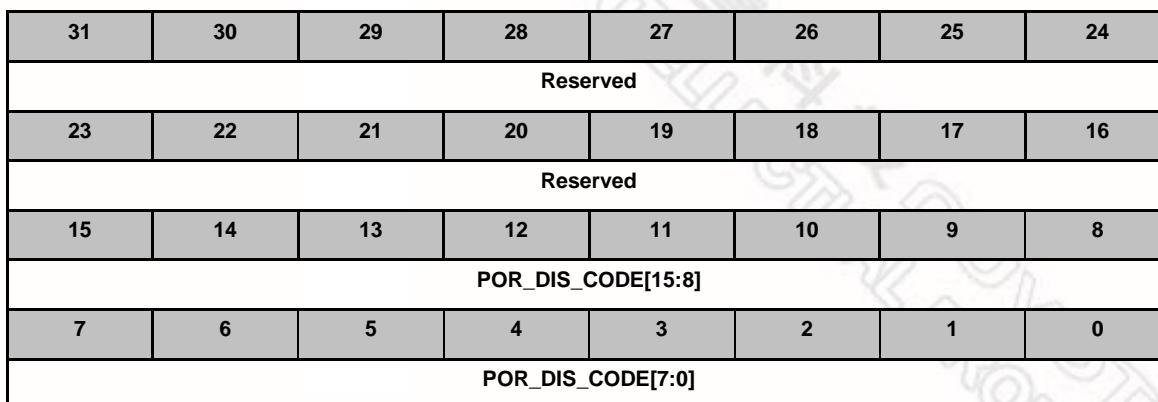
寄存器	偏移量	R/W	描述	复位后的值
TEMPCR	GCR_BA+0x1C	R/W	温度传感器控制寄存器	0x0000_0000

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

Bits	描述	
[31:1]	Reserved	保留
[0]	VTEMP_EN	<p>温度传感器使能 该位用于使能/禁用温度传感器功能。 1 = 使能温度传感器功能 0 = 禁用温度传感器功能</p> <p>该位置 1 后，温度值可以从 ADC 转换结果获得，这之前需要把 ADC channel 选择设为 channel 7，择一多路转换通道选择到温度传感器选项。具体的 ADC 转换功能请参考 ADC 功能章节。</p>

上电复位控制寄存器 (PORCR)

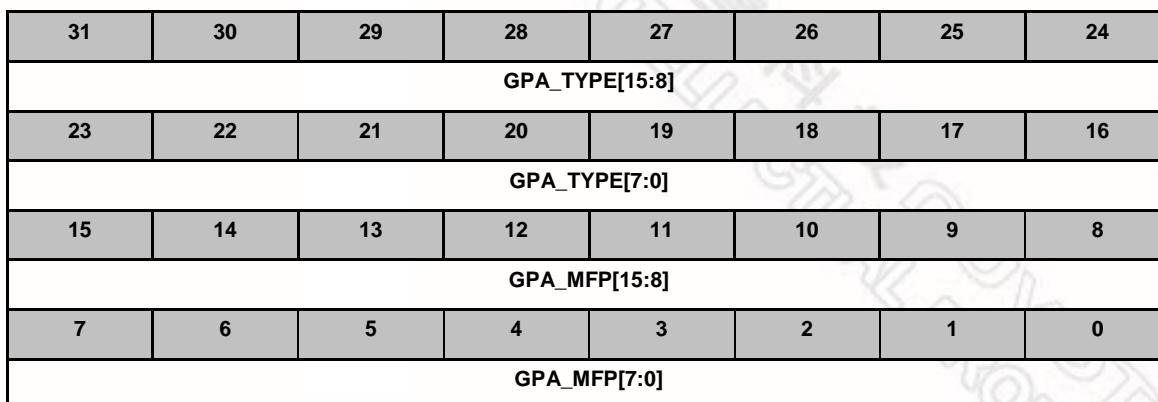
寄存器	偏移量	R/W	描述	复位后的值
PORCR	GCR_BA+0x24	R/W	上电复位控制寄存器	0x0000_00XX



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

GPIOA 多功能管脚控制寄存器 (GPA_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPA_MFP	GCR_BA+0x30	R/W	GPIOA 多功能与输入类型控制寄存器	0x0000_0000



Bits	描述																		
[31:16]	GPA_TYPEn	1 = 使能 GPIOA[15:0] I/O Schmitt 触发输入 0 = 禁用 GPIOA[15:0] I/O Schmitt 触发输入																	
[15]	GPA_MFP15	PA.15 管脚功能选择 该管脚功能取决于 GPA_MFP15 和 PA15_I2SMCLK (ALT_MFP[9])。 <table border="1"> <tr><td>PA15_I2SMCLK</td><td>GPA_MFP[15]</td><td>PA.15 功能</td></tr> <tr><td>0</td><td>0</td><td>GPIO</td></tr> <tr><td>0</td><td>1</td><td>PWM3 (PWM)</td></tr> <tr><td>1</td><td>1</td><td>I2SMCLK (I²S)</td></tr> </table>	PA15_I2SMCLK	GPA_MFP[15]	PA.15 功能	0	0	GPIO	0	1	PWM3 (PWM)	1	1	I2SMCLK (I ² S)					
PA15_I2SMCLK	GPA_MFP[15]	PA.15 功能																	
0	0	GPIO																	
0	1	PWM3 (PWM)																	
1	1	I2SMCLK (I ² S)																	
[14]	GPA_MFP14	PA.14 管脚功能选择 该管脚功能取决于 GPA_MFP14、EBI_HB_EN[7] (ALT_MFP[23]) 和 EBI_EN (ALT_MFP[11])。 <table border="1"> <tr><td>EBI_HB_EN[7]</td><td>EBI_EN</td><td>GPA_MFP[14]</td><td>PA.14 功能</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>PWM2 (PWM)</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>AD15 (EBI AD bus bit 15)</td></tr> </table>	EBI_HB_EN[7]	EBI_EN	GPA_MFP[14]	PA.14 功能	0	0	0	GPIO	0	0	1	PWM2 (PWM)	1	1	1	AD15 (EBI AD bus bit 15)	
EBI_HB_EN[7]	EBI_EN	GPA_MFP[14]	PA.14 功能																
0	0	0	GPIO																
0	0	1	PWM2 (PWM)																
1	1	1	AD15 (EBI AD bus bit 15)																
[13]	GPA_MFP13	PA.13 管脚功能选择 该管脚功能取决于 GPA_MFP13、EBI_HB_EN[6] (ALT_MFP[22]) 和 EBI_EN (ALT_MFP[11])。 <table border="1"> <tr><td>EBI_HB_EN[6]</td><td>EBI_EN</td><td>GPA_MFP[13]</td><td>PA.13 功能</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr> </table>	EBI_HB_EN[6]	EBI_EN	GPA_MFP[13]	PA.13 功能	0	0	0	GPIO									
EBI_HB_EN[6]	EBI_EN	GPA_MFP[13]	PA.13 功能																
0	0	0	GPIO																

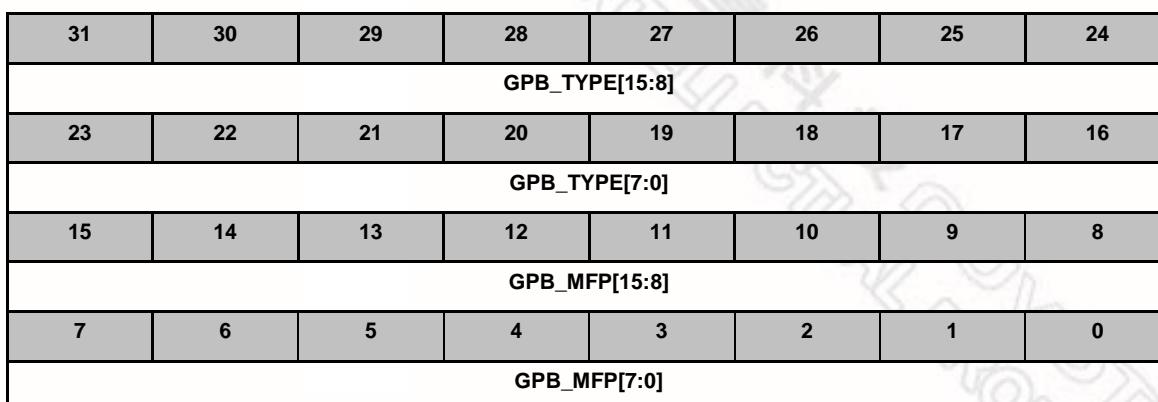
		0	0	1	PWM1 (PWM)	
		1	1	1	AD14 (EBI AD bus bit 14)	
[12]	GPA_MFP12	PA.12 管脚功能选择 该管脚功能取决于 GPA_MFP12、EBI_HB_EN[5] (ALT_MFP[21]) 和 EBI_EN (ALT_MFP[11])。				
		EBI_HB_EN[5]	EBI_EN	GPA_MFP[12]	PA.12 功能	
		0	0	0	GPIO	
		0	0	1	PWM0 (PWM)	
		1	1	1	AD13 (EBI AD bus bit 13)	
[11]	GPA_MFP11	PA.11 管脚功能选择 该管脚功能取决于 GPA_MFP11 和 EBI_EN (ALT_MFP[11])。				
		EBI_EN	GPA_MFP[11]	PA.11 功能		
		0	0	GPIO		
		0	1	SCL1 (I ² C)		
		1	1	nRD (EBI)		
[10]	GPA_MFP10	PA.10 管脚功能选择 该管脚功能取决于 GPA_MFP10 和 EBI_EN (ALT_MFP[11])。				
		EBI_EN	GPA_MFP[10]	PA.10 功能		
		0	0	GPIO		
		0	1	SDA1 (I ² C)		
		1	1	nWR (EBI)		
[9]	GPA_MFP9	PA.9 管脚功能选择 1 = PA.9 作为 I ² C0 SCL 0 = PA.9 作为 GPIOA[9]				
[8]	GPA_MFP8	PA.8 管脚功能选择 1 = PA.8 作为 I ² C0 SDA 0 = PA.8 作为 GPIOA[8]				
[7]	GPA_MFP7	PA.7 管脚功能选择 该管脚功能取决于 GPA_MFP7、PA7_S21 (ALT_MFP[2]) 和 EBI_EN (ALT_MFP[11])				
		EBI_EN	PA7_S2	GPA_MFP[7]	PA.7 功能	
		0	0	0	GPIO	
		0	0	1	ADC7 (ADC)	
		0	1	1	SPISS21 (SPI2)	
[6]	GPA_MFP6	PA.6 管脚功能选择				

		该管脚功能取决于 GPA_MFP6 和 EBI_EN (ALT_MFP[11])。			
		EBI_EN	GPA_MFP[6]	PA.6 功能	
		0	0	GPIO	
		0	1	ADC6 (ADC)	
		1	1	AD7 (EBI AD bus bit 7)	
[5]	GPA_MFP5	PA.5 管脚功能选择 该管脚功能取决于 GPA_MFP5 、 EBI_HB_EN[0] (ALT_MFP[16]) 和 EBI_EN (ALT_MFP[11])。			
		EBI_HB_EN[0]	EBI_EN	GPA_MFP[5]	PA.5 功能
		0	0	0	GPIO
		0	0	1	ADC5 (ADC)
		1	1	1	AD8 (EBI AD bus bit 8)
[4]	GPA_MFP4	PA.4 管脚功能选择 该管脚功能取决于 GPA_MFP4 、 EBI_HB_EN[1] (ALT_MFP[17]) 和 EBI_EN (ALT_MFP[11])。			
		EBI_HB_EN[1]	EBI_EN	GPA_MFP[4]	PA.4 功能
		0	0	0	GPIO
		0	0	1	ADC4 (ADC)
		1	1	1	AD9 (EBI AD bus bit 9)
[3]	GPA_MFP3	PA.3 管脚功能选择 该管脚功能取决于 GPA_MFP3 、 EBI_HB_EN[2] (ALT_MFP[18]) 和 EBI_EN (ALT_MFP[11])。			
		EBI_HB_EN[2]	EBI_EN	GPA_MFP[2]	PA.3 功能
		0	0	0	GPIO
		0	0	1	ADC3 (ADC)
		1	1	1	AD10 (EBI AD bus bit 10)
[2]	GPA_MFP2	PA.2 管脚功能选择 该管脚功能取决于 GPA_MFP2 、 EBI_HB_EN[3] (ALT_MFP[19]) 和 EBI_EN (ALT_MFP[11])。			
		EBI_HB_EN[3]	EBI_EN	GPA_MFP[2]	PA.2 功能
		0	0	0	GPIO
		0	0	1	ADC2 (ADC)
		1	1	1	AD11 (EBI AD bus bit 11)
[1]	GPA_MFP1	PA.1 管脚功能选择 该管脚功能取决于 GPA_MFP1 、 EBI_HB_EN[4] (ALT_MFP[20]) 和 EBI_EN (ALT_MFP[11])。			

		EBI_HB_EN[4]	EBI_EN	GPA_MFP[1]	PA.1 功能
		0	0	0	GPIO
		0	0	1	ADC1 (ADC)
		1	1	1	AD12 (EBI AD bus bit 12)
[0]	GPA_MFP0	PA.0 管脚功能选择 1 = PA.0 作为 ADC0 0 = PA.0 作为 GPIOA[0]			

GPIOB 多功能管脚控制寄存器 (GPB_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPB_MFP	GCR_BA+0x34	R/W	GPIOB 多功能与输入类型控制寄存器	0x0000_0000



Bits	描述													
[31:16]	GPB_TYPEn	1 = 使能 GPIOB[15:0] I/O Schmitt 触发输入 0 = 禁用 GPIOB[15:0] I/O Schmitt 触发输入												
[15]	GPB_MFP15	PB.15 管脚功能选择 该管脚功能取决于 GPB_MFP15 和 PB15_TOEX (ALT_MFP[24])。 <table border="1"> <tr><td>PB15_TOEX</td><td>GPB_MFP[15]</td><td>PB.15 功能</td></tr> <tr><td>0</td><td>0</td><td>GPIO</td></tr> <tr><td>0</td><td>1</td><td>/INT1</td></tr> <tr><td>1</td><td>1</td><td>TOEX (TMR0)</td></tr> </table>	PB15_TOEX	GPB_MFP[15]	PB.15 功能	0	0	GPIO	0	1	/INT1	1	1	TOEX (TMR0)
PB15_TOEX	GPB_MFP[15]	PB.15 功能												
0	0	GPIO												
0	1	/INT1												
1	1	TOEX (TMR0)												
[14]	GPB_MFP14	PB.14 管脚功能选择 该管脚功能取决于 GPB_MFP14 和 PB14_S31 (ALT_MFP[3])。 <table border="1"> <tr><td>PB14_S31</td><td>GPB_MFP[14]</td><td>PB.14 功能</td></tr> <tr><td>0</td><td>0</td><td>GPIO</td></tr> <tr><td>0</td><td>1</td><td>/INT0</td></tr> <tr><td>1</td><td>1</td><td>SPISS31 (SPI3)</td></tr> </table>	PB14_S31	GPB_MFP[14]	PB.14 功能	0	0	GPIO	0	1	/INT0	1	1	SPISS31 (SPI3)
PB14_S31	GPB_MFP[14]	PB.14 功能												
0	0	GPIO												
0	1	/INT0												
1	1	SPISS31 (SPI3)												
[13]	GPB_MFP13	PB.13 管脚功能选择 该管脚功能取决于 GPB_MFP13 和 EBI_EN (ALT_MFP[11])。 <table border="1"> <tr><td>EBI_EN</td><td>GPB_MFP[13]</td><td>PB.13 功能</td></tr> <tr><td>0</td><td>0</td><td>GPIO</td></tr> </table>	EBI_EN	GPB_MFP[13]	PB.13 功能	0	0	GPIO						
EBI_EN	GPB_MFP[13]	PB.13 功能												
0	0	GPIO												

		0	1	CPO1 (CMP)	
		1	1	AD1 (EBI AD bus bit 1)	
[12]	GPB_MFP12	PB.12 管脚功能选择 该管脚功能取决于 GPB_MFP12、PB12_CLKO (ALT_MFP[10]) 和 EBI_EN (ALT_MFP[11])。			
		EBI_EN	PB12_CLKO	GPB_MFP[12]	PB.12 功能
		0	0	0	GPIO
		0	0	1	CPO0 (CMP)
		0	1	1	CLKO (Clock Driver output)
		1	0	1	AD0 (EBI AD bus bit 0)
[11]	GPB_MFP11	PB.11 管脚功能选择 该管脚功能取决于 GPB_MFP11 和 PB11_PWM4 (ALT_MFP[4])。			
		PB11_PWM4	GPB_MFP[11]	PB.11 功能	
		0	0	GPIO	
		0	1	TM3	
		1	1	PWM4 (PWM)	
[10]	GPB_MFP10	PB.10 管脚功能选择 该管脚功能取决于 GPB_MFP10 和 PB10_S01 (ALT_MFP[0])。			
		PB10_S01	GPB_MFP[10]	PB.10 功能	
		0	0	GPIO	
		0	1	TM2	
		1	1	SPISS01 (SPI0)	
[9]	GPB_MFP9	PB.9 管脚功能选择 该管脚功能取决于 GPB_MFP9 和 PB9_S11 (ALT_MFP[1])。			
		PB9_S11	GPB_MFP[9]	PB.9 功能	
		0	0	GPIO	
		0	1	TM1	
		1	1	SPISS11 (SPI1)	
[8]	GPB_MFP8	PB.8 管脚功能选择 1 = PB.8 作为 TM0 (定时器/计数器外部触发时钟输入) 0 = PB.8 作为 GPIOB[8]			
[7]		PB.7 管脚功能选择 该管脚功能取决于 GPB_MFP7 和 EBI_EN (ALT_MFP[11])。			
		EBI_EN	GPB_MFP[7]	PB.7 功能	

		<table border="1"> <tr><td>0</td><td>0</td><td>GPIO</td></tr> <tr><td>0</td><td>1</td><td>CTS1 (UART1)</td></tr> <tr><td>1</td><td>1</td><td>nCS (EBI)</td></tr> </table>	0	0	GPIO	0	1	CTS1 (UART1)	1	1	nCS (EBI)																
0	0	GPIO																									
0	1	CTS1 (UART1)																									
1	1	nCS (EBI)																									
[6]	GPB_MFP6	PB.6 管脚功能选择 该管脚功能取决于 GPB_MFP6 和 EBI_EN (ALT_MFP[11])。 <table border="1"> <tr><th>EBI_EN</th><th>GPB_MFP[6]</th><th>PB.6 功能</th></tr> <tr><td>0</td><td>0</td><td>GPIO</td></tr> <tr><td>0</td><td>1</td><td>RTS1 (UART1)</td></tr> <tr><td>1</td><td>1</td><td>ALE (EBI)</td></tr> </table>	EBI_EN	GPB_MFP[6]	PB.6 功能	0	0	GPIO	0	1	RTS1 (UART1)	1	1	ALE (EBI)													
EBI_EN	GPB_MFP[6]	PB.6 功能																									
0	0	GPIO																									
0	1	RTS1 (UART1)																									
1	1	ALE (EBI)																									
[5]	PB.5 管脚功能选择 1 = PB.5 作为 UART1 TXD 0 = PB.5 作为 GPIOB[5]																										
[4]	PB.4 管脚功能选择 1 = PB.4 作为 UART1 RXD 0 = PB.4 作为 GPIOB[4]																										
[3]	GPB_MFP3	PB.3 管脚功能选择 该管脚功能取决于 GPB_MFP3、EBI_nWRH_EN (ALT_MFP[14])、EBI_EN (ALT_MFP[11]) 和 PB3_T3EX (ALT_MFP[27])。 <table border="1"> <tr><th>EBI_nWRH_EN</th><th>EBI_EN</th><th>GPB_MFP[3]</th><th>PB3_T3EX</th><th>PB.3 功能</th></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>CTS0 (UART0)</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>nWRH (EBI write high byte enable)</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>T3EX (TMR3)</td></tr> </table>	EBI_nWRH_EN	EBI_EN	GPB_MFP[3]	PB3_T3EX	PB.3 功能	0	0	0	0	GPIO	0	0	1	0	CTS0 (UART0)	1	1	1	0	nWRH (EBI write high byte enable)	0	0	1	1	T3EX (TMR3)
EBI_nWRH_EN	EBI_EN	GPB_MFP[3]	PB3_T3EX	PB.3 功能																							
0	0	0	0	GPIO																							
0	0	1	0	CTS0 (UART0)																							
1	1	1	0	nWRH (EBI write high byte enable)																							
0	0	1	1	T3EX (TMR3)																							
[2]	GPB_MFP2	PB.2 管脚功能选择 该管脚功能取决于 GPB_MFP2、EBI_nWRL_EN (ALT_MFP[13])、EBI_EN (ALT_MFP[11]) 和 PB2_T2EX (ALT_MFP[26])。 <table border="1"> <tr><th>EBI_nWRL_EN</th><th>EBI_EN</th><th>GPB_MFP[2]</th><th>PB2_T2EX</th><th>PB.2 功能</th></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>0</td><td>RTS0 (UART0)</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>0</td><td>nWRL (EBI write low byte enable)</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>1</td><td>T2EX (TMR2)</td></tr> </table>	EBI_nWRL_EN	EBI_EN	GPB_MFP[2]	PB2_T2EX	PB.2 功能	0	0	0	0	GPIO	0	0	1	0	RTS0 (UART0)	1	1	1	0	nWRL (EBI write low byte enable)	0	0	1	1	T2EX (TMR2)
EBI_nWRL_EN	EBI_EN	GPB_MFP[2]	PB2_T2EX	PB.2 功能																							
0	0	0	0	GPIO																							
0	0	1	0	RTS0 (UART0)																							
1	1	1	0	nWRL (EBI write low byte enable)																							
0	0	1	1	T2EX (TMR2)																							
[1]	GPB_MFP1	PB.1 管脚功能选择 1 = PB.1 作为 UART0 TXD 0 = PB.1 作为 GPIOB[1]																									



[0]	GPB_MFP0	PB.0 管脚功能选择 1 = PB.0 作为 UART0 RXD 0 = PB.0 作为 GPIOB[0]
-----	----------	--

GPIOC多功能管脚控制寄存器 (GPC_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPC_MFP	GCR_BA+0x38	R/W	GPIOC 多功能与输入类型控制寄存器	0x0000_0000

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

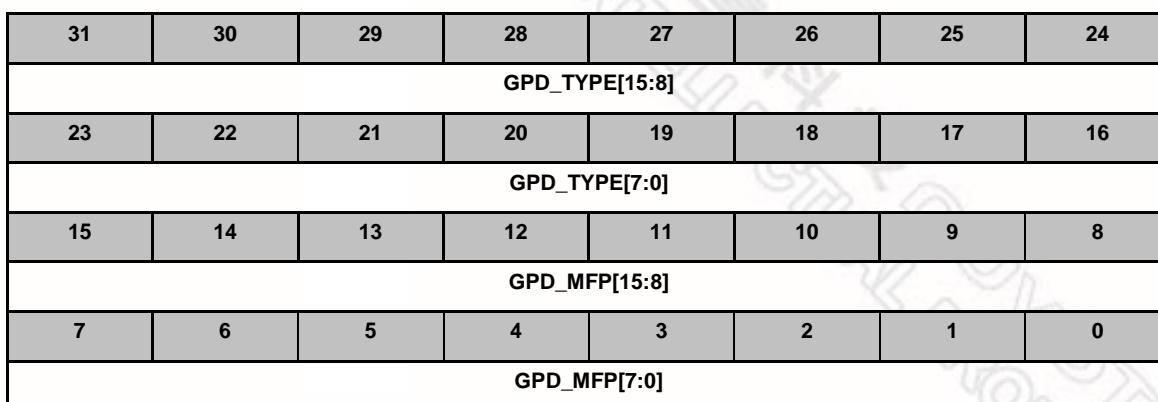
Bits	Descriptions														
[31:16]	GPC_TYPEn	1 = 使能 GPIOC[15:0] I/O Schmitt 触发输入 0 = 禁用 GPIOC[15:0] I/O Schmitt 触发输入													
[15]	GPC_MFP15	PC.15 管脚功能选择 该管脚功能取决于 GPC_MFP15 和 EBI_EN (ALT_MFP[11])。	<table border="1"> <tr> <td>EBI_EN</td> <td>GPC_MFP[15]</td> <td>PC.15 功能</td> </tr> <tr> <td>0</td> <td>0</td> <td>GPIO</td> </tr> <tr> <td>0</td> <td>1</td> <td>CPN1 (CMP)</td> </tr> <tr> <td>1</td> <td>1</td> <td>AD3 (EBI AD bus bit 3)</td> </tr> </table>	EBI_EN	GPC_MFP[15]	PC.15 功能	0	0	GPIO	0	1	CPN1 (CMP)	1	1	AD3 (EBI AD bus bit 3)
EBI_EN	GPC_MFP[15]	PC.15 功能													
0	0	GPIO													
0	1	CPN1 (CMP)													
1	1	AD3 (EBI AD bus bit 3)													
[14]	GPC_MFP14	PC.14 管脚功能选择 该管脚功能取决于 GPC_MFP14 和 EBI_EN (ALT_MFP[11])。	<table border="1"> <tr> <td>EBI_EN</td> <td>GPC_MFP[14]</td> <td>PC.14 功能</td> </tr> <tr> <td>0</td> <td>0</td> <td>GPIO</td> </tr> <tr> <td>0</td> <td>1</td> <td>CPP1 (CMP)</td> </tr> <tr> <td>1</td> <td>1</td> <td>AD2 (EBI AD bus bit 2)</td> </tr> </table>	EBI_EN	GPC_MFP[14]	PC.14 功能	0	0	GPIO	0	1	CPP1 (CMP)	1	1	AD2 (EBI AD bus bit 2)
EBI_EN	GPC_MFP[14]	PC.14 功能													
0	0	GPIO													
0	1	CPP1 (CMP)													
1	1	AD2 (EBI AD bus bit 2)													
[13]	GPC_MFP13	PC.13 管脚功能选择 1 = PC.13 作为 SPI1 MOSI1 (主机输出, 从机输入管脚-1) 0 = PC.13 作为 GPIOC[13]													
[12]	GPC_MFP12	PC.12 管脚功能选择 1 = PC.12 作为 SPI1 MISO1 (主机输入, 从机输出管脚-1)													

		0 = PC.12 作为 GPIOC[12]																
[11]	GPC_MFP11	PC.11 管脚功能选择 1 = PC.11 作为 SPI1 MOSI0 (主机输出, 从机输入管脚-0) 0 = PC.11 作为 GPIOC[11]																
[10]	GPC_MFP10	PC.10 管脚功能选择 1 = PC.10 作为 SPI1 MISO0 (主机输入, 从机输出管脚-0) 0 = PC.10 作为 GPIOC[10]																
[9]	GPC_MFP9	PC.9 管脚功能选择 1 = PC.9 作为 SPI1 SPICLK 0 = PC.9 作为 GPIOC[9]																
[8]	GPC_MFP8	PC.8 管脚功能选择 该管脚功能取决于 GPC_MFP8、EBI_MCLK_EN (ALT_MFP[12]) 和 EBI_EN (ALT_MFP[11])。 <table border="1"> <tr> <th>EBI_MCLK_EN</th> <th>EBI_EN</th> <th>GPC_MFP[8]</th> <th>PC.8 功能</th> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>GPIO</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>SPISS10 (SPI1)</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>MCLK (EBI Clock output)</td> </tr> </table>	EBI_MCLK_EN	EBI_EN	GPC_MFP[8]	PC.8 功能	0	0	0	GPIO	0	0	1	SPISS10 (SPI1)	1	1	1	MCLK (EBI Clock output)
EBI_MCLK_EN	EBI_EN	GPC_MFP[8]	PC.8 功能															
0	0	0	GPIO															
0	0	1	SPISS10 (SPI1)															
1	1	1	MCLK (EBI Clock output)															
[7]	GPC_MFP7	PC.7 管脚功能选择 该管脚功能取决于 GPC_MFP7 和 EBI_EN (ALT_MFP[11])。 <table border="1"> <tr> <th>EBI_EN</th> <th>GPC_MFP[7]</th> <th>PC.7 功能</th> </tr> <tr> <td>0</td> <td>0</td> <td>GPIO</td> </tr> <tr> <td>0</td> <td>1</td> <td>CPN0 (CMP)</td> </tr> <tr> <td>1</td> <td>1</td> <td>AD5 (EBI AD bus bit 5)</td> </tr> </table>	EBI_EN	GPC_MFP[7]	PC.7 功能	0	0	GPIO	0	1	CPN0 (CMP)	1	1	AD5 (EBI AD bus bit 5)				
EBI_EN	GPC_MFP[7]	PC.7 功能																
0	0	GPIO																
0	1	CPN0 (CMP)																
1	1	AD5 (EBI AD bus bit 5)																
[6]	GPC_MFP6	PC.6 管脚功能选择 该管脚功能取决于 GPC_MFP6 和 EBI_EN (ALT_MFP[11])。 <table border="1"> <tr> <th>EBI_EN</th> <th>GPC_MFP[6]</th> <th>PC.6 功能</th> </tr> <tr> <td>0</td> <td>0</td> <td>GPIO</td> </tr> <tr> <td>0</td> <td>1</td> <td>CPP0 (CMP)</td> </tr> <tr> <td>1</td> <td>1</td> <td>AD4 (EBI AD bus bit 4)</td> </tr> </table>	EBI_EN	GPC_MFP[6]	PC.6 功能	0	0	GPIO	0	1	CPP0 (CMP)	1	1	AD4 (EBI AD bus bit 4)				
EBI_EN	GPC_MFP[6]	PC.6 功能																
0	0	GPIO																
0	1	CPP0 (CMP)																
1	1	AD4 (EBI AD bus bit 4)																
[5]	GPC_MFP5	PC.5 管脚功能选择 1 = PC.5 作为 SPI0 MOSI1 (主机输出, 从机输入管脚-1) 0 = PC.5 作为 GPIOC[5]																
[4]	GPC_MFP4	PC.4 管脚功能选择 1 = PC.4 作为 SPI0 MISO1 (主机输入, 从机输出管脚-1) 0 = PC.4 作为 GPIOC[4]																

[3]	GPC_MFP3	PC.3 管脚功能选择												
		PC3_I2SDO (ALT_MFP[8]) 和 GPC_MFP[3] 决定了 PC.3 的功能。												
		<table border="1"> <thead> <tr> <th>PC3_I2SDO</th><th>GPC_MFP[3]</th><th>PC.3 功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>GPIO</td></tr> <tr> <td>0</td><td>1</td><td>MOSI00 (SPI0)</td></tr> <tr> <td>1</td><td>1</td><td>I2SDO (I²S)</td></tr> </tbody> </table>	PC3_I2SDO	GPC_MFP[3]	PC.3 功能	0	0	GPIO	0	1	MOSI00 (SPI0)	1	1	I2SDO (I ² S)
PC3_I2SDO	GPC_MFP[3]	PC.3 功能												
0	0	GPIO												
0	1	MOSI00 (SPI0)												
1	1	I2SDO (I ² S)												
PC.2 管脚功能选择														
[2]	GPC_MFP2	PC2_I2SDI (ALT_MFP[7]) 和 GPC_MFP[2] 决定了 PC.2 的功能。												
		<table border="1"> <thead> <tr> <th>PC2_I2SDI</th><th>GPC_MFP[2]</th><th>PC.2 功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>GPIO</td></tr> <tr> <td>0</td><td>1</td><td>MISO00 (SPI0)</td></tr> <tr> <td>1</td><td>1</td><td>I2SDI (I²S)</td></tr> </tbody> </table>	PC2_I2SDI	GPC_MFP[2]	PC.2 功能	0	0	GPIO	0	1	MISO00 (SPI0)	1	1	I2SDI (I ² S)
PC2_I2SDI	GPC_MFP[2]	PC.2 功能												
0	0	GPIO												
0	1	MISO00 (SPI0)												
1	1	I2SDI (I ² S)												
PC.1 管脚功能选择														
PC1_I2SBCLK (ALT_MFP[6]) 和 GPC_MFP[1] 决定了 PC.1 的功能。														
[1]	GPC_MFP1	<table border="1"> <thead> <tr> <th>PC1_I2SBCLK</th><th>GPC_MFP[1]</th><th>PC.1 功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>GPIO</td></tr> <tr> <td>0</td><td></td><td>SPICLK0 (SPI0)</td></tr> <tr> <td>1</td><td>1</td><td>I2SBCLK (I²S)</td></tr> </tbody> </table>	PC1_I2SBCLK	GPC_MFP[1]	PC.1 功能	0	0	GPIO	0		SPICLK0 (SPI0)	1	1	I2SBCLK (I ² S)
PC1_I2SBCLK	GPC_MFP[1]	PC.1 功能												
0	0	GPIO												
0		SPICLK0 (SPI0)												
1	1	I2SBCLK (I ² S)												
PC.0 管脚功能选择														
PC0_I2SLRCLK (ALT_MFP[5]) 和 GPC_MFP[0] 决定了 PC.0 的功能。														
<table border="1"> <thead> <tr> <th>PC0_I2SLRCLK</th><th>GPC_MFP[0]</th><th>PC.0 功能</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>GPIO</td></tr> <tr> <td>0</td><td>1</td><td>SPISS00 (SPI0)</td></tr> <tr> <td>1</td><td>1</td><td>I2SLRCLK (I²S)</td></tr> </tbody> </table>	PC0_I2SLRCLK	GPC_MFP[0]	PC.0 功能	0	0	GPIO	0	1	SPISS00 (SPI0)	1	1	I2SLRCLK (I ² S)		
PC0_I2SLRCLK	GPC_MFP[0]	PC.0 功能												
0	0	GPIO												
0	1	SPISS00 (SPI0)												
1	1	I2SLRCLK (I ² S)												
[0]	GPC_MFP0													

GPIOD 多功能管脚控制寄存器 (GPD_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPD_MFP	GCR_BA+0x3C	R/W	GPIOD 多功能与输入类型控制寄存器	0x0000_0000



Bits	描述	
[31:16]	GPD_TYPEn	1 = 使能 GPIOD[15:0] I/O Schmitt 触发输入 0 = 禁用 GPIOD[15:0] I/O Schmitt 触发输入
[15]	GPD_MFP15	PD.15 管脚功能选择 1 = PD.15 作为 UART2 TXD 0 = PD.15 作为 GPIOD[15]
[14]	GPD_MFP14	PD.14 管脚功能选择 1 = PD.14 作为 UART2 RXD 0 = PD.14 作为 GPIOD[14]
[13]	GPD_MFP13	PD.13 管脚功能选择 1 = PD.13 作为 SPI3 MOSI1 (主机输出, 从机输入管脚-1) 0 = PD.13 作为 GPIOD[13]
[12]	GPD_MFP12	PD.12 管脚功能选择 1 = PD.12 作为 SPI3 MISO1 (主机输入, 从机输出管脚-1) 0 = PD.12 作为 GPIOD[12]
[11]	GPD_MFP11	PD.11 管脚功能选择 1 = PD.11 作为 SPI3 MOSI0 (主机输出, 从机输入管脚-0) 0 = PD.11 作为 GPIOD[11]
[10]	GPD_MFP10	PD.10 管脚功能选择 1 = PD.10 作为 SPI3 MISO0 (主机输入, 从机输出管脚-0) 0 = PD.10 作为 GPIOD[10]

[9]	GPD_MFP9	PD.9 管脚功能选择 1 = PD.9 作为 SPI3 SPICLK 0 = PD.9 作为 GPIOD[9]
[8]	GPD_MFP8	PD.8 管脚功能选择 1 = PD.8 作为 SPI3 SS30 0 = PD.8 作为 GPIOD[8]
[7]	GPD_MFP7	PD.7 管脚功能选择 1 = PD.7 作为 CAN0 TX 0 = PD.7 作为 GPIOD[7]
[6]	GPD_MFP6	PD.6 管脚功能选择 1 = PD.6 作为 CAN0 RX 0 = PD.6 作为 GPIOD[6]
[5]	GPD_MFP5	PD.5 管脚功能选择 1 = PD.5 作为 SPI2 MOSI1 (主机输出, 从机输入管脚-1) 0 = PD.5 作为 GPIOD[5]
[4]	GPD_MFP4	PD.4 管脚功能选择 1 = PD.4 作为 SPI2 MISO1 (主机输入, 从机输出管脚-1) 0 = PD.4 作为 GPIOD[4]
[3]	GPD_MFP3	PD.3 管脚功能选择 1 = PD.3 作为 SPI2 MOSI0 (主机输出, 从机输入管脚-0) 0 = PD.3 作为 GPIOD[3]
[2]	GPD_MFP2	PD.2 管脚功能选择 1 = PD.2 作为 SPI2 MISO0 (主机输入, 从机输入管脚-0) 0 = PD.2 作为 GPIOD[2]
[1]	GPD_MFP1	PD.1 管脚功能选择 1 = PD.1 作为 SPI2 SPICLK 0 = PD.1 作为 GPIOD[1]
[0]	GPD_MFP0	PD.0 管脚功能选择 1 = PD.0 作为 SPI2 SS20 0 = PD.0 作为 GPIOD[0]

GPIOE 多功能管脚控制寄存器 (GPE_MFP)

寄存器	偏移量	R/W	描述	复位后的值
GPE_MFP	GCR_BA+0x40	R/W	GPIOE 多功能与输入类型控制寄存器	0x0000_0000

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

Bits	描述		
[31:16]	GPE_TYPEn		1 = 使能 GPIOE[15:0] I/O Schmitt 触发输入 0 = 禁用 GPIOE[15:0] I/O Schmitt 触发输入
[15:6]	Reserved		保留
[5]	PE.5 管脚功能选择 该管脚功能取决于 GPE_MFP5 和 PE5_T1EX (ALT_MFP[25])		
	GPE_MFP5	PE5_T1EX	GPE_MFP[5]
		0	0
		0	1
		1	1
[4:2]	Reserved		
[1]	PE.1 管脚功能选择 1 = PE.1 作为 PWM7 0 = PE.1 作为 GPIOE[1]		
[0]	PE.0 管脚功能选择 1 = PE.0 作为 PWM6 0 = PE.0 作为 GPIOE[0]		

复用多功能管脚控制寄存器 (ALT_MFP)

寄存器	偏移量	R/W	描述	复位后的值			
ALT_MFP	GCR_BA+0x50	R/W	复用多功能管脚控制寄存器	0x0000_0000			

31	30	29	28	27	26	25	24
Reserved			PB3_T3EX	PB2_T2EX	PE5_T1EX	PB15_T0EX	
23	22	21	20	19	18	17	16
EBI_HB_EN							
15	14	13	12	11	10	9	8
Reserved	EBI_nWRH_EN	EBI_nWRL_EN	EBI_MCLK_EN	EBI_EN	PB12_CLKO	PA15_I2SMC_LK	PC3_I2SDO
7	6	5	4	3	2	1	0
PC2_I2SDI	PC1_I2SBCLK	PC0_I2SLRCLK	PB11_PWM4	PB14_S31	PA7_S21	PB9_S11	PB10_S01

Bits	描述																									
[31:24]	Reserved 保留																									
[27]	PB3_T3EX GPB_MFP3、EBI_nWRH_EN (ALT_MFP[14])、EBI_EN (ALT_MFP[11]) 和 PB3_T3EX (ALT_MFP[27]) 决定了 PB.3 的功能。 <table border="1" style="margin-left: 20px;"> <tr> <td>EBI_nWRH_EN</td> <td>EBI_EN</td> <td>GPB_MFP[3]</td> <td>PB3_T3EX</td> <td>PB.3 功能</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>GPIO</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>CTS0 (UART0)</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>1</td> <td>nWRH (EBI 高字节写使能)</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>T3EX (TMR3)</td> </tr> </table>	EBI_nWRH_EN	EBI_EN	GPB_MFP[3]	PB3_T3EX	PB.3 功能	0	0	0	0	GPIO	0	0	1	1	CTS0 (UART0)	1	1	1	1	nWRH (EBI 高字节写使能)	0	0	1	1	T3EX (TMR3)
EBI_nWRH_EN	EBI_EN	GPB_MFP[3]	PB3_T3EX	PB.3 功能																						
0	0	0	0	GPIO																						
0	0	1	1	CTS0 (UART0)																						
1	1	1	1	nWRH (EBI 高字节写使能)																						
0	0	1	1	T3EX (TMR3)																						
[26]	PB2_T2EX GPB_MFP2、EBI_nWRL_EN (ALT_MFP[13])、EBI_EN (ALT_MFP[11]) 和 PB2_T2EX (ALT_MFP[26]) 决定了 PB.2 的功能。 <table border="1" style="margin-left: 20px;"> <tr> <td>EBI_nWRL_EN</td> <td>EBI_EN</td> <td>GPB_MFP[2]</td> <td>PB2_T2EX</td> <td>PB.2 功能</td> </tr> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> <td>GPIO</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>0</td> <td>RTS0 (UART0)</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> <td>0</td> <td>nWRL (EBI 低字节写使能)</td> </tr> <tr> <td>0</td> <td>0</td> <td>1</td> <td>1</td> <td>T2EX (TMR2)</td> </tr> </table>	EBI_nWRL_EN	EBI_EN	GPB_MFP[2]	PB2_T2EX	PB.2 功能	0	0	0	0	GPIO	0	0	1	0	RTS0 (UART0)	1	1	1	0	nWRL (EBI 低字节写使能)	0	0	1	1	T2EX (TMR2)
EBI_nWRL_EN	EBI_EN	GPB_MFP[2]	PB2_T2EX	PB.2 功能																						
0	0	0	0	GPIO																						
0	0	1	0	RTS0 (UART0)																						
1	1	1	0	nWRL (EBI 低字节写使能)																						
0	0	1	1	T2EX (TMR2)																						
[25]	PE5_T1EX GPE_MFP5 和 PE5_T1EX (ALT_MFP[25]) 决定了 PE.5 的功能。 <table border="1" style="margin-left: 20px;"> <tr> <td>PE5_T1EX</td> <td>GPE_MFP[5]</td> <td>PE.5 功能</td> </tr> <tr> <td>0</td> <td>0</td> <td>GPIO</td> </tr> </table>	PE5_T1EX	GPE_MFP[5]	PE.5 功能	0	0	GPIO																			
PE5_T1EX	GPE_MFP[5]	PE.5 功能																								
0	0	GPIO																								

		0	1	PWM5 (PWM)	
		1	1	T1EX (TMR1)	
[24]	PB15_T0EX	GPB_MFP15 和 PB15_T0EX (ALT_MFP[24]) 决定了 PB.15 的功能。			
		PB15_T0EX	GPB_MFP[15]	PB.15 功能	
		0	0	GPIO	
		0	1	/INT1	
		1	1	TOEX (TMR0)	
[23]	EBI_HB_EN[7]	EBI_HB_EN 用于将 GPIO 功能切换到 EBI 地址/数据总线高字节 (AD[15:8]) , EBI_HB_EN、EBI_EN 和相应的 GPx_MFP[y] 决定了 Px.y 的功能。 EBI_HB_EN[7]、EBI_EN 和 GPA_MFP[14] 决定了 PA.14 的功能。			
		EBI_HB_EN[7]	EBI_EN	GPA_MFP[14]	PA.14 功能
		0	0	0	GPIO
		0	0	1	PWM2 (PWM)
		1	1	1	AD15 (EBI AD bus bit 15)
[22]	EBI_HB_EN[6]	EBI_HB_EN[6]、EBI_EN 和 GPA_MFP[13] 决定了 PA.13 的功能。			
		EBI_HB_EN[6]	EBI_EN	GPA_MFP[13]	PA.13 功能
		0	0	0	GPIO
		0	0	1	PWM1 (PWM)
		1	1	1	AD14 (EBI AD bus bit 14)
[21]	EBI_HB_EN[5]	EBI_HB_EN[5]、EBI_EN 和 GPA_MFP[12] 决定了 PA.12 的功能。			
		EBI_HB_EN[5]	EBI_EN	GPA_MFP[12]	PA.12 功能
		0	0	0	GPIO
		0	0	1	PWM0 (PWM)
		1	1	1	AD13 (EBI AD bus bit 13)
[20]	EBI_HB_EN[4]	EBI_HB_EN[4]、EBI_EN 和 GPA_MFP[1] 决定了 PA.1 的功能。			
		EBI_HB_EN[4]	EBI_EN	GPA_MFP[1]	PA.1 功能
		0	0	0	GPIO
		0	0	1	ADC1 (ADC)
		1	1	1	AD12 (EBI AD bus bit 12)
[19]	EBI_HB_EN[3]	EBI_HB_EN[3]、EBI_EN 和 GPA_MFP[2] 决定了 PA.2 的功能。			
		EBI_HB_EN[3]	EBI_EN	GPA_MFP[2]	PA.2 功能
		0	0	0	GPIO
		0	0	1	ADC2 (ADC)

		1	1	1	AD11 (EBI AD bus bit 11)
[18]	EBI_HB_EN[2]	EBI_HB_EN[2]、EBI_EN 和 GPA_MFP[3] 决定了 PA.3 的功能。			
		EBI_HB_EN[2]	EBI_EN	GPA_MFP[3]	PA.3 功能
		0	0	0	GPIO
		0	0	1	ADC3 (ADC)
		1	1	1	AD10 (EBI AD bus bit 10)
[17]	EBI_HB_EN[1]	EBI_HB_EN[1]、EBI_EN 和 GPA_MFP[4] 决定了 PA.4 的功能。			
		EBI_HB_EN[1]	EBI_EN	GPA_MFP[4]	PA.4 功能
		0	0	0	GPIO
		0	0	1	ADC4 (ADC)
		1	1	1	AD9 (EBI AD bus bit 9)
[16]	EBI_HB_EN[0]	EBI_HB_EN[0]、EBI_EN 和 GPA_MFP[5] 决定了 PA.5 的功能。			
		EBI_HB_EN[0]	EBI_EN	GPA_MFP[5]	PA.5 功能
		0	0	0	GPIO
		0	0	1	ADC5 (ADC)
		1	1	1	AD8 (EBI AD bus bit 8)
[15]	Reserved	保留			
[14]	EBI_nWRH_EN	EBI_nWRH_EN、EBI_EN 和 GPB_MFP[3] 决定了 PB.3 的功能。			
		EBI_nWRH_EN	EBI_EN	GPB_MFP[3]	PB.3 功能
		0	0	0	GPIO
		0	0	1	CTS0 (UART0)
		1	1	1	nWRH (EBI 高字节写使能)
[13]	EBI_nWRL_EN	EBI_nWRL_EN、EBI_EN 和 GPB_MFP[2] 决定了 PB.2 的功能。			
		EBI_nWRL_EN	EBI_EN	GPB_MFP[2]	PB.2 功能
		0	0	0	GPIO
		0	0	1	RTS0 (UART0)
		1	1	1	nWRL (EBI 低字节写使能)
[12]	EBI_MCLK_EN	EBI_MCLK_EN、EBI_EN 和 GPC_MFP[8] 决定了 PC.8 功能。			
		EBI_MCLK_EN	EBI_EN	GPC_MFP[8]	PC.8 功能
		0	0	0	GPIO
		0	0	1	SPISS10 (SPI1)
		1	1	1	MCLK (EBI Clock output)
[11]	EBI_EN	EBI_EN 用于将 GPIO 功能切换到 EBI 功能 (AD[15:0], ALE, RE, WE, CS, MCLK)，并需要另外的寄存器 EBI_EN[7:0] 和 EBI_MCLK_EN 将 GPIO 切换到 EBI 功能			

(AD[15:8], MCLK)		
EBI_EN	GPA_MFP[6]	PA.6 功能
0	0	GPIO
0	1	ADC5 (ADC)
1	1	AD7 (EBI AD bus bit 7)
EBI_EN	GPA_MFP[7]	PA.7 功能
0	0	GPIO
0	1	ADC7 (ADC)
1	1	AD6 (EBI AD bus bit 6)
EBI_EN	GPC_MFP[7]	PC.7 功能
0	0	GPIO
0	1	CPN0 (CMP)
1	1	AD5 (EBI AD bus bit 5)
EBI_EN	GPC_MFP[6]	PC.6 功能
0	0	GPIO
0	1	CPP□(CMP)
1	1	AD4 (EBI AD bus bit 4)
EBI_EN	GPC_MFP[15]	PC.15 功能
0	0	GPIO
0	1	CPN1 (CMP)
1	1	AD3 (EBI AD bus bit 3)
EBI_EN	GPC_MFP[14]	PC.14 功能
0	0	GPIO
0	1	CPP1 (CMP)
1	1	AD2 (EBI AD bus bit 2)
EBI_EN	GPB_MFP[13]	PB.13 功能
0	0	GPIO
0	1	CPO1 (CIP)

		1	1	AD1 (EBI AD bus bit 1)
<hr/>				
	EBI_EN	PB12_CLKO	GPB_MFP[12]	PB.12 功能
	0	0	0	GPIO
	0	0	1	CPO0 (CMP)
	0	1	1	CLKO (Clock Driver output)
	1	1	1	AD0 (EBI AD bus bit 0)
<hr/>				
	EBI_EN	GPA_MFP[11]	PA.11 功能	
	0	0	GPIO	
	0	1	SCL1 (I ² C)	
	1	1	nRD (EBI)	
<hr/>				
	EBI_EN	GPA_MFP[10]	PA.10 功能	
	0	0	GPIO	
	0	1	SDA1 (I ² C)	
	1	1	nWR (EBI)	
<hr/>				
	EBI_EN	GPB_MFP[6]	PB.6 功能	
	0	0	GPIO	
	0	1	RTS1 (UART1)	
	1	1	ALE (EBI)	
<hr/>				
	EBI_EN	GPB_MFP[7]	PB.7 功能	
	0	0	GPIO	
	0	1	CTS1 (UART1)	
	1	1	nCS (EBI)	
[10]	PB12_CLKO	PB12_CLKO、GPB_MFP[12] 和 EBI_EN (ALT_MFP[11]) 决定了 PB.12 的功能。		
		EBI_EN	PB12_CLKO	GPB_MFP[12]
		0	0	0
		0	0	1
		0	1	1
		1	1	1
[9]	PA15_I2SMCLK	PA15_I2SMCLK 和 GPA_MFP[15] 决定了 PA.15 功能。		

		PA15_I2SMCLK	GPA_MFP[15]	PA.15 功能	
		0	0	GPIO	
		0		PWM3 (PWM)	
		1	1	I2SMCLK (I ² S)	
[8]	PC3_I2SDO	PC3_I2SDO 和 GPC_MFP[3] 决定了 PC.3 的功能。			
		PC3_I2SDO	GPC_MFP[3]	PC.3 功能	
		0	0	GPIO	
		0	1	MOSI00 (SPI0)	
		1	1	I2SDO (I ² S)	
[7]	PC2_I2SDI	PC2_I2SDI 和 GPC_MFP[2] 决定了 PC.2 的功能。			
		PC2_I2SDI	GPC_MFP[2]	PC.2 功能	
		0	0	GPIO	
		0	1	MISO00 (SPI0)	
		1	1	I2SDI (I ² S)	
[6]	PC1_I2SBCLK	PC1_I2SBCLK 和 GPC_MFP[1] 决定了 PC.1 的功能。			
		PC1_I2SBCLK	GPC_MFP[1]	PC.1 功能	
		0	0	GPIO	
		0	1	SPICLK0 (SPI0)	
		1	1	I2SBCLK (I ² S)	
[5]	PC0_I2SLRCLK	PC0_I2SLRCLK 和 GPC_MFP[0] 决定了 PC.0 的功能。			
		PC0_I2SLRCLK	GPC_MFP[0]	PC.0 功能	
		0	0	GPIO	
		0	1	SPISS00 (SPI0)	
		1	1	I2SLRCLK (I ² S)	
[4]	PB11_PWM4	PB11_PWM4 和 GPB_MFP[11] 决定了 PB.11 的功能。			
		PB11_PWM4	GPB_MFP[11]	PB.11 功能	
		0	0	GPIO	
		0	1	TM3	
		1	1	PWM4 (PWM)	
[3]	PB14_S31	PB14_S31 和 GPB_MFP[14] 决定了 PB.14 的功能。			
		PB14_S31	GPB_MFP[14]	PB.14 功能	
		0	0	GPIO	

		0	1	/INT0	
		1	1	SPISS31 (SPI3)	
[2]	PA7_S21	PA7_S21、GPA_MFP[7] 和 EBI_EN (ALT_MFP[11]) 决定了 PA.7 的功能。			
		EBI_EN	PA7_S21	GPA_MFP[7]	PA.7 功能
		0	0	0	GPIO
		0	0	1	ADC7 (ADC)
		0	1	1	SPISS21 (SPI2)
		1	0	1	AD6 (EBI AD bus bit 6)
[1]	PB9_S11	PB9_S11 和 GPB_MFP[9] 决定了 PB.9 的功能。			
		PB9_S11	GPB_MFP[9]	PB.9 功能	
		0	0	GPIO	
		0	1	TM1	
		1	1	SPISS11 (SPI1)	
[0]	PB10_S01	PB10_S01 和 GPB_MFP[10] 决定了 PB.10 的功能。			
		PB10_S01	GPB_MFP[10]	PB.10 功能	
		0	0	GPIO	
		0	1	TM2	
		1	1	SPISS01 (SPI0)	

寄存器写保护控制寄存器 (REGWRPROT)

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

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

该位寄存器用于禁用/使能保护寄存器，读取得到 REGPROTDIS 状态。

寄存器	偏移量	R/W	描述	复位后的值
REGWRPROT	GCR_BA+0x100	R/W	寄存器写保护控制寄存器	0x0000_0000

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

Bits	描述	
[31:16]	Reserved	保留
[7:0]	REGWRPROT	寄存器写保护码 (Write Only) 一些寄存器具有写保护功能。写这些保护寄存器必须通过依次写入“59h”，“16h”，“88h”到 REGWRPROT 解除锁定状态。这个时序完成之后，REGPROTDIS 位将被置1，写保护寄存器可以正常写入数据。
[0]	REGPROTDIS	寄存器写保护禁用标志 (Read only) 1 = 写保护已禁用。可以写入受保护寄存器 0 = 写保护已使能。不能写入受保护寄存器 受保护的寄存器有： IPRSTC1: 地址 0x5000_0008 BODCR: 地址 0x5000_0018 PORCR: 地址 0x5000_0024 PWRCON: 地址 0x5000_0200 (在电源唤醒中断被清时，bit[6]不受保护)



		<p>APBCLK bit[0]: 地址 0x5000_0208 (bit[0] 是看门狗时钟使能)</p> <p>CLKSEL0: 地址 0x5000_0210 (选择HCLK 和 CPU STCLK 时钟源)</p> <p>CLKSEL1 bit[1:0]: 地址 0x5000_0214 (用于看门狗时钟源选择)</p> <p>ISPCON: 地址 0x5000_C000 (Flash ISP 控制寄存器)</p> <p>WTCR: 地址 0x4000_4000</p> <p>FATCON: 地址 0x5000_C018</p>
--	--	---



5.2.6 系统定时器 (SysTick)

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

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

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

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

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



5.2.6.1 系统定时器控制寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
SCS_BA = 0xE000_E000				
SYST_CSR	SCS_BA+0x10	R/W	SysTick 控制与状态寄存器	0x0000_0000
SYST_RVR	SCS_BA+0x14	R/W	SysTick 重新加载值寄存器	0xFFFF_FFFF
SYST_CVR	SCS_BA+0x18	R/W	SysTick 当前值寄存器	0xFFFF_FFFF



5.2.6.2 系统定时器控制寄存器描述

SysTick 控制与状态寄存器 (SYST_CSR)

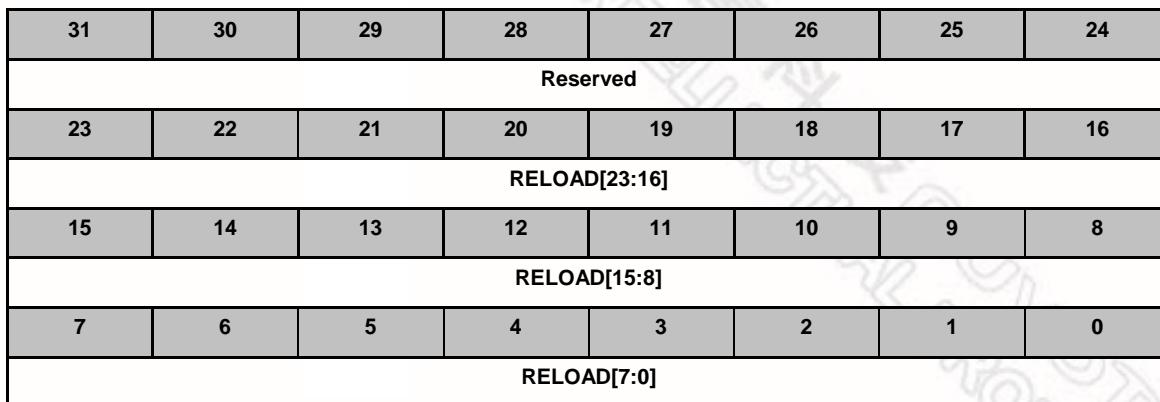
寄存器	偏移量	R/W	描述	复位后的值
SYST_CSR	SCS_BA+0x10	R/W	SysTick 控制与状态寄存器	0x0000_0000

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

Bits	描述	
[31:17]	Reserved	保留
[16]	COUNTFLAG	从上次读取该寄存器后，如果定时器计数到0，则返回1。 计数从1到0，COUNTFLAG 置位。 在读或者写当前值寄存器(SYST_CVR)时，COUNTFLAG 被清零。
[15:3]	Reserved	保留
[2]	CLKSRC	1 = 内核时钟用作 SysTick. 0 = 时钟源为外部参考时钟
[1]	TICKINT	1 = 向下计数到 0 将引起 SysTick 异常而挂起。软件清 SysTick 当前值寄存器 (SYST_CVR) 将不会导致 SysTick 挂起。 0 = 向下计数到 0 不会引起 SysTick 异常而挂起。软件根据 COUNTFLAG 来确定是否已经发生计数到 0。
[0]	ENABLE	1 = 计数器运行于连拍方式 (multi-shot manner) 0 = 禁用计数器

SysTick 重新加载值寄存器 (SYST_RVR)

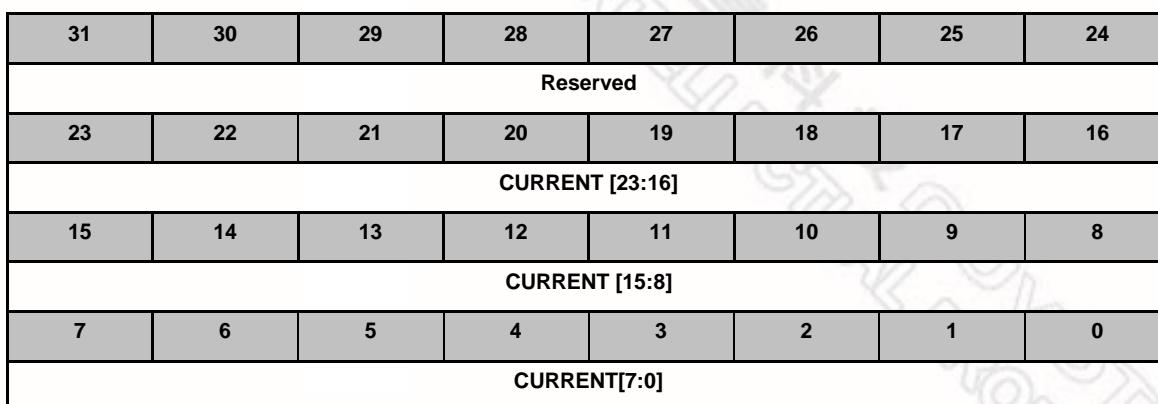
寄存器	偏移量	R/W	描述	复位后的值
SYST_RVR	SCS_BA+0x14	R/W	SysTick 重新加载值寄存器	0xFFFF_FFFF



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

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



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

5.2.7 嵌套向量中断控制器 (NVIC)

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

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

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

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

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

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

5.2.7.1 异常模式和系统中断映射

NuMicro™ NUC100 系列支持 表 5-2 所列的异常模式。与所有中断一样，软件可以对其中一些中断设置4级优先级。最高优先级为“0”，最低优先级为“3”，所有用户可配置的优先级的默认值为“0”。
注意：优先级为“0”在整个系统中为第4优先级，排在“Reset”，“NMI”与“Hard Fault”之后。

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

表 5-2 异常模式

向量号	中断号 (Bit in Interrupt Registers)	中断名称	源 IP	中断描述
0 ~ 15	-	-	-	系统异常
16	0	BOD_OUT	Brown-Out	欠压检测中断
17	1	WDT_INT	WDT	看门狗定时器中断
18	2	EINT0	GPIO	PB.14 管脚上的外部信号中断
19	3	EINT1	GPIO	PB.15 管脚上的外部信号中断
20	4	GPAB_INT	GPIO	PA[15:0]/PB[13:0] 的外部信号中断
21	5	GPCDE_INT	GPIO	PC[15:0]/PD[15:0]/PE[15:0] 的外部信号中断
22	6	PWMA_INT	PWM0~3	PWM0, PWM1, PWM2 与 PWM3 中断
23	7	PWMB_INT	PWM4~7	PWM4, PWM5, PWM6 与 PWM7 中断
24	8	TMR0_INT	TMR0	Timer 0 中断
25	9	TMR1_INT	TMR1	Timer 1 中断
26	10	TMR2_INT	TMR2	Timer 2 中断

27	11	TMR3_INT	TMR3	Timer 3 中断
28	12	UART02_INT	UART0/2	UART0 与 UART2 中断
29	13	UART1_INT	UART1	UART1 中断
30	14	SPI0_INT	SPI0	SPI0 中断
31	15	SPI1_INT	SPI1	SPI1 中断
32	16	SPI2_INT	SPI2	SPI2 中断
33	17	SPI3_INT	SPI3	SPI3 中断
34	18	I2C0_INT	I ² C0	I ² C0 中断
35	19	I2C1_INT	I ² C1	I ² C1 中断
36	20	CAN0_INT	CAN0	CAN0 中断
37	21	Reserved	Reserved	保留
38	22	Reserved	Reserved	保留
39	23	USB_INT	USBD	USB 2.0 FS 设备中断
40	24	PS2_INT	PS/2	PS/2 中断
41	25	ACMP_INT	ACMP	模拟比较器-0 或 模拟比较器-1 中断
42	26	PDMA_INT	PDMA	PDMA 中断
43	27	I2S_INT	I ² S	I ² S 中断
44	28	PWRWU_INT	CLKC	从掉电状态唤醒的时钟控制器中断
45	29	ADC_INT	ADC	ADC 中断
46	30	Reserved	Reserved	保留
47	31	RTC_INT	RTC	RTC 中断

表 5-3 系统中断映射

5.2.7.2 向量表

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

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

表 5-4 向量表格式

5.2.7.3 操作说明

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

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

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

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

5.2.7.4 NVIC 控制寄存器

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

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

IRQ0 ~ IRQ31 设置使能控制寄存器 (NVIC_ISER)

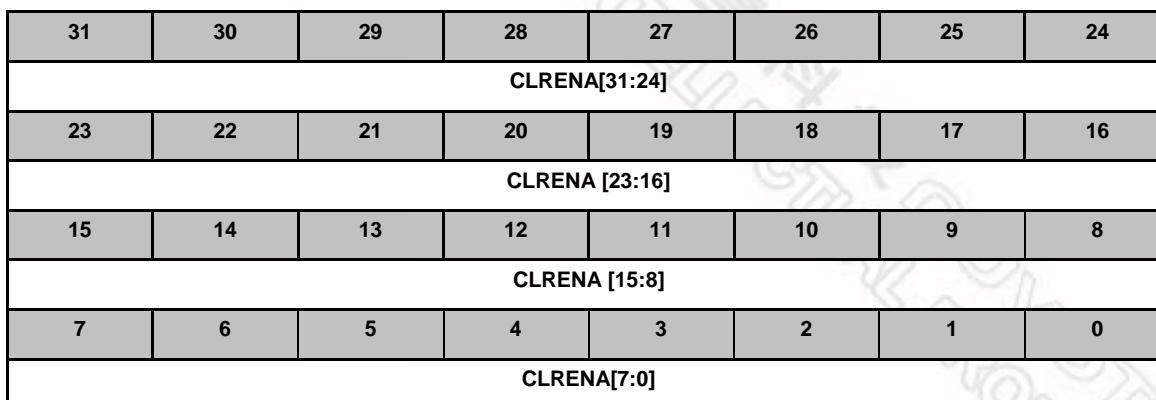
寄存器	偏移量	R/W	描述	复位后的值
NVIC_ISER	SCS_BA+0x100	R/W	IRQ0 ~ IRQ31 设置使能控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	SETENA	使能一个或者多个中断，每位代表 IRQ0 ~ IRQ31 的中断号（向量号：从16到47）。 写 1 使能相关中断 写 0 无效 读取该寄存器返回当前使能状态。

IRQ0 ~ IRQ31 清使能控制寄存器 (NVIC_ICER)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ICER	SCS_BA+0x180	R/W	IRQ0 ~ IRQ31 清使能控制寄存器	0x0000_0000



Bits	描述	
[31:0]	CLRENA	禁用一个或者多个中断，每位代表 IRQ0 ~ IRQ31 的中断号（向量号：从16到47）。 写 1 禁用相应中断 写 0 无效 读取该寄存器返回当前使能状态。

IRQ0 ~ IRQ31 设置挂起控制寄存器 (NVIC_ISPR)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ISPR	SCS_BA+0x200	R/W	IRQ0 ~ IRQ31 设置挂起控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	SETPEND	写 1, 由软件控制挂起相应中断。每位代表 IRQ0 ~ IRQ31 的中断号 (向量号: 从16到47)。 写 0 无效 读取该寄存器返回当前等待处理的中断状态。

IRQ0 ~ IRQ31 清挂起控制寄存器 (NVIC_ICPR)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_ICPR	SCS_BA+0x280	R/W	IRQ0 ~ IRQ31 清挂起控制寄存器	0x0000_0000

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

Bits	描述	
[31:0]	CLRPEND	写 1 清除, 由软件控制清除等待处理的中断, 每位代表 IRQ0 ~ IRQ31 的中断号 (向量号: 从16到47)。 写 0 无效 读取该寄存器返回当前等待处理的中断状态。

IRQ0 ~ IRQ3 中断优先级控制寄存器 (NVIC_IPR0)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR0	SCS_BA+0x400	R/W	IRQ0 ~ IRQ3 中断优先级控制寄存器	0x0000_0000

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

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

IRQ4 ~ IRQ7 中断优先级控制寄存器 (NVIC_IPR1)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR1	SCS_BA+0x404	R/W	IRQ4 ~ IRQ7 中断优先级控制寄存器	0x0000_0000

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

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

IRQ8 ~ IRQ11 中断优先级控制寄存器 (NVIC_IPR2)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR2	SCS_BA+0x408	R/W	IRQ8 ~ IRQ11 中断优先级控制寄存器	0x0000_0000

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

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
[29:24]	Reserved	保留
[23:22]	PRI_10	IRQ10 优先级 “0”表示最高优先级 & “3”表示最低优先级
[21:16]	Reserved	保留
[15:14]	PRI_9	IRQ9 优先级 “0”表示最高优先级 & “3”表示最低优先级
[13:8]	Reserved	保留
[7:6]	PRI_8	IRQ8 优先级 “0”表示最高优先级 & “3”表示最低优先级
[5:0]	Reserved	保留

IRQ12 ~ IRQ15 中断优先级控制寄存器 (NVIC_IPR3)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR3	SCS_BA+0x40C	R/W	IRQ12 ~ IRQ15 中断优先级控制寄存器	0x0000_0000

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

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

IRQ16 ~ IRQ19 中断优先级控制寄存器 (NVIC_IPR4)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR4	SCS_BA+0x410	R/W	IRQ16 ~ IRQ19 中断优先级控制寄存器	0x0000_0000

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

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

IRQ20 ~ IRQ23 中断优先级控制寄存器 (NVIC_IPR5)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR5	SCS_BA+0x414	R/W	IRQ20 ~ IRQ23 中断优先级控制寄存器	0x0000_0000

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

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

IRQ24 ~ IRQ27 中断优先级控制寄存器 (NVIC_IPR6)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR6	SCS_BA+0x418	R/W	IRQ24 ~ IRQ27 中断优先级控制寄存器	0x0000_0000

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

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

IRQ28 ~ IRQ31 中断优先级控制寄存器 (NVIC_IPR7)

寄存器	偏移量	R/W	描述	复位后的值
NVIC_IPR7	SCS_BA+0x41C	R/W	IRQ28 ~ IRQ31 中断优先级控制寄存器	0x0000_0000

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

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

5.2.7.5 中断源控制寄存器

除了 NVIC 相关的中断控制寄存器外，NuMicro™ NUC100 系列还有一些特殊寄存器便于中断处理，包括“中断源识别”，“NMI 源选择”与“中断测试模式”。描述如下：

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

寄存器	偏移量	R/W	描述	复位后的值
INT_BA = 0x5000_0300				
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) 中断源识别	0XXXX_XXXX
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 中断源识别	0XXXX_XXXX
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 中断源识别	0XXXX_XXXX
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 中断源识别	0XXXX_XXXX
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (GPA/B) 中断源识别	0XXXX_XXXX
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GPC/D/E) 中断源识别	0XXXX_XXXX
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) 中断源识别	0XXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (PWMB) 中断源识别	0XXXX_XXXX
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 中断源识别	0XXXX_XXXX
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 中断源识别	0XXXX_XXXX
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 中断源识别	0XXXX_XXXX
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 中断源识别	0XXXX_XXXX
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (URTO) 中断源识别	0XXXX_XXXX
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (URT1) 中断源识别	0XXXX_XXXX
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 中断源识别	0XXXX_XXXX
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) 中断源识别	0XXXX_XXXX
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (SPI2) 中断源识别	0XXXX_XXXX
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (SPI3) 中断源识别	0XXXX_XXXX
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I ² C0) 中断源识别	0XXXX_XXXX
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I ² C1) 中断源识别	0XXXX_XXXX
IRQ20_SRC	INT_BA+0x50	R	IRQ20 (CAN0) 中断源识别	0XXXX_XXXX
IRQ21_SRC	INT_BA+0x54	R	IRQ21 (保留) 中断源识别	0XXXX_XXXX
IRQ22_SRC	INT_BA+0x58	R	IRQ22 (保留) 中断源识别	0XXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (USB0) 中断源识别	0XXXX_XXXX

IRQ24_SRC	INT_BA+0x60	R	IRQ24 (PS/2) 中断源识别	0XXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (ACMP) 中断源识别	0XXXX_XXXX
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (PDMA) 中断源识别	0XXXX_XXXX
IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (I ² S) 中断源识别	0XXXX_XXXX
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) 中断源识别	0XXXX_XXXX
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (ADC) 中断源识别	0XXXX_XXXX
IRQ30_SRC	INT_BA+0x78	R	IRQ30 (保留) 中断源识别	0XXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	R	IRQ31 (RTC) 中断源识别	0XXXX_XXXX
NMI_SEL	INT_BA+0x80	R/W	NMI 中断源选择控制寄存器	0x0000_0000
MCU_IRQ	INT_BA+0x84	R/W	MCU 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	R	IRQ0 (BOD) 中断源识别 ：	0xFFFF_FFFF
	INT_BA+0x7C		IRQ31 (RTC) 中断源识别	

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

Bits	地址	INT-Num	描述
[2:0]	INT_BA+0x00	0	Bit2: 0 Bit1: 0 Bit0: BOD_INT
[2:0]	INT_BA+0x04	1	Bit2: 0 Bit1: 0 Bit0: WDT_INT
[2:0]	INT_BA+0x08	2	Bit2: 0 Bit1: 0 Bit0: EINT0 – PB.14 上的外部中断 0
[2:0]	INT_BA+0x0C	3	Bit2: 0 Bit1: 0 Bit0: EINT1 – PB.15 上的外部中断 1
[2:0]	INT_BA+0x10	4	Bit2: 0 Bit1: GPB_INT Bit0: GPA_INT
[2:0]	INT_BA+0x14	5	Bit2: GPE_INT Bit1: GPD_INT Bit0: GPC_INT
[3:0]	INT_BA+0x18	6	Bit3: PWM3_INT Bit2: PWM2_INT

			Bit1: PWM1_INT Bit0: PWM0_INT
[3:0]	INT_BA+0x1C	7	Bit3: PWM7_INT Bit2: PWM6_INT Bit1: PWM5_INT Bit0: PWM4_INT
[2:0]	INT_BA+0x20	8	Bit2: 0 Bit1: 0 Bit0: TMR0_INT
[2:0]	INT_BA+0x24	9	Bit2: 0 Bit1: 0 Bit0: TMR1_INT
[2:0]	INT_BA+0x28	10	Bit2: 0 Bit1: 0 Bit0: TMR2_INT
[2:0]	INT_BA+0x2C	11	Bit2: 0 Bit1: 0 Bit0: TMR3_INT
[2:0]	INT_BA+0x30	12	Bit2: 0 Bit1: 0 Bit0: URT0_INT
[2:0]	INT_BA+0x34	13	Bit2: 0 Bit1: 0 Bit0: URT1_INT
[2:0]	INT_BA+0x38	14	Bit2: 0 Bit1: 0 Bit0: SPI0_INT
[2:0]	INT_BA+0x3C	15	Bit2: 0 Bit1: 0 Bit0: SPI1_INT
[2:0]	INT_BA+0x40	16	Bit2: 0 Bit1: 0 Bit0: SPI2_INT
[2:0]	INT_BA+0x44	17	Bit2: 0 Bit1: 0 Bit0: SPI3_INT
[2:0]	INT_BA+0x48	18	Bit2: 0 Bit1: 0 Bit0: I2C0_INT

[2:0]	INT_BA+0x4C	19	Bit2: 0 Bit1: 0 Bit0: I2C1_INT
[2:0]	INT_BA+0x50	20	Bit2: 0 Bit1: 0 Bit0: CAN0_INT
[2:0]	INT_BA+0x54	21	保留
[2:0]	INT_BA+0x58	22	保留
[2:0]	INT_BA+0x5C	23	Bit2: 0 Bit1: 0 Bit0: USB_INT
[2:0]	INT_BA+0x60	24	Bit2: 0 Bit1: 0 Bit0: PS2_INT
[2:0]	INT_BA+0x64	25	Bit2: 0 Bit1: 0 Bit0: ACMP_INT
[2:0]	INT_BA+0x68	26	Bit2: 0 Bit1: 0 Bit0: PDMA_INT
[2:0]	INT_BA+0x6C	27	Bit2: 0 Bit1: 0 Bit0: I2S_INT
[2:0]	INT_BA+0x70	28	Bit2: 0 Bit1: 0 Bit0: PWRWU_INT
[2:0]	INT_BA+0x74	29	Bit2: 0 Bit1: 0 Bit0: ADC_INT
[2:0]	INT_BA+0x78	30	保留
[2:0]	INT_BA+0x7C	31	Bit2: 0 Bit1: 0 Bit0: RTC_INT

NMI 中断源选择控制寄存器 (NMI_SEL)

寄存器	偏移量	R/W	描述	复位后的值
NMI_SEL	INT_BA+0x80	R/W	NMI 中断源选择控制寄存器	0x0000_0000

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

Bits	描述	
[31:8]	Reserved	保留
[8]	NMI_EN	NMI 中断使能 1 = 使能 NMI 中断 0 = 禁用 NMI 中断
[7:5]	Reserved	保留
[4:0]	NMI_SEL	NMI 中断源选择 通过设置 NMI_SEL 可以在外围设备中断中选择 Cortex-M0 的 NMI 中断。

MCU 中断请求源寄存器 (MCU_IRQ)

寄存器	偏移量	R/W	描述	复位后的值
MCU_IRQ	INT_BA+0x84	R/W	MCU 中断请求源寄存器	0x0000_0000

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

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



5.2.8 系统控制寄存器

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

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

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

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

CPUID 寄存器 (CPUID)

寄存器	偏移量	R/W	描述	复位后的值
CPUID	SCS_BA+0xD00	R	CPUID 寄存器	0x410C_C200

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

Bits	描述	
[31:24]	IMPLEMENTER	ARM 分配的执行码 (ARM = 0x41)
[23:20]	Reserved	保留
[19:16]	PART	ARMv6-M 读取值 0xC
[15:4]	PARTNO	读取值 0xC20.
[3:0]	REVISION	读取值 0x0

中断控制和状态寄存器 (ICSR)

寄存器	偏移量	R/W	描述				复位后的值
ICSR	SCS_BA+0xD04	R/W	中断控制和状态寄存器				0x0000_0000

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

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

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



应用程序中断和复位寄存器 (AIRCR)

寄存器	偏移量	R/W	描述	复位后的值
AIRCR	SCS_BA+0xD0C	R/W	应用程序中断和复位寄存器	0xFA05_0000

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

Bits	描述	
[31:16]	VECTORKEY	写该寄存器时，写入值必须为 0x05FA，否则写动作将产生不可预测的结果。
[15:3]	Reserved	保留
[2]	SYSRESETREQ	向该位写 1，产生复位信号给芯片表示有复位请求。 该位为只写位，在复位时自动清零。
[1]	VECTCLRACTIVE	该位置 1，将清除所有固定及可配置异常的活动状态。 该位为只写位，且只在内核停止时可写。 注：调试器负责重新初始化堆栈。
[0]	Reserved	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+0xD10	R/W	系统控制寄存器	0x0000_0000

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

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

系统处理优先级寄存器2 (SHPR2)

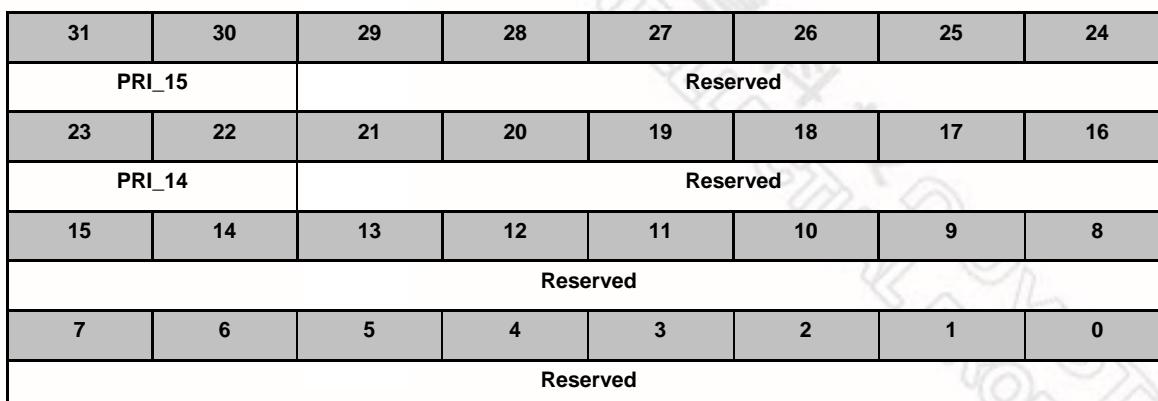
寄存器	偏移量	R/W	描述	复位后的值
SHPR2	SCS_BA+0xD1C	R/W	系统处理优先级寄存器 2	0x0000_0000

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

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

系统处理优先级寄存器 3 (SHPR3)

寄存器	偏移量	R/W	描述	复位后的值
SHPR3	SCS_BA+0xD20	R/W	系统处理优先级寄存器 3	0x0000_0000



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



5.3 时钟控制器

5.3.1 概述

时钟控制器为整个芯片提供时钟源，包括系统时钟和所有外围设备时钟，该控制器还通过个别时钟的开或关，时钟源选择和分频器来进行功耗控制。CPU 使能 PWR_DOWN_EN 位后，同时CPU Cortex-M0 内核执行 WFI 指令，芯片将进入掉电模式，直到唤醒中断发生，芯片才会退出掉电模式。在掉电模式下，时钟控制器关闭外部 4~24 MHz 晶振和内部 22.1184 MHz 振荡器，以降低整个系统的功耗。

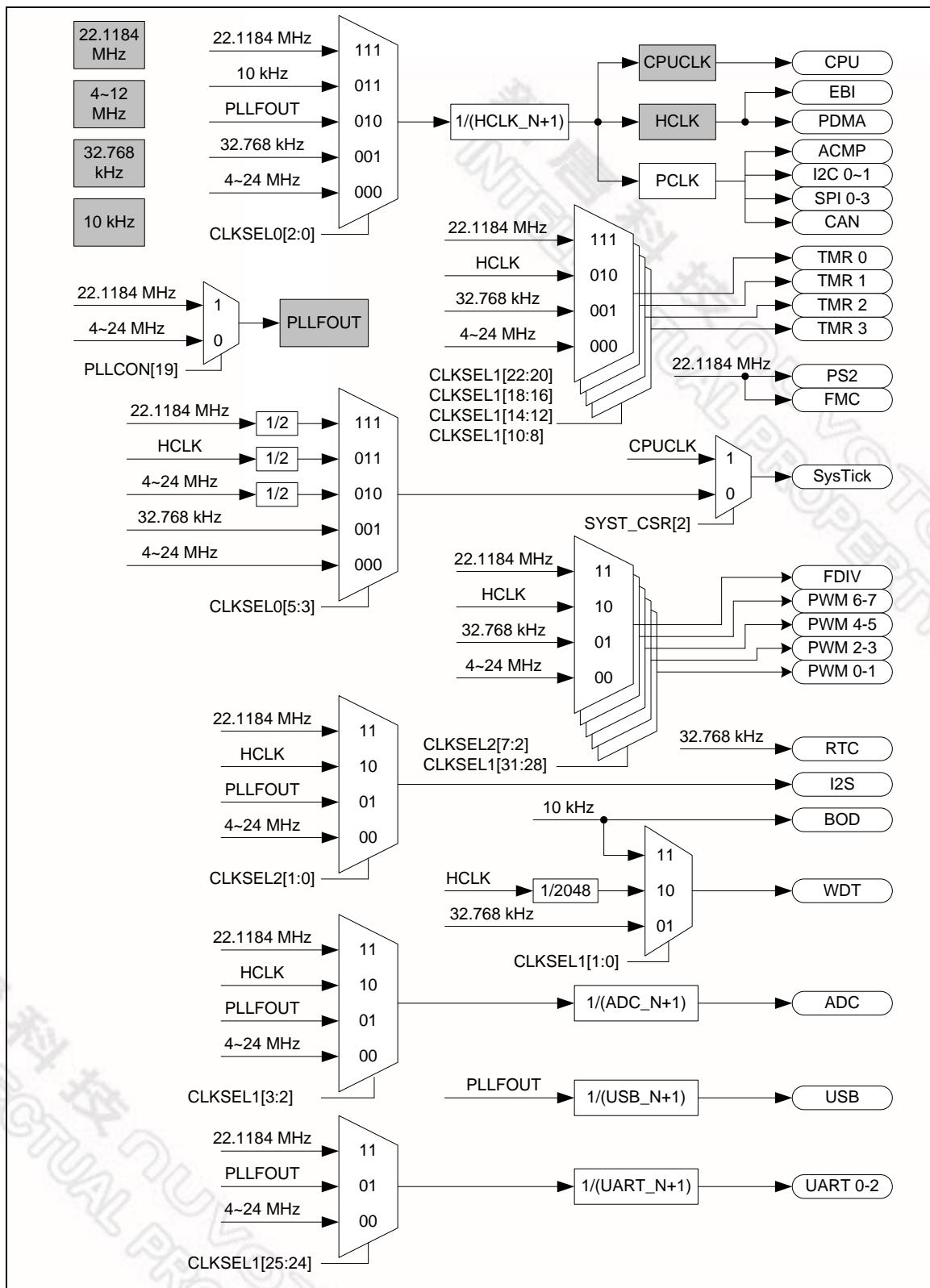


图 5-4 时钟发生器全局框图



新唐科技 NUVOTON
INTELLECTUAL PROPERTY

新唐科技 NUVOTON
INTELLECTUAL PROPERTY

5.3.2 时钟发生器

时钟发生器由如下 5 个时钟源组成：

- 一个外部 32.768 kHz 晶振
- 一个外部 4~24 MHz 晶振
- 一个可编程的 PLL FOUT (PLL 由外部 4~24 MHz 晶振和内部 22.1184 MHz 振荡器提供时钟源)
- 一个内部 22.1184 MHz 振荡器
- 一个内部 10 kHz 振荡器

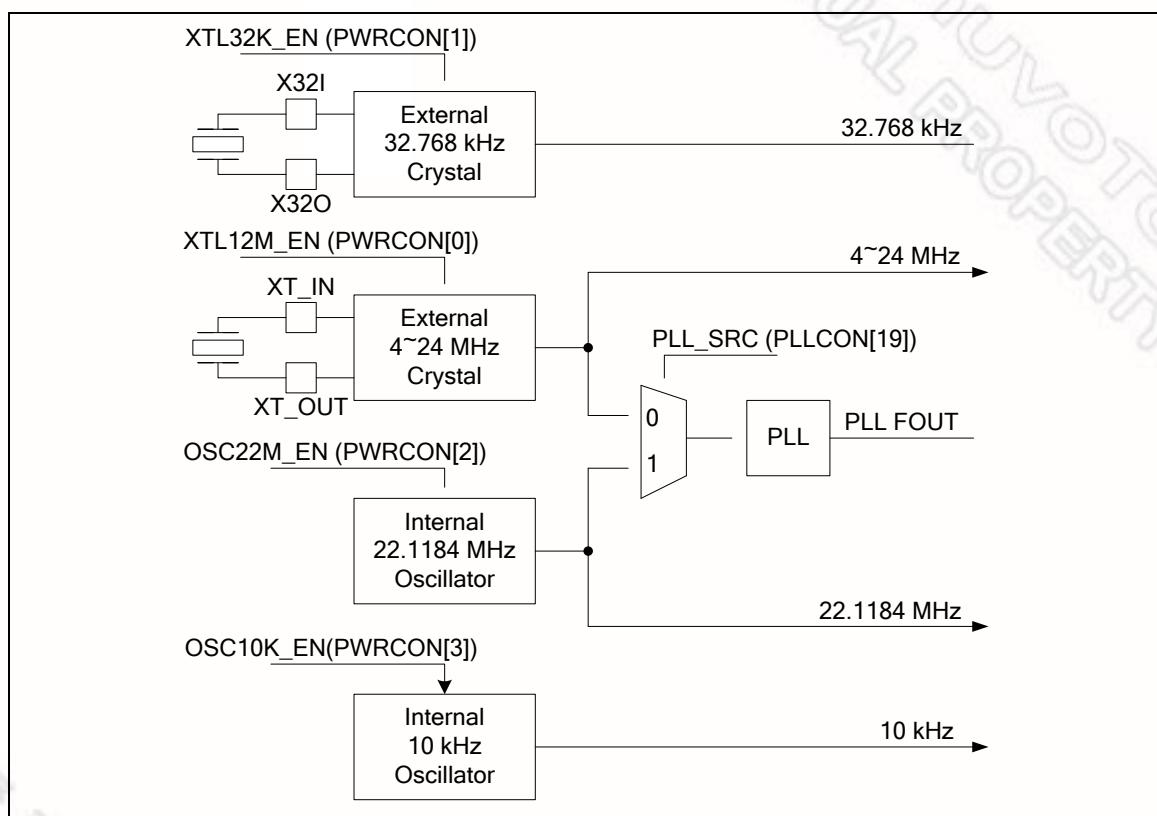


图 5-5 时钟发生器框图

5.3.3 系统时钟 & SysTick 时钟

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

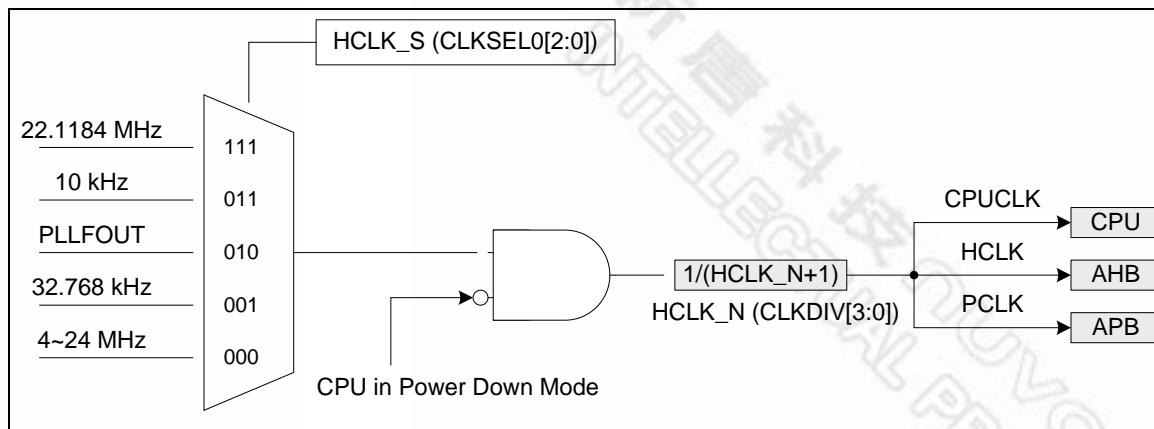


图 5-6 系统时钟框图

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

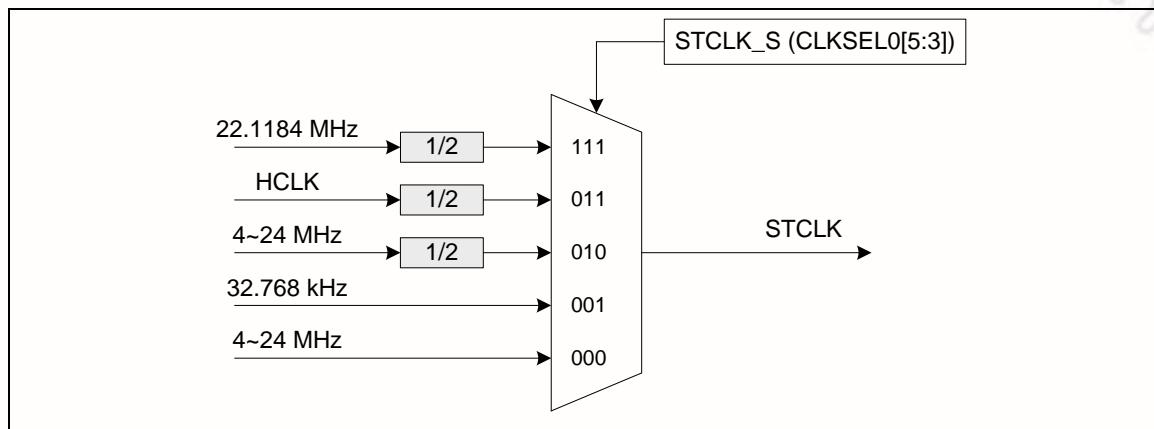


图 5-7 SysTick 时钟控制框图



5.3.4 外围设备时钟

不同的外设，其外围设备时钟有不同的时钟源切换。请参考 5.3.7 节的寄存器 CLKSEL1 和 CLKSEL2。

5.3.5 掉电模式时钟

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

如下时钟仍在工作：

- 时钟发生器
 - ◆ 内部 10 kHz 振荡器时钟
 - ◆ 外部 32.768 kHz 晶振时钟
- 外设时钟（当这些 IP 采用外部 32.768 kHz 晶振或者 10 kHz 振荡器作为时钟源时）

5.3.6 分频器输出

该器件包含一个由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，该多路转换器接到 CLKO 引脚上。因此有16种分频时钟选择，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} 为时钟分频器输出频率， N 为 FSEL (FRQDIV[3:0]) 中的4位值。

当 DIVIDER_EN (FRQDIV[4]) 置1时，分级计数器开始计数。当 DIVIDER_EN (FRQDIV[4]) 置0时，分级计数器持续计数直到分频时钟达到低电平并保持低电平。

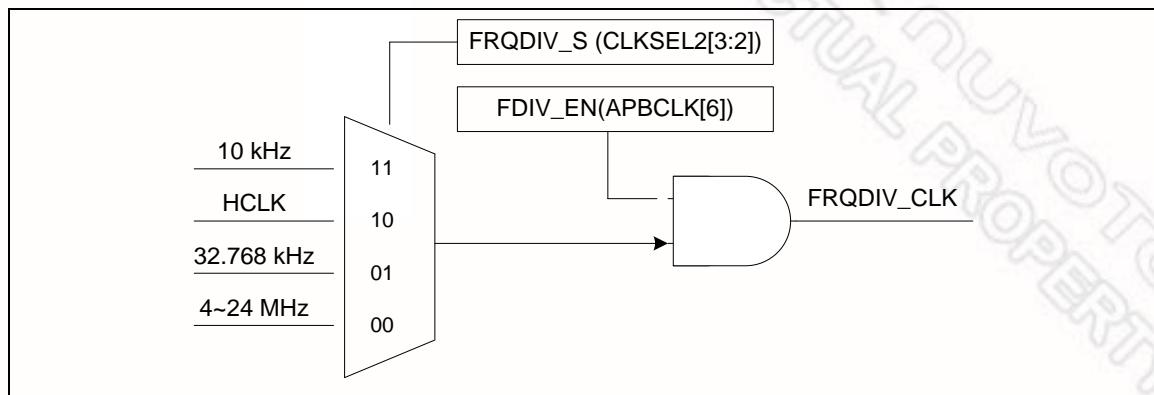


图 5-8 分频器的时钟源

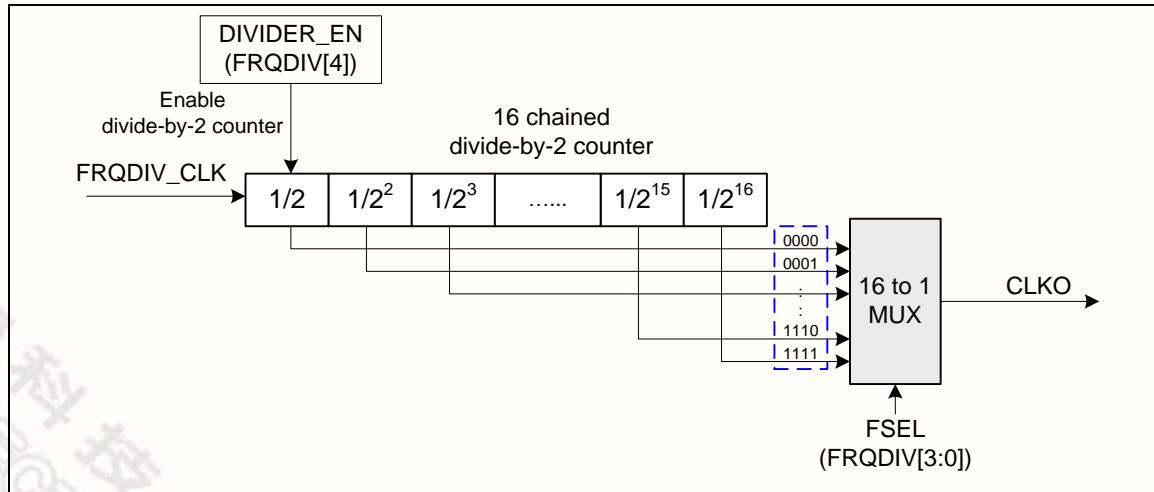


图 5-9 分频器的框图

5.3.7 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
CLK_BA = 0x5000_0200				
PWRCON	CLK_BA+0x00	R/W	系统掉电控制寄存器	0x0000_001X
AHBCLK	CLK_BA+0x04	R/W	AHB 设备时钟使能控制寄存器	0x0000_000D
APBCLK	CLK_BA+0x08	R/W	APB 设备时钟使能控制寄存器	0x0000_000X
CLKSTATUS	CLK_BA+0x0C	R/W	时钟状态监控寄存器	0x0000_00XX
CLKSEL0	CLK_BA+0x10	R/W	时钟源选择控制寄存器 0	0x0000_003X
CLKSEL1	CLK_BA+0x14	R/W	时钟源选择控制寄存器 1	0xFFFF_FFFF
CLKSEL2	CLK_BA+0x1C	R/W	时钟源选择控制寄存器 2	0x0000_00FF
CLKDIV	CLK_BA+0x18	R/W	时钟分频数目寄存器	0x0000_0000
PLLCON	CLK_BA+0x20	R/W	PLL 控制寄存器	0x0005_C22E
FRQDIV	CLK_BA+0x24	R/W	分频器控制寄存器	0x0000_0000



5.3.8 寄存器描述

掉电控制寄存器 (PWRCON)

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

寄存器	偏移量	R/W	描述	复位后的值
PWRCON	CLK_BA+0x00	R/W	系统掉电控制寄存器	0x0000_001X

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

Bits	描述	
[31:9]	Reserved	保留
[8]	PD_WAIT_CPU	<p>该位控制进入掉电模式的条件 (写保护位) 1 = 在 PD_WAIT_CPU 和 PWR_DOWN_EN 位都置 1 而且 CPU 执行 WFI 指令时, 芯片进入掉电模式。 0 = 在 PWR_DOWN_EN 位置 1 时, 芯片进入掉电模式。</p>
[7]	PWR_DOWN_EN	<p>系统掉电模式使能位 (写保护位) 该位置 ‘1’, 使能芯片的掉电模式, 激活芯片的掉电行为取决于 PD_WAIT_CPU 位。 (a) 如果 PD_WAIT_CPU 为 0, 则芯片会在置位 PWR_DOWN_EN 后, 立即进入掉电模式。 (b) 如果 PD_WAIT_CPU 为 1, 则芯片会一直运行到 CPU 的休眠模式被激活, 然后芯片才会进入掉电模式。 当芯片从掉电模式中被唤醒, 该位自动清零。用户需要重新设置该位才能使能下一次的掉电。 在掉电模式下, 外部 4~24 MHz 晶振与内部 22.1184 MHz 振荡器将被禁用, 但是外部 32.768 kHz 晶振与内部 10 kHz 振荡器不受掉电模式的控制。 在掉电模式下, PLL 与系统时钟将被禁用, 忽略时钟源选择。如果外设时钟源为外部 32.768 kHz 晶振或者内部 10 kHz 振荡器, 则外设的时钟不受掉电模式的控制。 1 = 芯片立即进入掉电模式或者等待 CPU 休眠命令 WFI 0 = 执行 WFI 命令, 芯片工作于正常模式或者芯片进入 idle 模式</p>
[6]	PD_WU_STS	芯片掉电模式唤醒中断标志

		该位由掉电唤醒事件设置，表示从掉电模式恢复。 如果GPIO, USB, UART, WDT, CAN, ACMP, BOD 或者 RTC 被唤醒，该标准置位。 写 1 该位清零。 注意：只有在 PD_WU_INT_EN (PWRCON[5]) 被置 1 的时候，该位才工作。
[5]	PD_WU_INT_EN	掉电模式唤醒中断使能 (写保护位) 0 = 禁用 1 = 使能 当 PD_WU_STS 和 PD_WU_INT_EN 都为高时，中断将产生。
[4]	PD_WU_DLY	使能唤醒延时计数器 (写保护位) 当芯片从掉电模式中被唤醒时，时钟控制将会延迟一定的时钟周期以等待系统时钟稳定。 当芯片工作在外部 4~24 MHz 晶振条件下时，延迟时钟周期为 4096 时钟周期；当芯片工作在内部 22.1184 MHz 振荡器条件下时，延迟时钟周期为 256 时钟周期。 1 = 使能时钟周期的延迟 0 = 禁用时钟周期的延迟
[3]	OSC10K_EN	内部 10 kHz 振荡器使能控制 (写保护位) 1 = 使能内部 10 kHz 振荡器 0 = 禁用内部 10 kHz 振荡器
[2]	OSC22M_EN	内部 22.1184 MHz 振荡器使能控制 (写保护位) 1 = 使能内部 22.1184 MHz 振荡器 0 = 禁用内部 22.1184 MHz 振荡器
[1]	XTL32K_EN	外部 32.768 kHz 晶振使能控制 (写保护位) 1 = 使能外部 32.768 kHz 晶振 (正常操作) 0 = 禁用外部 32.768 kHz 晶振
[0]	XTL12M_EN	外部 4~24 MHz 晶振使能控制 (写保护位) 该位的缺省值由 flash 控制器的用户配置寄存器 config0 [26:24] 设置。当缺省的时钟源为外部 4~24 MHz 晶振时，该位自动置 1。 1 = 使能外部 4~24 MHz 晶振 0 = 禁用外部 4~24 MHz 晶振



模式 寄存器/指令	PWR_DOWN_EN	PD_WAIT_CPU	CPU 运行 WFI 指令	禁用时钟
正常运行模式	0	0	NO	通过控制寄存器禁用所有时钟
Idle 模式 (CPU 进入休眠模式)	0	0	YES	仅禁用 CPU 时钟
掉电模式	1	0	NO	大部分时钟被禁用，仅 10 kHz/32.768 kHz 及 RTC/WDT/Timer/PWM 外设时钟可运行。
掉电模式 (CPU 进入深度休眠模式)	1	1	YES	大部分时钟被禁用，仅 10 kHz/32.768 kHz 及 RTC/WDT/Timer/PWM 外设时钟可运行。

表 5-5 掉电模式控制表

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

AHB 设备时钟使能控制寄存器 (AHBCLK)

该寄存器各位用于使能/禁用系统时钟, PDMA 时钟以及 EBI 时钟。

寄存器	偏移量	R/W	描述	复位后的值
AHBCLK	CLK_BA+0x04	R/W	AHB 设备时钟使能控制寄存器	0x0000_000D

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

Bits	描述	
[31:4]	Reserved	保留
[3]	EBI_EN	EBI 控制器时钟使能控制 1 = 使能 EBI engine 时钟 0 = 禁用 EBI engine 时钟
[2]	ISP_EN	Flash ISP 控制器时钟使能控制 1 = 使能 Flash ISP engine 时钟 0 = 禁用 Flash ISP engine 时钟
[1]	PDMA_EN	PDMA 控制器时钟使能控制 1 = 使能 PDMA engine 时钟 0 = 禁用 PDMA engine 时钟
[0]	Reserved	保留

APB 设备时钟使能控制寄存器 (APBCLK)

该寄存器各位用于使能/禁用外设控制器时钟。

寄存器	偏移量	R/W	描述	复位后的值
APBCLK	CLK_BA+0x08	R/W	APB 设备时钟使能控制寄存器	0x0000_000X

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

Bits	描述	
[31]	PS2_EN	PS/2 时钟使能控制 1 = 使能 PS/2 时钟 0 = 禁用 PS/2 时钟
[30]	ACMP_EN	模拟比较器时钟使能控制 1 = 使能模拟比较器时钟 0 = 禁用模拟比较器时钟
[29]	I2S_EN	I²S 时钟使能控制 1 = 使能 I ² S 时钟 0 = 禁用 I ² S 时钟
[28]	ADC_EN	ADC 时钟使能控制 1 = 使能 ADC 时钟 0 = 禁用 ADC 时钟
[27]	USBD_EN	USB 2.0 FS 设备控制器时钟使能控制 1 = 使能 USB 时钟 0 = 禁用 USB 时钟
[26:25]	Reserved	保留
[24]	CAN0_EN	CAN 总线控制器-0 时钟使能控制 1 = 使能 CAN0 时钟 0 = 禁用 CAN0 时钟

[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]	PWM01_EN	PWM_01 时钟使能控制 1 = 使能 PWM01 时钟 0 = 禁用 PWM01 时钟
[19]	Reserved	保留
[18]	UART2_EN	UART2 时钟使能控制 1 = 使能 UART2 时钟 0 = 禁用 UART2 时钟
[17]	UART1_EN	UART1 时钟使能控制 1 = 使能 UART1 时钟 0 = 禁用 UART1 时钟
[16]	UART0_EN	UART0 时钟使能控制 1 = 使能 UART0 时钟 0 = 禁用 UART0 时钟
[15]	SPI3_EN	SPI3 时钟使能控制 1 = 使能 SPI3 时钟 0 = 禁用 SPI3 时钟
[14]	SPI2_EN	SPI2 时钟使能控制 1 = 使能 SPI2 时钟 0 = 禁用 SPI2 时钟
[13]	SPI1_EN	SPI1 时钟使能控制 1 = 使能 SPI1 时钟 0 = 禁用 SPI1 时钟
[12]	SPI0_EN	SPI0 时钟使能控制 1 = 使能 SPI0 时钟 0 = 禁用 SPI0 时钟
[11:10]	Reserved	保留

[9]	I2C1_EN	I²C1 时钟使能控制 1 = 使能 I ² C1 时钟 0 = 禁用 I ² C1 时钟
[8]	I2C0_EN	I²C0 时钟使能控制 1 = 使能 I ² C0 时钟 0 = 禁用 I ² C0 时钟
[7]	Reserved	保留
[6]	FDIV_EN	分频器输出时钟使能控制 1 = 使能 FDIV 时钟 0 = 禁用 FDIV 时钟
[5]	TMR3_EN	Timer3 时钟使能控制 1 = 使能 Timer3 时钟 0 = 禁用 Timer3 时钟
[4]	TMR2_EN	Timer2 时钟使能控制 1 = 使能 Timer2 时钟 0 = 禁用 Timer2 时钟
[3]	TMR1_EN	Timer1 时钟使能控制 1 = 使能 Timer1 时钟 0 = 禁用 Timer1 时钟
[2]	TMR0_EN	Timer0 时钟使能控制 1 = 使能 Timer0 时钟 0 = 禁用 Timer0 时钟
[1]	RTC_EN	Real-Time-Clock APB 接口时钟使能控制 该位仅用于控制 RTC APB 时钟，RTC engine 的时钟源由外部 32.768 kHz 晶振提供。 1 = 使能 RTC 时钟 0 = 禁用 RTC 时钟
[0]	WDT_EN	看门狗定时器时钟使能控制 (写保护位) 该位是写保护位。这意味着编程该位时需要向地址 0x5000_0100 依次写入“59h”，“16h”，“88h”解除寄存器保护。详细操作请参考寄存器 REGWRPROT（地址：GCR_BA+0x100）。 1 = 使能看门狗定时器时钟 0 = 禁用看门狗定时器时钟

时钟状态寄存器 (CLKSTATUS)

该寄存器各位用于监控芯片时钟源是否稳定，时钟切换是否失败。

寄存器	偏移量	R/W	描述	复位后的值
CLKSTATUS	CLK_BA+0x0C	R/W	时钟状态监控寄存器	0x0000_00XX

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CLK_SW_FAIL	Reserved		OSC22M_STB	OSC10K_STB	PLL_STB	XTL32K_STB	XTL12M_STB

Bits	描述	
[31:8]	Reserved	保留
[7]	CLK_SW_FAIL	时钟切换失败标志 (写保护位) 1 = 时钟切换失败 0 = 时钟切换成功 该位在软件切换系统时钟源时更新。如果切换的目标时钟稳定，则该位被设置为 0，如果切换的目标时钟不稳定，则该位被设置为 1。 该位写1清零。
[6:5]	Reserved	保留
[4]	OSC22M_STB	内部 22.1184 MHz 振荡器时钟源稳定标志 1 = 内部 22.1184 MHz 振荡器时钟稳定 0 = 内部 22.1184 MHz 振荡器时钟不稳定或者被禁用 该位为只读位。
[3]	OSC10K_STB	内部 10 kHz 振荡器时钟源稳定标志 1 = 内部 10 kHz 振荡器时钟稳定 0 = 内部 10 kHz 振荡器时钟不稳定或者被禁用 该位为只读位。
[2]	PLL_STB	内部 PLL 时钟源稳定标志 1 = 内部 PLL 时钟稳定 0 = 内部 PLL 时钟不稳定或者被禁用

		该位为只读位。
[1]	XTL32K_STB	外部 32.768 kHz 晶振时钟源稳定标志 1 = 外部 32.768 kHz 晶振时钟稳定 0 = 外部 32.768 kHz 晶振时钟不稳定或者被禁用 该位为只读位。
[0]	XTL12M_STB	外部 4~24 MHz 晶振时钟源稳定标志 1 = 外部 4~24 MHz 晶振时钟稳定 0 = 外部 4~24 MHz 晶振时钟不稳定或者被禁用 该位为只读位。

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

Bits	描述	
[31:6]	Reserved	保留
[5:3]	STCLK_S	<p>Cortex_M0 SysTick 时钟源选择 (写保护位) 如果 SYST_CSR[2]=0, SysTick 使用下面列举的时钟源 该位是受保护位。编程时, 需要向地址 0x5000_0100 依次写入 “59h”, “16h”, “88h” 来解锁寄存器保护。请参考寄存器 REGWRPROT (地址: GCR_BA+0x100)。</p> <p>000 = 时钟源为外部 4~24 MHz 晶振时钟 001 = 时钟源为外部 32.768 kHz 晶振时钟 010 = 时钟源为外部 4~24 MHz 晶振时钟/2分频 011 = 时钟源为 HCLK/2分频 111 = 时钟源为内部 22.1184 MHz 振荡器时钟/2分频</p>
[2:0]	HCLK_S	<p>HCLK 时钟源选择 (写保护) 1. 在时钟切换前, 相关时钟源 (预选和新选) 必须打开。 2. 在任何复位后, 这 3 位的缺省值将从 Flash 控制器的用户配置寄存器 CFOSC (<u>Config0[26:24]</u>) 加载。因此缺省值可能为 000b 或者 111b。 3. 该位是受保护位。编程时, 需要向地址 0x5000_0100 依次写入 “59h”, “16h”, “88h” 来解锁寄存器保护。请参考寄存器 REGWRPROT (地址: GCR_BA+0x100)。</p> <p>000 = 时钟源为外部 4~24 MHz 晶振时钟 001 = 时钟源为外部 32.768 kHz 晶振时钟 010 = 时钟源为 PLL 时钟 011 = 时钟源为内部 10 kHz 振荡器时钟 111 = 时钟源为内部 22.1184 MHz 振荡器时钟</p>

时钟源选择控制寄存器 1 (CLKSEL1)

在时钟切换之前，必须打开相关的时钟源（预选和新选）。

寄存器	偏移量	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
PWM23_S	PWM01_S			Reserved		UART_S	
23	22	21	20	19	18	17	16
Reserved	TMR3_S			Reserved	TMR2_S		
15	14	13	12	11	10	9	8
Reserved	TMR1_S			Reserved	TMR0_S		
7	6	5	4	3	2	1	0
Reserved				ADC_S		WDT_S	

Bits	描述
[31:30]	PWM23_S PWM2 和 PWM3 时钟源选择 PWM2 和 PWM3 使用相同的时钟源和相同的分频。 00 = 时钟源为外部 4~24 MHz 晶振时钟 01 = 时钟源为外部 32.768 kHz 晶振时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 振荡器时钟
[29:28]	PWM01_S PWM0 和 PWM1时钟源选择 PWM0 和 PWM1使用相同的时钟源和相同的分频。 00 = 时钟源为外部 4~24 MHz 晶振时钟 01 = 时钟源为外部 32.768 kHz 晶振时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 振荡器时钟
[27:26]	Reserved
[25:24]	UART_S UART 时钟源选择 00 = 时钟源为外部 4~24 MHz 晶振时钟 01 = 时钟源为 PLL 时钟 11 = 时钟源为内部 22.1184 MHz 振荡器时钟
[23]	Reserved

[22:20]	TMR3_S	TIMER3 时钟源选择 000 = 时钟源为外部 4~24 MHz 晶振时钟 001 = 时钟源为外部 32.768 kHz 晶振时钟 010 = 时钟源为 HCLK 111 = 时钟源为内部 22.1184 MHz 振荡器时钟
[19]	Reserved	保留
[18:16]	TMR2_S	TIMER2 时钟源选择 000 = 时钟源为外部 4~24 MHz 晶振时钟 001 = 时钟源为外部 32.768 kHz 晶振时钟 010 = 时钟源为 HCLK 111 = 时钟源为内部 22.1184 MHz 振荡器时钟
[15]	Reserved	保留
[14:12]	TMR1_S	TIMER1 时钟源选择 000 = 时钟源为外部 4~24 MHz 晶振时钟 001 = 时钟源为外部 32.768 kHz 晶振时钟 010 = 时钟源为 HCLK 111 = 时钟源为内部 22.1184 MHz 振荡器时钟
[11]	Reserved	保留
[10:8]	TMR0_S	TIMER0 时钟源选择 000 = 时钟源为外部 4~24 MHz 晶振时钟 001 = 时钟源为外部 32.768 kHz 晶振时钟 010 = 时钟源为 HCLK 111 = 时钟源为内部 22.1184 MHz 振荡器时钟
[7:4]	Reserved	保留
[3:2]	ADC_S	ADC 时钟源选择 00 = 时钟源为外部 4~24 MHz 晶振时钟 01 = 时钟源来自 PLL 时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 振荡器时钟
[1:0]	WDT_S	看门狗定时器时钟源选择 (写保护位) 该位是受保护位。编程时，需要向地址 0x5000_0100 依次写入 “59h”, “16h”, “88h” 来解锁寄存器保护。请参考寄存器 REGWRPROT (地址: GCR_BA+0x100)。 01 = 时钟源为外部 32.768 kHz 晶振时钟 10 = 时钟源为 HCLK /2048 clock 11 = 时钟源为内部 10 kHz 振荡器时钟

时钟源选择控制寄存器 2 (CLKSEL2)

在时钟切换之前，必须打开相关的时钟源（预选和新选）。

寄存器	偏移量	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
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PWM67_S		PWM45_S		FRQDIV_S		I2S_S	

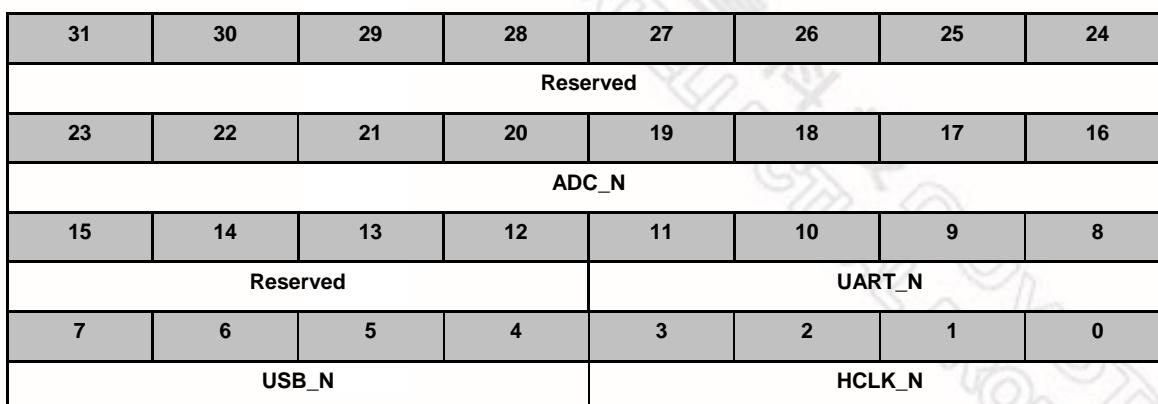
Bits	描述	
[31:8]	Reserved	保留
[7:6]	PWM67_S	PWM6 和 PWM7 时钟源选择 PWM6 和 PWM7 使用相同的时钟源和相同的分频。 00 = 时钟源为外部 4~24 MHz 晶振时钟 01 = 时钟源为外部 32.768 kHz 晶振时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 振荡器时钟
[5:4]	PWM45_S	PWM4 和 PWM5 时钟源选择 PWM4 和 PWM5 使用相同的时钟源和相同的分频。 00 = 时钟源为外部 4~24 MHz 晶振时钟 01 = 时钟源为外部 32.768 kHz 晶振时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 振荡器时钟
[3:2]	FRQDIV_S	时钟分频器时钟源选择 00 = 时钟源为外部 4~24 MHz 晶振时钟 01 = 时钟源为外部 32.768 kHz 晶振时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 振荡器时钟
[1:0]	I2S_S	I²S 时钟源选择



		00 = 时钟源为外部 4~24 MHz 晶振时钟 01 = 时钟源为 PLL 时钟 10 = 时钟源为 HCLK 11 = 时钟源为内部 22.1184 MHz 振荡器时钟
--	--	--

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



Bits	描述	
[31:24]	Reserved	保留
[23:16]	ADC_N	ADC 时钟源的时钟除频数 ADC 时钟频率 = (ADC 时钟源频率) / (ADC_N + 1)
[15:12]	Reserved	保留
[11:8]	UART_N	UART 时钟源的时钟除频数 UART 时钟频率= (UART 时钟源频率) / (UART_N + 1)
[7:4]	USB_N	USB 时钟由 PLL 时钟除频数 USB 时钟频率 = (PLL 频率) / (USB_N + 1)
[3:0]	HCLK_N	HCLK 时钟源的时钟除频数 HCLK 时钟频率 = (HCLK 时钟源频率) / (HCLK_N + 1)

PLL 控制寄存器 (PLLCON)

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

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

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

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



[8:0]	FB_DV	PLL 反馈分频控制管脚 参考下表公式
-------	--------------	------------------------

输出时钟频率设置

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

约束条件：

1. $3.2MHz < F_{IN} < 150MHz$
2. $800KHz < \frac{F_{IN}}{2 * NR} < 8MHz$
3. $100MHz < F_{CO} = F_{IN} \times \frac{NF}{NR} < 200MHz$
 $120MHz < F_{CO}$ is preferred

符号	描述
F_{OUT}	输出时钟频率
F_{IN}	输入（参考）时钟频率
NR	输入分频 ($IN_DV + 2$)
NF	反馈分频 ($FB_DV + 2$)
NO	$OUT_DV = "00"$: $NO = 1$ $OUT_DV = "01"$: $NO = 2$ $OUT_DV = "10"$: $NO = 2$ $OUT_DV = "11"$: $NO = 4$

默认频率设置

默认值： 0xC22E

$F_{IN} = 12 MHz$

$NR = (1+2) = 3$

$NF = (46+2) = 48$

$NO = 4$

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



频率分频器控制寄存器 (FRQDIV)

寄存器	偏移量	R/W	描述	复位后的值
FRQDIV	CLK_BA+ 24	R/W	频率分频控制寄存器	0x0000_0000

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

Bits	描述	
[31:5]	Reserved	保留
[4]	DIVIDER_EN	<p>频率分频器使能位 0 = 禁用频率分频器 1 = 使能频率分频器</p>
[3:0]	FSEL	<p>分频器输出频率选择位 输出频率的公式: $F_{out} = F_{in}/2^{(N+1)}$ F_{in} 为输入时钟频率 F_{out} 为分频器输出时钟频率 N 为 FSEL[3:0] 的4位值。</p>

5.4 USB 设备控制器 (USB)

5.4.1 概述

该器件有一组 USB 2.0 全速设备控制器和收发器，符合 USB 2.0 全速设备规范，支持 control/bulk/interrupt/ isochronous 传输类型。

在该设备控制器中，包含两个主接口：APB 总线和来自 USB PHY 收发器的 USB 总线。CPU 能够通过 APB 总线编程控制寄存器。在该控制器中内置有 512 字节的 SRAM 作为数据缓存。输入或输出传输，需要通过 APB 接口或 SIE 向 SRAM 写数据或从 SRAM 读数据。用户需要通过缓存分段寄存器 (BUFSEGx) 为每个端点缓存设置有效的 SRAM 地址。

USB 设备控制器共有 6 个可配置的端点。每个端点可以配置为 IN 或者 OUT 类型。所有的操作包括 Control, Bulk, Interrupt 和 Isochronous transfer 传输都由端点模块来执行。端点控制模块可以还用来管理数据同步时序，端点状态控制，当前起始地址，当前事务状态和每个端点的数据缓存状态。

该控制器有 4 种不同的中断事件，分别是唤醒功能，设备插拔事件，USB 事件（如 IN ACK, OUT ACK 等）和 BUS 事件（如 suspend 和 resume 等）。任何时间都将会引发一个中断，用户只需要在中断事件状态寄存器 (USB_INTSTS) 中检查相关事件标志以获知发生何种中断，然后检测相关的 USB 端点状态寄存器 (USB_EPSTS) 以获知在该端点上发生何种事件。

USB 设备控制器有一个软件禁用功能，用于模拟设备从主机分离的情况。如果用户使能 DRVSE0 位 (USB_DRVSE0)，USB 控制器将使 USB_DP 和 USB_DM 输出低电平禁止其功能。在禁用 DRVSE0 位之后，主机将重新枚举 USB 设备。

参考文献：通用串行总线规范修订版 1.1

5.4.2 特征

该通用串行总线 (USB) 为一个带有单独连接器的串行接口，可以连接所有 USB 外设到主机系统。下面是 USB 的一些特征。

- 兼容 USB 2.0 全速规范
- 提供 1 个中断向量，4 个中断事件 (WAKEUP, FLDET, USB and BUS)
- 支持 Control/Bulk/Interrupt/Isochronous 传输类型
- 支持在没有总线活动超过 3 ms 之后的暂停功能
- 为可配置的 Control/Bulk/Interrupt/Isochronous 传输类型提供 6 个端点和最大 512 字节的缓存
- 提供远程唤醒功能

5.4.3 框图

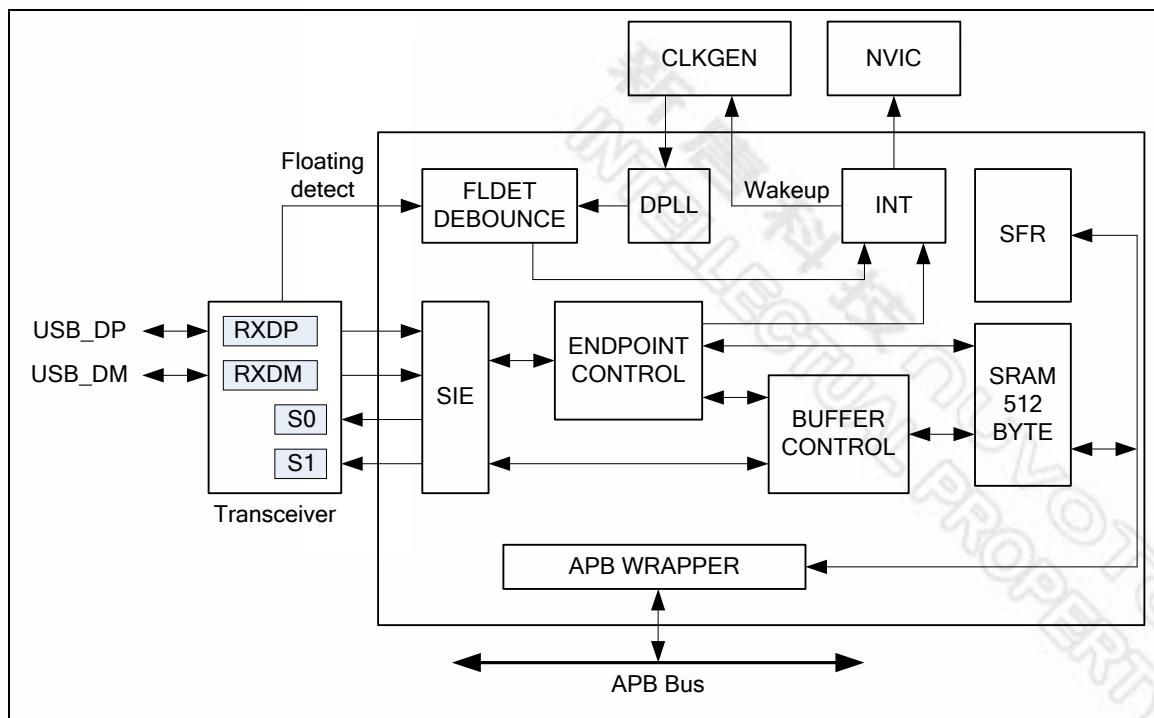


图 5-10 USB 框图



5.4.4 功能描述

5.4.4.1 SIE (串行接口引擎)

SIE 是设备控制器的前端，处理大多数 USB 协议包。SIE 一般包括向上发送信号到事务处理层面。处理功能包括：

- 包识别，事务排序
- SOP, EOP, RESET, RESUME 信号检测/产生
- Clock/Data 分离
- NRZI 数据编/解码与比特填塞 (bit-stuffing)
- CRC 产生和校验 (for Token 和 Data)
- Packet ID (PID) 产生和校验/解码
- 串-并/ 并-串 转换

5.4.4.2 端点控制

该控制器有 6 端点。每个端点可以配置成 Control, Bulk, Interrupt, 或 Isochronous 传输类型。所有操作包括 Control, Bulk, Interrupt 和 Isochronous 传输都在该模块内执行。该模块还可以用来管理数据同步时序，端点状态控制，当前端点起始地址，当前事务状态和每个端点的数据缓存状态。

5.4.4.3 数字锁相环

USB 数据的比特率为 12 MHz。DPLL 采用由时钟控制器产生的 48 MHz 频率来锁定 RXDP 和 RXDM 的输入数据。12 MHz 的比特率时钟也是由 DPLL 转换而来。

5.4.4.4 插拔去抖动

USB 设备可以进行热插拔操作，为了监测USB设备被拔出时的状态，设备控制器提供了硬件去抖动以防止在 USB 插拔时产生的抖动问题，悬空检测中断产生于 USB 进行插拔操作的10 ms后，用户可以通过读取“USB_FLDET”寄存器的值，来应答 USB 的插拔。“FLDET”标志代表当前总线上没有经过去抖动处理的状态。如果 FLDET 为 1，则意味着控制器已经插入 USB。如果用户要通过这个标志轮询来检测 USB 的状态，则必要时需要添加软件去抖动功能。

5.4.4.5 中断

该 USB 提供 1 个中断信号有 4 个中断事件的中断向量 (WAKEUP, FLDET, USB and BUS)。WAKEUP 事件被用来在掉电模式下唤醒系统时钟。（掉电模式功能在系统掉电控制寄存器 PWRCON 中定义）。FLDET 事件用于 USB 插拔。USB 事件告知用户一些 USB 请求，如 IN ACK, OUT ACK 等。BUS 事件告知用户一些总线时间，如中止、恢复等。用户必须在 USB 设备控制器的中断使能寄存器 (USB_INTEN) 设置相应的位以使能 USB 中断。

唤醒中断只会出现在芯片进入掉电模式且唤醒事件发生时。在芯片进入掉电模式后，USB_DP 和 USB_DM 的任何变化都能唤醒芯片（假定 USB 唤醒功能被使能）。若这个变化不是有意而为的，那么只有唤醒中断会发生。若 USB 唤醒超过 20 ms 后，没有其他 USB 中断事件发生，唤醒中断将发生。下图为唤醒中断的控制流程。

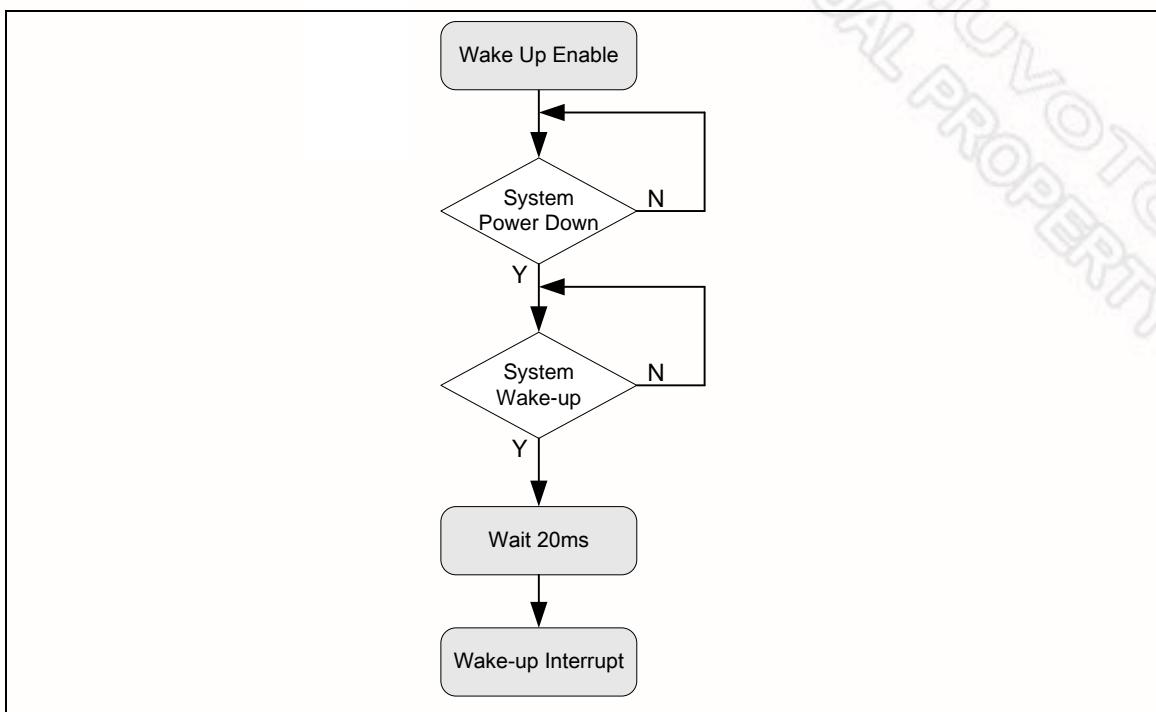


图 5-11 唤醒中断的操作流程

USB 中断告知用户总线上的所有 USB 事件，用户可以通过读特殊功能寄存器 EPSTS (USB_EPSTS[25:8]) 和 EPEVT5~0 (USB_INTSTS[21:16]) 来获知是哪类请求，对哪个端点采取必要的响应。

与 USB 中断相似，BUS 中断告知用户一些总线事件，如 USB 复位、中止、超时和恢复。用户可以读特殊功能寄存器 USB_ATTR 获取总线事件。

5.4.4.6 省电

当芯片进入掉电模式时，USB 自动关闭 PHY 收发器省电。此外，在特殊环境下，用户可以给特殊功能寄存器 USB_ATTR[4] 写入“0”关闭 PHY 进入省电状态。

5.4.4.7 缓冲控制

USB 控制器有 512 字节的 SRAM，6个端点共享该缓冲区。在 USB 功能有效之前，用户需要在缓冲段寄存器配置每个端点的有效起始地址。BUFFER CONTROL 模块用来控制每个端点的有效起始地址和 SRAM 的大小（在寄存器 MXPLD 定义）。

图 5-12 根据 BUFSEG 和 MXPLD 寄存器的内容描述每个端点的起始地址。如果 BUFSEG0 设置为 0x08h，MXPLD0 设置为 0x40h，则端点0的SRAM 大小从 USB_BA+0x108h 开始，到 USB_BA+0x148h 结束。（注：USB SRAM 的基地址为 USB_BA+0x100h）。

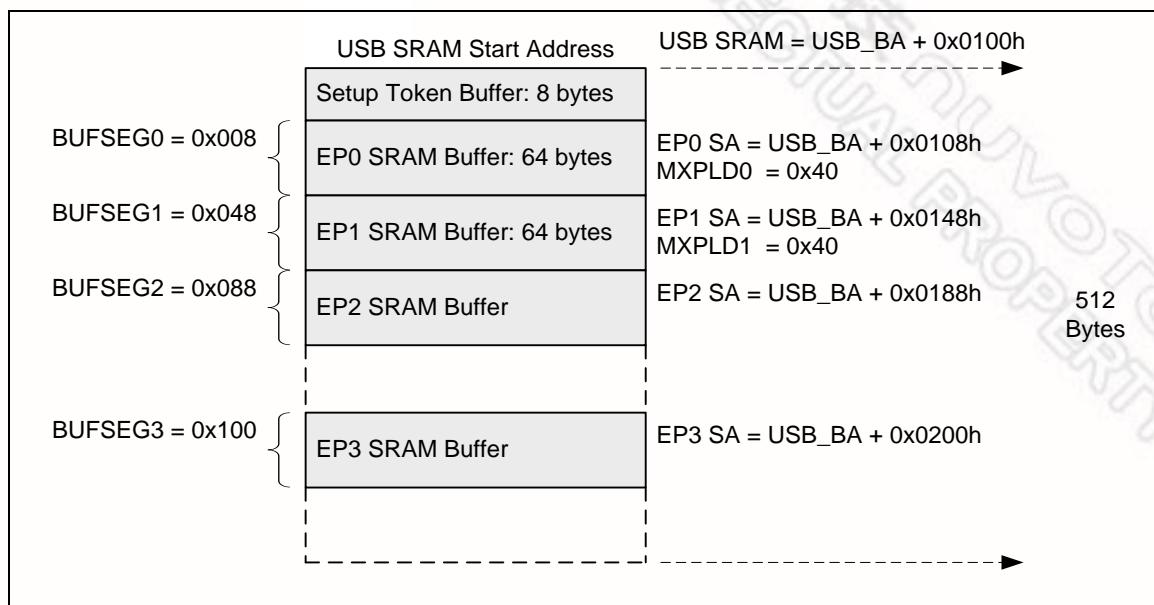


图 5-12 端点 SRAM 的结构

5.4.4.8 与 USB 外设通信处理

用户可以采用中断或轮询检测 `USB_INTSTS` 来监测 USB 通信，在 USB 通信发生时，`USB_INTSTS` 由硬件设置，并向 CPU 发送中断请求（如果相关中断使能），在没有中断时，用户可以轮询检测 `USB_INTSTS` 来获取事件，以下是控制流的中断使能。

USB 主机向设备控制器请求数据时，用户需要预先准备相关的数据到指定的端点缓存。在将数据写入缓冲区后，用户需要写入实际数据长度到指定的 MAXPLD 寄存器。一旦这个寄存器被写入数据，内部信号“`In_Rdy`”会被设置，当收到主机发送的相关 IN token 之后，缓冲数据将被立刻传送。在传送指定数据之后，信号“`In_Rdy`”会由硬件自动清除。

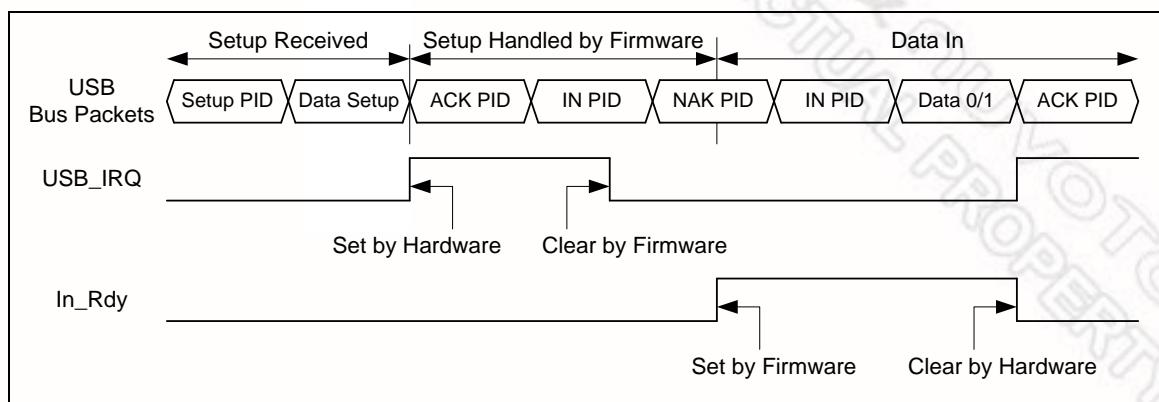


图 5-13 数据传入事务紧跟 Setup 事务

USB 主机要发送数据到设备控制器的 OUT 端点，硬件将这些数据存在指定的端点缓存里，通信完成后，硬件在相关的 MAXPLD 寄存器记录数据长度，并清除“`Out_Rdy`”信号，这将会避免硬件在用户没有取走当前数据时接收下一个数据。一旦用户处理了这次通信时，由软件写入相关的寄存器“MAXPLD”来设置“`Out_Rdy`”信号以接收下一次通信。

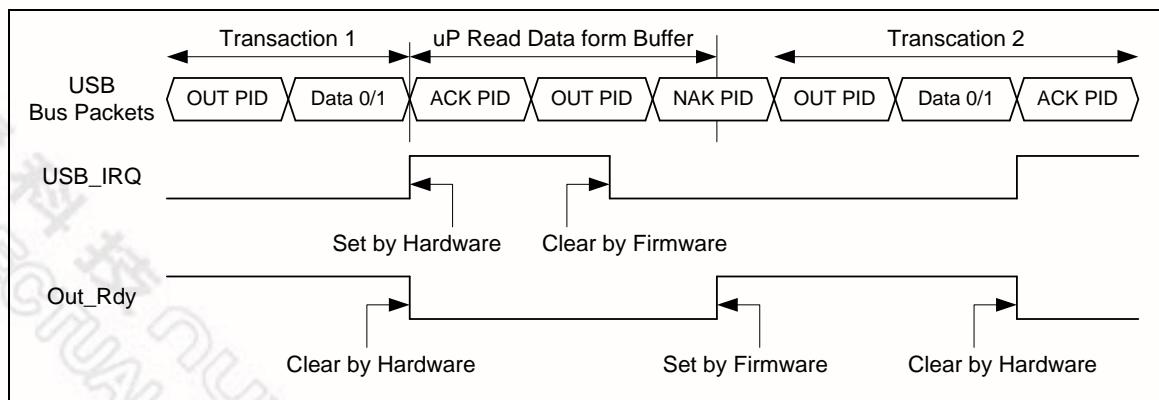


图 5-14 数据输出图

5.4.5 寄存器与内存映射

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

寄存器	偏移量	R/W	描述	复位后的值
USB_BA = 0x4006_0000				
USB_INTEN	USB_BA+0x000	R/W	USB 中断使能寄存器	0x0000_0000
USB_INTSTS	USB_BA+0x004	R/W	USB 中断事件状态寄存器	0x0000_0000
USB_FADDR	USB_BA+0x008	R/W	USB 设备功能地址寄存器	0x0000_0000
USB_EPSTS	USB_BA+0x00C	R	USB 端点状态寄存器	0x0000_00x0
USB_ATTR	USB_BA+0x010	R/W	USB 总线状态与属性寄存器	0x0000_0040
USB_FLDET	USB_BA+0x014	R	USB 悬空检测寄存器	0x0000_0000
USB_BUFSEG	USB_BA+0x018	R/W	Setup 令牌 (Token) 缓存段寄存器	0x0000_0000
USB_BUFSEG0	USB_BA+0x020	R/W	端点 0 缓存段寄存器	0x0000_0000
USB_MXPLD0	USB_BA+0x024	R/W	端点 0 最大有效负荷寄存器	0x0000_0000
USB_CFG0	USB_BA+0x028	R/W	端点 0 配置寄存器	0x0000_0000
USB_CFGP0	USB_BA+0x02C	R/W	端点 0 设置延迟与清除 In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG1	USB_BA+0x030	R/W	端点 1 缓存段寄存器	0x0000_0000
USB_MXPLD1	USB_BA+0x034	R/W	端点 1 最大有效负荷寄存器	0x0000_0000
USB_CFG1	USB_BA+0x038	R/W	端点 1 配置寄存器	0x0000_0000
USB_CFGP1	USB_BA+0x03C	R/W	端点 1 设置延迟与清除 In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG2	USB_BA+0x040	R/W	端点 2 缓存段寄存器	0x0000_0000
USB_MXPLD2	USB_BA+0x044	R/W	端点 2 最大有效负荷寄存器	0x0000_0000
USB_CFG2	USB_BA+0x048	R/W	端点 2 配置寄存器	0x0000_0000
USB_CFGP2	USB_BA+0x04C	R/W	端点 2 设置延迟与清除 In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG3	USB_BA+0x050	R/W	端点 3 缓存段寄存器	0x0000_0000
USB_MXPLD3	USB_BA+0x054	R/W	端点 3 最大有效负荷寄存器	0x0000_0000
USB_CFG3	USB_BA+0x058	R/W	端点 3 配置寄存器	0x0000_0000
USB_CFGP3	USB_BA+0x05C	R/W	端点 3 设置延迟与清除 In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG4	USB_BA+0x060	R/W	端点 4 缓存段寄存器	0x0000_0000
USB_MXPLD4	USB_BA+0x064	R/W	端点 4 最大有效负荷寄存器	0x0000_0000

USB_CFG4	USB_BA+0x068	R/W	端点 4 配置寄存器	0x0000_0000
USB_CFGP4	USB_BA+0x06C	R/W	端点 4 设置延迟与清除 In/Out 准备控制寄存器	0x0000_0000
USB_BUFSEG5	USB_BA+0x070	R/W	端点 5 缓存段寄存器	0x0000_0000
USB_MXPLD5	USB_BA+0x074	R/W	端点 5 最大有效负荷寄存器	0x0000_0000
USB_CFG5	USB_BA+0x078	R/W	端点 5 配置寄存器	0x0000_0000
USB_CFGP5	USB_BA+0x07C	R/W	端点 5 设置延迟与清除 In/Out 准备控制寄存器	0x0000_0000
USB_DRVSE0	USB_BA+0x090	R/W	USB 驱动 SE0 控制寄存器	0x0000_0001

内存类型	地址	大小	描述
USB_BA = 0x4006_0000			
SRAM	USB_BA+0x100 ~ USB_BA+0x2FF	512 Bytes	SRAM 用于整个端点缓存 参考章节 5.4.4.7 的端点 SRAM 结构和描述



5.4.6 寄存器描述

USB 中断使能寄存器 (USB_INTEN)

寄存器	偏移量	R/W	描述	复位后的值
USB_INTEN	USB_BA+0x000	R/W	USB 中断使能寄存器	0x0000_0000

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

Bits	描述	
[31:16]	Reserved	保留
[15]	INNAK_EN	激活收到 IN Token 时的 NAK 的功能及其状态 1 = 当该位置 1 且收到 IN token 时应答 NAK, NAK 的状态被更新到端点状态寄存器 USB_EPSTS, 将发生 INNAK 中断事件。 0 = 当它置 0 并且收到 IN token 应答 NAK 时, NAK 状态不会被更新到端点状态寄存器 USB_EPSTS, 不会产生 INNAK 中断事件。
[14:9]	Reserved	保留
[8]	WAKEUP_EN	唤醒功能使能控制 1 = 使能 USB 唤醒 CPU 功能 0 = 禁用 USB 唤醒 CPU 功能
[7:4]	Reserved	保留
[3]	WAKEUP_IE	USB 唤醒 CPU 中断使能 1 = 使能唤醒 CPU 中断 0 = 禁用唤醒 CPU 中断
[2]	FLDET_IE	悬空检测中断使能 1 = 使能悬空检测中断 0 = 禁用悬空检测中断
[1]	USB_IE	USB 事件中断使能 1 = 使能 USB 事件中断 0 = 禁用 USB 事件中断



[0]	BUS_IE	总线事件中断使能 1 = 使能总线事件中断 0 = 禁用总线事件中断
-----	---------------	--

USB 中断事件状态寄存器 (USB_INTSTS)

该寄存器是 USB 中断事件状态寄存器，通过向相应位写 ‘1’ 实现清零。

寄存器	偏移量	R/W	描述	复位后的值
USB_INTSTS	USB_BA+0x004	R/W	USB 中断事件状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
SETUP	Reserved						
23	22	21	20	19	18	17	16
Reserved		EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				WAKEUP_STS	FLDET_STS	USB_STS	BUS_STS

Bits	描述	
[31]	SETUP	Setup 事件状态 1 = Setup 事件发生，向 USB_INTSTS[31] 写 1 清标志 0 = 无 Setup 事件
[30:22]	Reserved	保留
[21]	EPEVT5	端点 5 的 USB 事件状态 1 = 端点 5 上的 USB 事件，检查 USB_EPSTS[25:23] 可知发生了何种 USB 事件，向 USB_INTSTS[21] 或者 USB_INTSTS[1] 写 1 清零。 0 = 端点 5 没有事件发生
[20]	EPEVT4	端点 4 的 USB 事件状态 1 = 端点 4 上的 USB 事件，检查 USB_EPSTS[22:20] 可知发生了何种 USB 事件，向 USB_INTSTS[20] 或者 USB_INTSTS[1] 写 1 清零。 0 = 端点 4 没有事件发生
[19]	EPEVT3	端点 3 的 USB 事件状态 1 = 端点 3 上的 USB 事件，检查 USB_EPSTS[19:17] 可知发生了何种 USB 事件，向 USB_INTSTS[19] 或者 USB_INTSTS[1] 写 1 清零。 0 = 端点 3 没有事件发生
[18]	EPEVT2	端点 2 的 USB 事件状态 1 = 端点 2 上的 USB 事件，检查 USB_EPSTS[16:14] 可知发生了何种 USB 事件，向 USB_INTSTS[18] 或者 USB_INTSTS[1] 写 1 清零。 0 = 端点 2 没有事件发生

[17]	EPEVT1	端点 1 的 USB 事件状态 1 = 端点 1 上的 USB 事件，检查 USB_EPSTS[13:11] 可知发生了何种 USB 事件，向 USB_INTSTS[17] 或者 USB_INTSTS[1] 写 1 清零。 0 = 端点 1 没有事件发生
[16]	EPEVT0	端点 0 的 USB 事件状态 1 = 端点 0 上的 USB 事件，检查 USB_EPSTS[10:8] 可知发生了何种 USB 事件，向 USB_INTSTS[16] 或者 USB_INTSTS[1] 写 1 清零。 0 = 端点 0 没有事件发生
[15:4]	Reserved	保留
[3]	WAKEUP_STS	唤醒中断状态 1 = 唤醒事件发生，向 USB_INTSTS[3] 写 1 清零 0 = 无唤醒事件发生
[2]	FLDET_STS	悬空检测中断状态 1 = USB 总线上有连接/分离事件，向 USB_INTSTS[2] 写 1 清零 0 = USB 总线上无连接/分离事件
[1]	USB_STS	USB 事件中断状态 USB 事件包括 Setup Token, IN Token, OUT ACK, ISO IN, or ISO OUT 1 = USB 事件发生，检查 EPSTS0~5[2:0] 可知发生了何种 USB 事件，向 USB_INTSTS[1] 或者 EPSTS0~5 以及 SETUP (USB_INTSTS[31]) 写 1 清零 0 = 无 USB 事件发生
[0]	BUS_STS	总线中断状态 总线事件意味着总线上存在着暂停或者恢复功能。 1 = 总线事件发生；检查 USB_ATTR[3:0] 可知发生了何种总线事件，向 USB_INTSTS[0] 写 1 清零。 0 = 无总线事件发生



USB 设备功能地址寄存器 (USB_FADDR)

在 USB 总线上作为设备地址的 7 位值。

寄存器	偏移量	R/W	描述	复位后的值
USB_FADDR	USB_BA+0x008	R/W	USB 设备功能地址寄存器	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	FADDR						

Bits	描述	
[31:7]	Reserved	保留
[6:0]	FADDR	USB 设备功能地址

USB 端点状态寄存器 (USB_EPSTS)

寄存器	偏移量	R/W	描述	复位后的值
USB_EPSTS	USB_BA+0x00C	R	USB 端点状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						EPSTS5[2:1]	
23	22	21	20	19	18	17	16
EPSTS5[0]	EPSTS4[2:0]			EPSTS3[2:0]			EPSTS2[2]
15	14	13	12	11	10	9	8
EPSTS2[1:0]		EPSTS1[2:0]			EPSTS0[2:0]		
7	6	5	4	3	2	1	0
OVERRUN	Reserved						

Bits	描述	
[31:26]	Reserved	保留
[25:23]	EPSTS5	<p>端点 5 总线状态 这些位用来表示该端点的当前状态。</p> <p>000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end</p>
[22:20]	EPSTS4	<p>端点 4 总线状态 这些位用来表示该端点的当前状态。</p> <p>000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end</p>
[19:17]	EPSTS3	<p>端点 3 总线状态 这些位用来表示该端点的当前状态。</p> <p>000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK</p>

		110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[16:14]	EPSTS2	<p>端点 2 总线状态 这些位用来表示该端点的当前状态。</p> <p>000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end</p>
[13:11]	EPSTS1	<p>端点 1 总线状态 这些位用来表示该端点的当前状态。</p> <p>000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end</p>
[10:8]	EPSTS0	<p>端点 0 总线状态 这些位用来表示该端点的当前状态。</p> <p>000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end</p>
[7]	OVERRUN	<p>Overrun 表示接收到数据是否大于有效负载最大值。</p> <p>1 = 表明主机 Out Data 多于 MXPLD 寄存器中定义的 Max Payload 或 Setup Data 多于 8 字节 0 = 没有溢出</p>
[6:0]	Reserved	保留

USB 总线状态与属性寄存器 (USB_ATTR)

寄存器	偏移量	R/W	描述	复位后的值
USB_ATTR	USB_BA+0x010	R/W	USB 总线状态与属性寄存器	0x0000_0040

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					BYTEM	PWRDN	DPPU_EN
7	6	5	4	3	2	1	0
USB_EN	Reserved	RWAKEUP	PHY_EN	TIMEOUT	RESUME	SUSPEND	USRST

Bits	描述	
[31:11]	Reserved	保留
[10]	BYTEM	CPU 访问 USB SRAM 字节大小模式选择 1 = Byte Mode: 从 CPU 到 USB SRAM 传输的大小只能是 Byte 0 = Word Mode: 从 CPU 到 USB SRAM 传输的大小只能是 Word
[9]	PWRDN	PHY 收发器掉电, 低电平有效 1 = 打开 PHY 收发器的相关电路 0 = 关闭 PHY 收发器的相关电路
[8]	DPPU_EN	使能 USB_DP 的上拉电阻 1 = USB_DP 总线上的上拉电阻有效 0 = 禁用 USB_DP 总线上的上拉电阻
[7]	USB_EN	USB 控制器使能控制 1 = 使能 USB 控制器 0 = 禁用 USB 控制器
[6]	Reserved	保留
[5]	RWAKEUP	远程唤醒 1 = 使 USB 总线到 K 状态 (USB_DP low, USB_DM: high), 用于远程唤醒 0 = 将 USB 总线从 K 状态释放
[4]	PHY_EN	PHY 收发器功能使能控制 1 = 使能 PHY 收发器功能 0 = 禁用 PHY 收发器功能

[3]	TIMEOUT	超时状态 1 = 总线无响应超过 18 bits 0 = 没有超时 该位为只读位。
[2]	RESUME	恢复状态 1 = 从暂停状态恢复 0 = 无总线恢复 该位为只读位。
[1]	SUSPEND	暂停状态 1 = 总线空闲超过 3ms, 或线缆拔出或主机休眠 0 = 总线没有暂停 该位为只读位。
[0]	USBRST	USB 复位状态 1 = 当 SE0 (single-ended 0) 超过 2.5us, 总线复位 0 = 总线没有复位 该位为只读位。

悬空检测寄存器 (USB_FLDET)

寄存器	偏移量	R/W	描述	复位后的值
USB_FLDET	USB_BA+0x014	R	USB 悬空检测寄存器	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							FLDET

Bits	描述	
[31:1]	Reserved	保留
[0]	FLDET	设备悬空检测 1 = USB 控制器 接入总线时，该位被置 1。 0 = USB 控制器没有接入总线。

缓存段寄存器 (USB BUFSEG)

仅用于 Setup token。

寄存器	偏移量	R/W	描述	复位后的值
USB_BUFSEG	USB_BA+0x018	R/W	Setup Token 缓存段寄存器	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
BUFSEG[7:3]					Reserved		

Bits	描述	
[31:9]	Reserved	保留
[8:3]	BUFSEG	<p>该部分用来表示 Setup Token 相对 USB SRAM 起始地址的偏移量。 Setup token 在 USB SRAM 中的有效起始地址是 $USB_SRAM\ 地址 + \{ BUFSEG[8:3], 3'b000 \}$ $USB_SRAM\ 地址 = USB_BA+0x100h.$ 注：只用于 Setup token。</p>
[2:0]	Reserved	保留

缓存段寄存器 (BUFSEGx) x = 0~5

寄存器	偏移量	R/W	描述	复位后的值
USB_BUFSEG0	USB_BA+0x020	R/W	端点 0 缓存段寄存器	0x0000_0000
USB_BUFSEG1	USB_BA+0x030	R/W	端点 1 缓存段寄存器	0x0000_0000
USB_BUFSEG2	USB_BA+0x040	R/W	端点 2 缓存段寄存器	0x0000_0000
USB_BUFSEG3	USB_BA+0x050	R/W	端点 3 缓存段寄存器	0x0000_0000
USB_BUFSEG4	USB_BA+0x060	R/W	端点 4 缓存段寄存器	0x0000_0000
USB_BUFSEG5	USB_BA+0x070	R/W	端点 5 缓存段寄存器	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							BUFSEG[8]x
7	6	5	4	3	2	1	0
BUFSEG[7:3]x					Reserved		

Bits	描述	
[31:9]	Reserved	保留
[8:3]	BUFSEGx	该部分用来表示每个端点相对USB SRAM 起始地址的偏移地址。端点有效的起始地址是 USB_SRAM 地址 + { BUFSEG[8:3], 3'b000 } USB_SRAM 地址 = USB_BA+0x100h. 参考章节 5.4.4.7 中关于端点 SRAM 结构与描述。
[2:0]	Reserved	保留

最大有效负载寄存器 (USB_MXPLDx) x = 0~5

寄存器	偏移量	R/W	描述	复位后的值
USB_MXPLD0	USB_BA+0x024	R/W	端点 0 最大有效负载寄存器	0x0000_0000
USB_MXPLD1	USB_BA+0x034	R/W	端点 1 最大有效负载寄存器	0x0000_0000
USB_MXPLD2	USB_BA+0x044	R/W	端点 2 最大有效负载寄存器	0x0000_0000
USB_MXPLD3	USB_BA+0x054	R/W	端点 3 最大有效负载寄存器	0x0000_0000
USB_MXPLD4	USB_BA+0x064	R/W	端点 4 最大有效负载寄存器	0x0000_0000
USB_MXPLD5	USB_BA+0x074	R/W	端点 5 最大有效负载寄存器	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							MXPLD[8]
7	6	5	4	3	2	1	0
MXPLD[7:0]							

Bits	描述	
[31:9]	Reserved	保留
[8:0]	MXPLD	<p>最大有效负载</p> <p>用于定义发送到主机 (IN token) 的数据长度或从主机接收到的 (OUT token) 实际数据长度。也用于表示端点是否准备好发送 (IN token) 或接受 (OUT token)。</p> <p>(1). CPU 写该寄存器</p> <p>IN token, MXPLD 的值用于定义要发送到数据长度并表示数据缓存以已经准备好了。</p> <p>OUT token, 表示控制器准备接收主机的数据, MXPLD 的值表示从主机发送过来的最大数据长度。</p> <p>(2). CPU 读该寄存器</p> <p>IN token, MXPLD 的值表示发生到主机的数据长度</p> <p>OUT token, MXPLD 的值表示从主机接收到的实际数据长度</p> <p>注: 一旦 MXPLD 被写, 数据包将在 IN/OUT token 到达后立即发送/接收。</p>

配置寄存器 (USB_CFGx) x = 0~5

寄存器	偏移量	R/W	描述	复位后的值
USB_CFG0	USB_BA+0x028	R/W	端点 0 的配置寄存器	0x0000_0000
USB_CFG1	USB_BA+0x038	R/W	端点 1 的配置寄存器	0x0000_0000
USB_CFG2	USB_BA+0x048	R/W	端点 2 的配置寄存器	0x0000_0000
USB_CFG3	USB_BA+0x058	R/W	端点 3 的配置寄存器	0x0000_0000
USB_CFG4	USB_BA+0x068	R/W	端点 4 的配置寄存器	0x0000_0000
USB_CFG5	USB_BA+0x078	R/W	端点 5 的配置寄存器	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						CSTALL	Reserved
7	6	5	4	3	2	1	0
DSQ_SYNC	STATE		ISOCH	EP_NUM			

Bits	描述	
[31:10]	Reserved	保留
[9]	CSTALL	清除 STALL 响应 1 = 在 setup 阶段允许设备清除 STALL 0 = 在 setup 阶段禁止设备清除 STALL
[8]	Reserved	保留
[7]	DSQ_SYNC	数据时序同步 1 = DATA1 PID 0 = DATA0 PID 指明在接下来的 IN token 传输中是 DATA0 或 DATA1 PID。H/W 将基于该位自动触发 IN token。
[6:5]	STATE	端点状态 00 = 端点禁用 01 = 输出端点 10 = 输入端点 11 = 无定义

[4]	ISOCH	Isochronous 端点 该位用来设置端点为 Isochronous 端点，无握手 1 = Isochronous 端点 0 = 非 Isochronous 端点
[3:0]	EP_NUM	端点号 用于定义当前端点的端点号。

额外配置寄存器 (USB_CFGPx) x = 0~5

寄存器	偏移量	R/W	描述	复位后的值
USB_CFGP0	USB_BA+0x02C	R/W	端点 0 设置 Stall 与清除 In/Out 准备控制寄存器	0x0000_0000
USB_CFGP1	USB_BA+0x03C	R/W	端点 1 设置 Stall 与清除 In/Out 准备控制寄存器	0x0000_0000
USB_CFGP2	USB_BA+0x04C	R/W	端点 2 设置 Stall 与清除 In/Out 准备控制寄存器	0x0000_0000
USB_CFGP3	USB_BA+0x05C	R/W	端点 3 设置 Stall 与清除 In/Out 准备控制寄存器	0x0000_0000
USB_CFGP4	USB_BA+0x06C	R/W	端点 4 设置 Stall 与清除 In/Out 准备控制寄存器	0x0000_0000
USB_CFGP5	USB_BA+0x07C	R/W	端点 5 设置 Stall 与清除 In/Out 准备控制寄存器	0x0000_0000

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

Bits	描述	
[31:2]	Reserved	保留
[1]	SSTALL	设置STALL 1 = 设置设备自动响应 STALL 0 = 禁止设备响应 STALL
[0]	CLRRDY	清除准备 当用户设置 MXPLD 寄存器，表示该端点准备好发送或接收数据。如果用户想在传输开始前关闭传输，需要设置该位为 1来进行关闭，该位会自动清零。 IN token, 写 '1' 清除 IN token 时发送数据到 USB 的准备信号 OUT token, 写 '1' 清除 OUT token 时从 USB 接收数据的准备信号 该位只能写 1，读取返回值总是 0。

USB 驱动 SE0 寄存器 (USB_DRVSE0)

寄存器	偏移量	R/W	描述	复位后的值
USB_DRVSE0	USB_BA+0x090	R/W	USB PHY 驱动 SE0	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							DRVSE0

Bits	描述	
[31:1]	Reserved	保留
[0]	DRVSE0	USB 总线驱动 SE0 (Single Ended Zero) 在 USB_DP 和 USB_DM 都被拉低时, 为 SE0 1 = USB PHY 收发器驱动 SE0 0 = 无



5.5 通用 I/O (GPIO)

5.5.1 概述

NuMicro™ NUC130/NUC140 有 80 个通用 I/O 管脚，这些管脚可以和其他功能管脚共享，这取决于芯片的配置。80个管脚分配在GPIOA, GPIOB, GPIOC, GPIOD 与 GPIOE五个端口上，每个端口最多16个管脚。每个管脚都是独立的，都有相应的寄存器位来控制管脚功能模式与数据。

I/O 管脚上的 I/O 类型可由软件独立地配置为输入，输出，开漏或准双端模式。复位之后，所有管脚的 I/O 管脚类型均为准双端模式。端口数据寄存器 `GPIOx_DOUT[15:0]` 的值为 `0x000_FFFF`。每个 I/O 管脚有一个阻值 $110\text{ K}\Omega\sim300\text{ K}\Omega$ 的弱上拉电阻接到 V_{DD} 上， V_{DD} 的范围从 5.0 V 到 2.5 V。

5.5.2 特征

- 四种 I/O 模式：
 - ◆ 准双端模式
 - ◆ 推挽输出
 - ◆ 开漏输出
 - ◆ 高阻态输入
- 可选的 TTL/Schmitt 触发输入
- I/O 管脚可配置为 边沿/电平触发的中断源
- 支持 high driver 与 high sink 的IO 模式

5.5.3 功能描述

5.5.3.1 输入模式说明

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

5.5.3.2 输出模式说明

设置 GPIOx_PMD (PMDn[1:0]) 为 01b, GPIOx port [n] 为输出模式, 则 I/O 管脚支持数字输出功能, 有拉/灌电流能力。GPIOx_DOUT 相应位的值被驱动到相应管脚上。

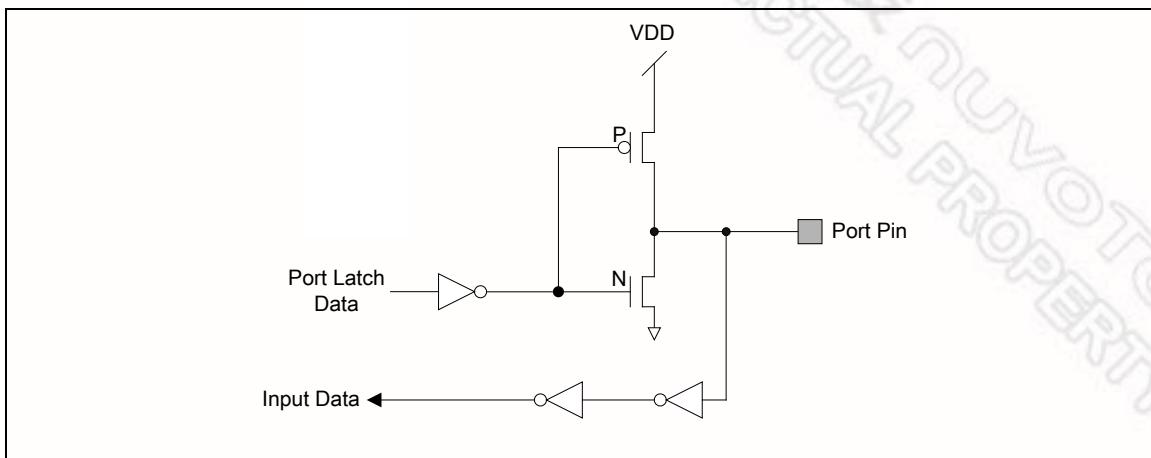


图 5-15 推挽输出

5.5.3.3 开漏模式说明

设置 GPIOx_PMD (PMDn[1:0]) 为 10b, GPIOx port [n] 为开漏模式, 则 I/O 管脚数字输出功能仅支持灌电流, 且需要外加一个上拉电阻用于驱动到高电平。如果 GPIOx_DOUT 相应位 bit[n] 为 '0', 管脚上输出低。如果 GPIOx_DOUT 相应位 bit[n] 为 '1', 该管脚则输出为高, 可以由外部上拉电阻控制。

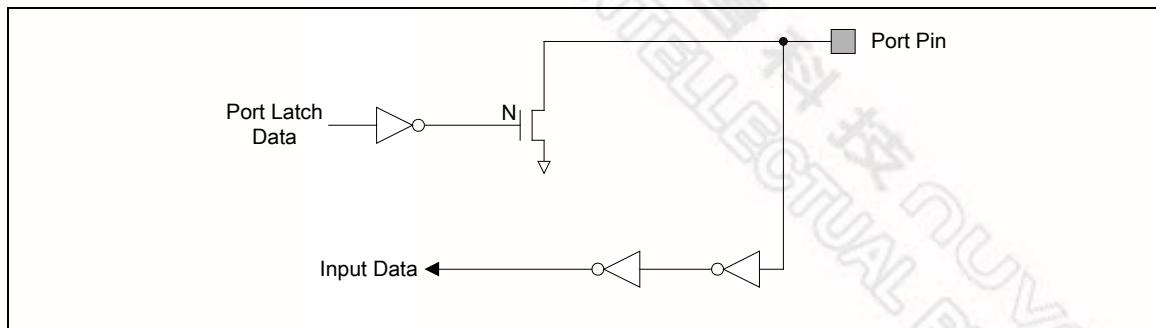


图 5-16 开漏输出

5.5.3.4 准双端模式说明

设置 GPIOx_PMD (PMDn[1:0]) 为 11b, GPIOx port [n] 为准双端模式, 则 I/O 管脚同时支持数字输出和输入功能, 但 source 电流仅达数百 μ A。要实现数字输入, 需要先将 GPIOx_DOUT 相应位置'1'。准双端输出是 80C51 及其多数派生产品所共有的模式。若 GPIOx_DOUT 相应 bit[n] 位为 '0', 则管脚上输出为 "低"。若 GPIOx_DOUT 相应 bit[n] 位为 '1', 则该管脚将检测管脚值。若管脚值为高, 没有任何动作; 若管脚值为低, 则该管脚在 2 个时钟周期内强制置高, 然后禁止强输出驱动, 管脚状态由内部上拉电阻控制。注: 准双端模式的电流大小仅有约 200 μ A 到 30 μ A (相应 VDD 电压从 5.0 V 到 2.5 V)。

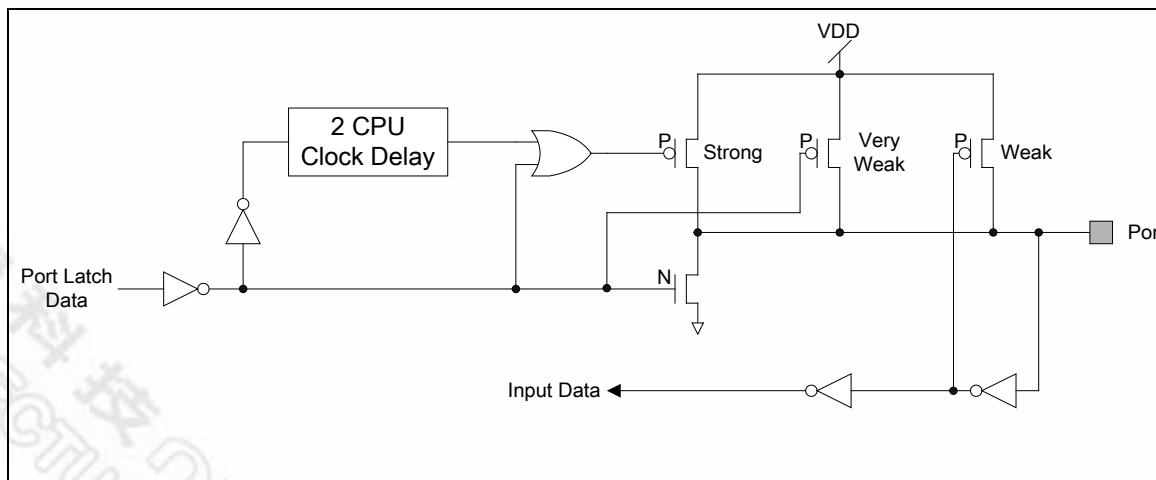


图 5-17 准双端 I/O 模式

5.5.4 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
GP_BA = 0x5000_4000				
GPIOA_PMD	GP_BA+0x000	R/W	GPIO 端口 A 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOA_OFFD	GP_BA+0x004	R/W	GPIO 端口 A 管脚 OFF 数字使能	0x0000_0000
GPIOA_DOUT	GP_BA+0x008	R/W	GPIO 端口 A 数据输出值	0x0000_FFFF
GPIOA_DMASK	GP_BA+0x00C	R/W	GPIO 端口 A 数据输出写屏蔽	0x0000_0000
GPIOA_PIN	GP_BA+0x010	R	GPIO 端口 A 管脚数值	0x0000_XXXX
GPIOA_DBEN	GP_BA+0x014	R/W	GPIO 端口 A 去抖动使能	0x0000_0000
GPIOA_IMD	GP_BA+0x018	R/W	GPIO 端口 A 中断模式控制	0x0000_0000
GPIOA_IEN	GP_BA+0x01C	R/W	GPIO 端口 A 中断使能	0x0000_0000
GPIOA_ISRC	GP_BA+0x020	R/W	GPIO 端口 A 中断源标志	0xXXXX_XXXX
GPIOB_PMD	GP_BA+0x040	R/W	GPIO 端口 B 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOB_OFFD	GP_BA+0x044	R/W	GPIO 端口 B 管脚 OFF 数字使能	0x0000_0000
GPIOB_DOUT	GP_BA+0x048	R/W	GPIO 端口 B 数据输出值	0x0000_FFFF
GPIOB_DMASK	GP_BA+0x04C	R/W	GPIO 端口 B 数据输出写屏蔽	0x0000_0000
GPIOB_PIN	GP_BA+0x050	R	GPIO 端口 B 管脚数值	0x0000_XXXX
GPIOB_DBEN	GP_BA+0x054	R/W	GPIO 端口 B 去抖动使能	0x0000_0000
GPIOB_IMD	GP_BA+0x058	R/W	GPIO 端口 B 中断模式控制	0x0000_0000
GPIOB_IEN	GP_BA+0x05C	R/W	GPIO 端口 B 中断使能	0x0000_0000
GPIOB_ISRC	GP_BA+0x060	R/W	GPIO 端口 B 中断源标志	0xXXXX_XXXX
GPIOC_PMD	GP_BA+0x080	R/W	GPIO 端口 C 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOC_OFFD	GP_BA+0x084	R/W	GPIO 端口 C 管脚 OFF 数字使能	0x0000_0000
GPIOC_DOUT	GP_BA+0x088	R/W	GPIO 端口 C 数据输出值	0x0000_FFFF
GPIOC_DMASK	GP_BA+0x08C	R/W	GPIO 端口 C 数据输出写屏蔽	0x0000_0000
GPIOC_PIN	GP_BA+0x090	R	GPIO 端口 C 管脚数值	0x0000_XXXX
GPIOC_DBEN	GP_BA+0x094	R/W	GPIO 端口 C 去抖动使能	0x0000_0000
GPIOC_IMD	GP_BA+0x098	R/W	GPIO 端口 C 中断模式控制	0x0000_0000
GPIOC_IEN	GP_BA+0x09C	R/W	GPIO 端口 C 中断使能	0x0000_0000

寄存器	偏移量	R/W	描述	复位后的值
GPIOC_ISRC	GP_BA+0x0A0	R/W	GPIO 端口 C 中断源标志	0xXXXX_XXXX
GPIOD_PMD	GP_BA+0x0C0	R/W	GPIO 端口 D 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOD_OFFD	GP_BA+0x0C4	R/W	GPIO 端口 D 管脚 OFF 数字使能	0x0000_0000
GPIOD_DOUT	GP_BA+0x0C8	R/W	GPIO 端口 D 数据输出值	0x0000_FFFF
GPIOD_DMASK	GP_BA+0x0CC	R/W	GPIO 端口 D 数据输出写屏蔽	0x0000_0000
GPIOD_PIN	GP_BA+0x0D0	R	GPIO 端口 D 管脚数值	0x0000_XXXX
GPIOD_DBEN	GP_BA+0x0D4	R/W	GPIO 端口 D 去抖动使能	0x0000_0000
GPIOD_IMD	GP_BA+0x0D8	R/W	GPIO 端口 D 中断模式控制	0x0000_0000
GPIOD_IEN	GP_BA+0x0DC	R/W	GPIO 端口 D 中断使能	0x0000_0000
GPIOD_ISRC	GP_BA+0x0E0	R/W	GPIO 端口 D 中断源标志	0xXXXX_XXXX
GPIOE_PMD	GP_BA+0x100	R/W	GPIO 端口 E 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOE_OFFD	GP_BA+0x104	R/W	GPIO 端口 E 管脚 OFF 数字使能	0x0000_0000
GPIOE_DOUT	GP_BA+0x108	R/W	GPIO 端口 E 数据输出值	0x0000_FFFF
GPIOE_DMASK	GP_BA+0x10C	R/W	GPIO 端口 E 数据输出写屏蔽	0x0000_0000
GPIOE_PIN	GP_BA+0x110	R	GPIO 端口 E 管脚数值	0x0000_XXXX
GPIOE_DBEN	GP_BA+0x114	R/W	GPIO 端口 E 去抖动使能	0x0000_0000
GPIOE_IMD	GP_BA+0x118	R/W	GPIO 端口 E 中断模式控制	0x0000_0000
GPIOE_IEN	GP_BA+0x11C	R/W	GPIO 端口 E 中断使能	0x0000_0000
GPIOE_ISRC	GP_BA+0x120	R/W	GPIO 端口 E 中断源标志	0xXXXX_XXXX
DBNCECON	GP_BA+0x180	R/W	去抖动循环控制	0x0000_0020
GPIOA0_DOUT	GP_BA+0x200	R/W	GPIO PA.0 位输出/输入控制	0x0000_0001
GPIOA1_DOUT	GP_BA+0x204	R/W	GPIO PA.1 位输出/输入控制	0x0000_0001
GPIOA2_DOUT	GP_BA+0x208	R/W	GPIO PA.2 位输出/输入控制	0x0000_0001
GPIOA3_DOUT	GP_BA+0x20C	R/W	GPIO PA.3 位输出/输入控制	0x0000_0001
GPIOA4_DOUT	GP_BA+0x210	R/W	GPIO PA.4 位输出/输入控制	0x0000_0001
GPIOA5_DOUT	GP_BA+0x214	R/W	GPIO PA.5 位输出/输入控制	0x0000_0001
GPIOA6_DOUT	GP_BA+0x218	R/W	GPIO PA.6 位输出/输入控制	0x0000_0001
GPIOA7_DOUT	GP_BA+0x21C	R/W	GPIO PA.7 位输出/输入控制	0x0000_0001
GPIOA8_DOUT	GP_BA+0x220	R/W	GPIO PA.8 位输出/输入控制	0x0000_0001

寄存器	偏移量	R/W	描述	复位后的值
GPIOA9_DOUT	GP_BA+0x224	R/W	GPIO PA.9 位输出/输入控制	0x0000_0001
GPIOA10_DOUT	GP_BA+0x228	R/W	GPIO PA.10 位输出/输入控制	0x0000_0001
GPIOA11_DOUT	GP_BA+0x22C	R/W	GPIO PA.11 位输出/输入控制	0x0000_0001
GPIOA12_DOUT	GP_BA+0x230	R/W	GPIO PA.12 位输出/输入控制	0x0000_0001
GPIOA13_DOUT	GP_BA+0x234	R/W	GPIO PA.13 位输出/输入控制	0x0000_0001
GPIOA14_DOUT	GP_BA+0x238	R/W	GPIO PA.14 位输出/输入控制	0x0000_0001
GPIOA15_DOUT	GP_BA+0x23C	R/W	GPIO PA.15 位输出/输入控制	0x0000_0001
GPIOB0_DOUT	GP_BA+0x240	R/W	GPIO PB.0 位输出/输入控制	0x0000_0001
GPIOB1_DOUT	GP_BA+0x244	R/W	GPIO PB.1 位输出/输入控制	0x0000_0001
GPIOB2_DOUT	GP_BA+0x248	R/W	GPIO PB.2 位输出/输入控制	0x0000_0001
GPIOB3_DOUT	GP_BA+0x24C	R/W	GPIO PB.3 位输出/输入控制	0x0000_0001
GPIOB4_DOUT	GP_BA+0x250	R/W	GPIO PB.4 位输出/输入控制	0x0000_0001
GPIOB5_DOUT	GP_BA+0x254	R/W	GPIO PB.5 位输出/输入控制	0x0000_0001
GPIOB6_DOUT	GP_BA+0x258	R/W	GPIO PB.6 位输出/输入控制	0x0000_0001
GPIOB7_DOUT	GP_BA+0x25C	R/W	GPIO PB.7 位输出/输入控制	0x0000_0001
GPIOB8_DOUT	GP_BA+0x260	R/W	GPIO PB.8 位输出/输入控制	0x0000_0001
GPIOB9_DOUT	GP_BA+0x264	R/W	GPIO PB.9 位输出/输入控制	0x0000_0001
GPIOB10_DOUT	GP_BA+0x268	R/W	GPIO PB.10 位输出/输入控制	0x0000_0001
GPIOB11_DOUT	GP_BA+0x26C	R/W	GPIO PB.11 位输出/输入控制	0x0000_0001
GPIOB12_DOUT	GP_BA+0x270	R/W	GPIO PB.12 位输出/输入控制	0x0000_0001
GPIOB13_DOUT	GP_BA+0x274	R/W	GPIO PB.13 位输出/输入控制	0x0000_0001
GPIOB14_DOUT	GP_BA+0x278	R/W	GPIO PB.14 位输出/输入控制	0x0000_0001
GPIOB15_DOUT	GP_BA+0x27C	R/W	GPIO PB.15 位输出/输入控制	0x0000_0001
GPIOC0_DOUT	GP_BA+0x280	R/W	GPIO PC.0 位输出/输入控制	0x0000_0001
GPIOC1_DOUT	GP_BA+0x284	R/W	GPIO PC.1 位输出/输入控制	0x0000_0001
GPIOC2_DOUT	GP_BA+0x288	R/W	GPIO PC.2 位输出/输入控制	0x0000_0001
GPIOC3_DOUT	GP_BA+0x28C	R/W	GPIO PC.3 位输出/输入控制	0x0000_0001
GPIOC4_DOUT	GP_BA+0x290	R/W	GPIO PC.4 位输出/输入控制	0x0000_0001
GPIOC5_DOUT	GP_BA+0x294	R/W	GPIO PC.5 位输出/输入控制	0x0000_0001

寄存器	偏移量	R/W	描述	复位后的值
GPIOC6_DOUT	GP_BA+0x298	R/W	GPIO PC.6 位输出/输入控制	0x0000_0001
GPIOC7_DOUT	GP_BA+0x29C	R/W	GPIO PC.7 位输出/输入控制	0x0000_0001
GPIOC8_DOUT	GP_BA+0x2A0	R/W	GPIO PC.8 位输出/输入控制	0x0000_0001
GPIOC9_DOUT	GP_BA+0x2A4	R/W	GPIO PC.9 位输出/输入控制	0x0000_0001
GPIOC10_DOUT	GP_BA+0x2A8	R/W	GPIO PC.10 位输出/输入控制	0x0000_0001
GPIOC11_DOUT	GP_BA+0x2AC	R/W	GPIO PC.11 位输出/输入控制	0x0000_0001
GPIOC12_DOUT	GP_BA+0x2B0	R/W	GPIO PC.12 位输出/输入控制	0x0000_0001
GPIOC13_DOUT	GP_BA+0x2B4	R/W	GPIO PC.13 位输出/输入控制	0x0000_0001
GPIOC14_DOUT	GP_BA+0x2B8	R/W	GPIO PC.14 位输出/输入控制	0x0000_0001
GPIOC15_DOUT	GP_BA+0x2BC	R/W	GPIO PC.15 位输出/输入控制	0x0000_0001
GPIOD0_DOUT	GP_BA+0x2C0	R/W	GPIO PD.0 位输出/输入控制	0x0000_0001
GPIOD1_DOUT	GP_BA+0x2C4	R/W	GPIO PD.1 位输出/输入控制	0x0000_0001
GPIOD2_DOUT	GP_BA+0x2C8	R/W	GPIO PD.2 位输出/输入控制	0x0000_0001
GPIOD3_DOUT	GP_BA+0x2CC	R/W	GPIO PD.3 位输出/输入控制	0x0000_0001
GPIOD4_DOUT	GP_BA+0x2D0	R/W	GPIO PD.4 位输出/输入控制	0x0000_0001
GPIOD5_DOUT	GP_BA+0x2D4	R/W	GPIO PD.5 位输出/输入控制	0x0000_0001
GPIOD6_DOUT	GP_BA+0x2D8	R/W	GPIO PD.6 位输出/输入控制	0x0000_0001
GPIOD7_DOUT	GP_BA+0x2DC	R/W	GPIO PD.7 位输出/输入控制	0x0000_0001
GPIOD8_DOUT	GP_BA+0x2E0	R/W	GPIO PD.8 位输出/输入控制	0x0000_0001
GPIOD9_DOUT	GP_BA+0x2E4	R/W	GPIO PD.9 位输出/输入控制	0x0000_0001
GPIOD10_DOUT	GP_BA+0x2E8	R/W	GPIO PD.10 位输出/输入控制	0x0000_0001
GPIOD11_DOUT	GP_BA+0x2EC	R/W	GPIO PD.11 位输出/输入控制	0x0000_0001
GPIOD12_DOUT	GP_BA+0x2F0	R/W	GPIO PD.12 位输出/输入控制	0x0000_0001
GPIOD13_DOUT	GP_BA+0x2F4	R/W	GPIO PD.13 位输出/输入控制	0x0000_0001
GPIOD14_DOUT	GP_BA+0x2F8	R/W	GPIO PD.14 位输出/输入控制	0x0000_0001
GPIOD15_DOUT	GP_BA+0x2FC	R/W	GPIO PD.15 位输出/输入控制	0x0000_0001
GPIOE0_DOUT	GP_BA+0x300	R/W	GPIO PE.0 位输出/输入控制	0x0000_0001
GPIOE1_DOUT	GP_BA+0x304	R/W	GPIO PE.1 位输出/输入控制	0x0000_0001
GPIOE2_DOUT	GP_BA+0x308	R/W	GPIO PE.2 位输出/输入控制	0x0000_0001

寄存器	偏移量	R/W	描述	复位后的值
GPIOE3_DOUT	GP_BA+0x30C	R/W	GPIO PE.3 位输出/输入控制	0x0000_0001
GPIOE4_DOUT	GP_BA+0x310	R/W	GPIO PE.4 位输出/输入控制	0x0000_0001
GPIOE5_DOUT	GP_BA+0x314	R/W	GPIO PE.5 位输出/输入控制	0x0000_0001
GPIOE6_DOUT	GP_BA+0x318	R/W	GPIO PE.6 位输出/输入控制	0x0000_0001
GPIOE7_DOUT	GP_BA+0x31C	R/W	GPIO PE.7 位输出/输入控制	0x0000_0001
GPIOE8_DOUT	GP_BA+0x320	R/W	GPIO PE.8 位输出/输入控制	0x0000_0001
GPIOE9_DOUT	GP_BA+0x324	R/W	GPIO PE.9 位输出/输入控制	0x0000_0001
GPIOE10_DOUT	GP_BA+0x328	R/W	GPIO PE.10 位输出/输入控制	0x0000_0001
GPIOE11_DOUT	GP_BA+0x32C	R/W	GPIO PE.11 位输出/输入控制	0x0000_0001
GPIOE12_DOUT	GP_BA+0x330	R/W	GPIO PE.12 位输出/输入控制	0x0000_0001
GPIOE13_DOUT	GP_BA+0x334	R/W	GPIO PE.13 位输出/输入控制	0x0000_0001
GPIOE14_DOUT	GP_BA+0x338	R/W	GPIO PE.14 位输出/输入控制	0x0000_0001
GPIOE15_DOUT	GP_BA+0x33C	R/W	GPIO PE.15 位输出/输入控制	0x0000_0001



5.5.5 寄存器描述

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

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_PMD	GP_BA+0x000	R/W	GPIO 端口 A 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOB_PMD	GP_BA+0x040	R/W	GPIO 端口 B 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOC_PMD	GP_BA+0x080	R/W	GPIO 端口 C 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOD_PMD	GP_BA+0x0C0	R/W	GPIO 端口 D 管脚 I/O 模式控制	0xFFFF_FFFF
GPIOE_PMD	GP_BA+0x100	R/W	GPIO 端口 E 管脚 I/O 模式控制	0xFFFF_FFFF

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

Bits	描述	
[2n+1:2n]	PMDn	GPIOx I/O Pin[n] 模式控制 决定 GPIOx 管脚的 I/O 类型 00 = GPIO port [n] 管脚为输入模式 01 = GPIO port [n] 管脚为输出模式 10 = GPIO port [n] 管脚为开漏模式 11 = GPIO port [n] 管脚为准双向模式

GPIO 端口 [A/B/C/D/E] 管脚 OFF 数字使能寄存器 (GPIOx_OFFD)

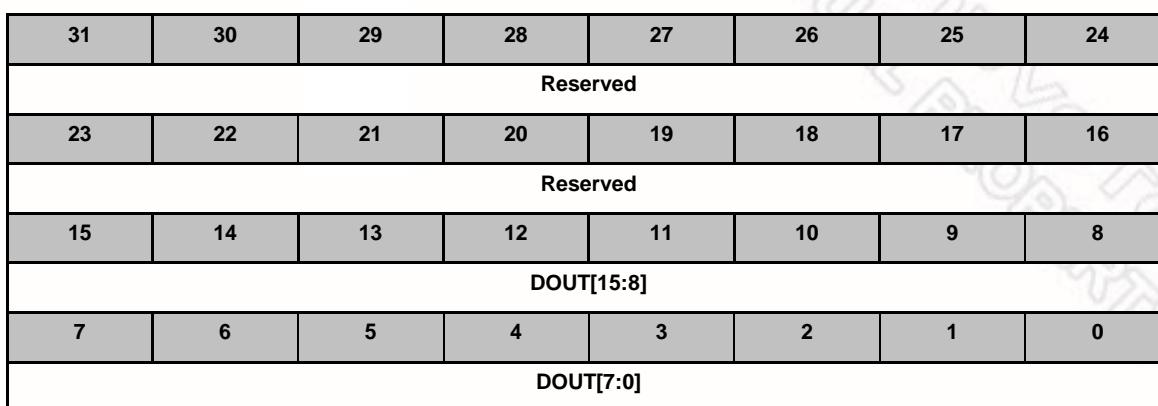
寄存器	偏移量	R/W	描述	复位后的值
GPIOA_OFFD	GP_BA+0x004	R/W	GPIO 端口 A 管脚 OFF 数字使能	0x0000_0000
GPIOB_OFFD	GP_BA+0x044	R/W	GPIO 端口 B 管脚 OFF 数字使能	0x0000_0000
GPIOC_OFFD	GP_BA+0x084	R/W	GPIO 端口 C 管脚 OFF 数字使能	0x0000_0000
GPIOD_OFFD	GP_BA+0x0C4	R/W	GPIO 端口 D 管脚 OFF 数字使能	0x0000_0000
GPIOE_OFFD	GP_BA+0x104	R/W	GPIO 端口 E 管脚 OFF 数字使能	0x0000_0000

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

Bits	描述	
[16:31]	OFFD	GPIOx Pin[n] OFF 数字输入通道使能 用于控制 bit 对应的 GPIO 的数字输入通道是否禁用。若输入为模拟信号，用户可以关闭数字输入通道避免漏电。 1 = 禁用 IO 数字输入通道（数字输入拉低） 0 = 使能 IO 数字输入通道
[0:15]	Reserved	保留

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

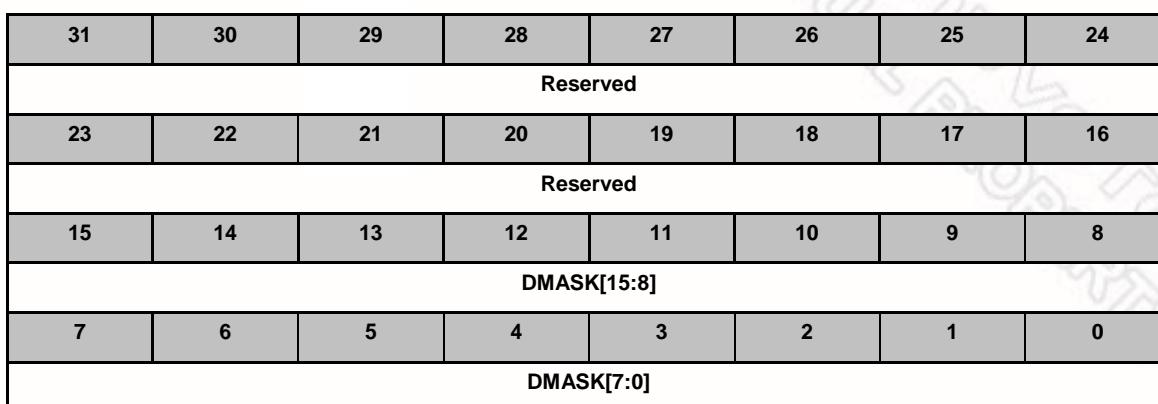
寄存器	偏移量	R/W	描述	复位后的值
GPIOA_DOUT	GP_BA+0x008	R/W	GPIO 端口 A 数据输出值	0x0000_FFFF
GPIOB_DOUT	GP_BA+0x048	R/W	GPIO 端口 B 数据输出值	0x0000_FFFF
GPIOC_DOUT	GP_BA+0x088	R/W	GPIO 端口 C 数据输出值	0x0000_FFFF
GPIOD_DOUT	GP_BA+0x0C8	R/W	GPIO 端口 D 数据输出值	0x0000_FFFF
GPIOE_DOUT	GP_BA+0x108	R/W	GPIO 端口 E 数据输出值	0x0000_FFFF



Bits	描述	
[31:16]	Reserved	保留
[n]	DOUT[n]	<p>GPIOx Pin[n] 数据输出值</p> <p>在 GPIO 管脚被配置成输出, 开漏和准双向模式时, 控制 GPIO 管脚的状态。</p> <p>1 = GPIO 管脚被配置成输出, 开漏和准双向模式时, GPIO 端口 [A/B/C/D/E] Pin[n] 为高。</p> <p>0 = GPIO 管脚被配置成输出, 开漏和准双向模式时, GPIO 端口 [A/B/C/D/E] Pin[n] 为低。</p>

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

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_DMASK	GP_BA+0x00C	R/W	GPIO 端口 A 数据输出写屏蔽	0xFFFF_0000
GPIOB_DMASK	GP_BA+0x04C	R/W	GPIO 端口 B 数据输出写屏蔽	0xFFFF_0000
GPIOC_DMASK	GP_BA+0x08C	R/W	GPIO 端口 C 数据输出写屏蔽	0xFFFF_0000
GPIOD_DMASK	GP_BA+0x0CC	R/W	GPIO 端口 D 数据输出写屏蔽	0xFFFF_0000
GPIOE_DMASK	GP_BA+0x10C	R/W	GPIO 端口 E 数据输出写屏蔽	0xFFFF_0000



Bits	描述	
[31:16]	Reserved	保留
[n]	DMASK[n]	<p>端口 [A/B/C/D/E] 数据输出写屏蔽 用于保护相应寄存器 GPIOx_DOUT bit[n]。在设置 DMASK bit[n] 为'1'，相应的 GPIOx_DOUT[n] bit 被保护。写信号被屏蔽时，不能向保护位写数据。</p> <p>1 = 相应的 GPIOx_DOUT[n] bit 被保护 0 = 相应的 GPIOx_DOUT[n] bit 能被更新</p> <p>注：该功能只保护相应的 GPIOx_DOUT[n] bit，而不会保护控制寄存器 (GPIOAx_DOUT, GPIOBx_DOUT, GPIOCx_DOUT, GPIODx_DOUT, GPIOEx_DOUT) 的相应的位。</p>

GPIO 端口 [A/B/C/D/E] 管脚数值 (GPIOx_PIN)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_PIN	GP_BA+0x010	R	GPIO 端口 A 管脚数值	0x0000_XXXX
GPIOB_PIN	GP_BA+0x050	R	GPIO 端口 B 管脚数值	0x0000_XXXX
GPIOC_PIN	GP_BA+0x090	R	GPIO 端口 C 管脚数值	0x0000_XXXX
GPIOD_PIN	GP_BA+0x0D0	R	GPIO 端口 D 管脚数值	0x0000_XXXX
GPIOE_PIN	GP_BA+0x110	R	GPIO 端口 E 管脚数值	0x0000_XXXX

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

Bits	描述	
[31:16]	Reserved	保留
[n]	PIN[n]	<p>端口 [A/B/C/D/E] 管脚数值</p> <p>这些位的值反映了各个 GPIO 管脚的真实状态。</p> <p>如果某位为 1，则相应的管脚状态为高，反之则为低。</p>

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

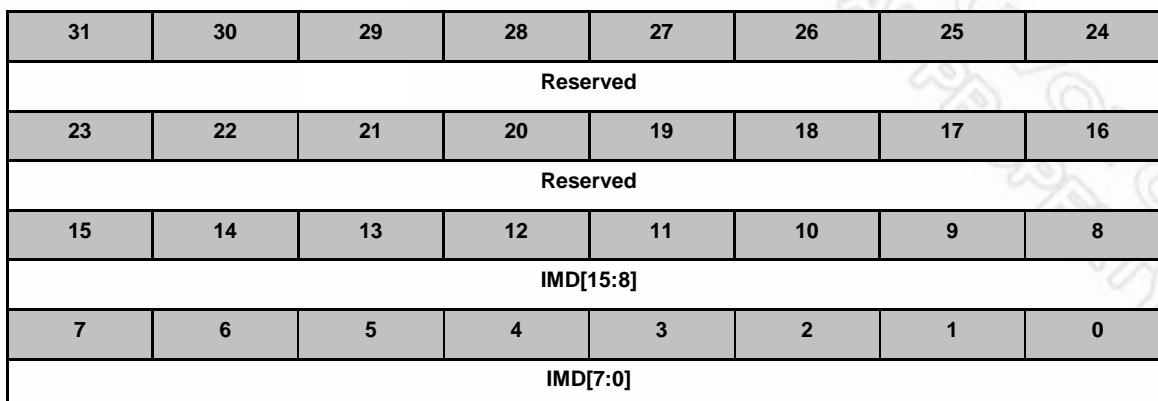
寄存器	偏移量	R/W	描述	复位后的值
GPIOA_DBEN	GP_BA+0x014	R/W	GPIO 端口 A 去抖动使能	0xXXXX_0000
GPIOB_DBEN	GP_BA+0x054	R/W	GPIO 端口 B 去抖动使能	0xXXXX_0000
GPIOC_DBEN	GP_BA+0x094	R/W	GPIO 端口 C 去抖动使能	0xXXXX_0000
GPIOD_DBEN	GP_BA+0x0D4	R/W	GPIO 端口 D 去抖动使能	0xXXXX_0000
GPIOE_DBEN	GP_BA+0x114	R/W	GPIO 端口 E 去抖动使能	0xXXXX_0000

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

Bits	描述	
[31:16]	Reserved	保留
[n]	DBEN[n]	<p>端口 [A/B/C/D/E] 输入信号去抖动使能</p> <p>DBEN[n] 用于使能相应位的去抖动功能。如果输入信号脉冲宽度不能被两个连续的去抖动采样周期所采样，则输入信号被视为信号抖动，从而不会触发中断。去抖动的时钟源由 DBNCECON[4] 控制，去抖动的采样周期由 DBNCECON[3:0] 控制。</p> <p>DBEN[n] 仅用于边沿触发 “edge-trigger” 中断，不能用于电平触发 (“level trigger”) 中断。</p> <p>1 = 使能 bit[n] 去抖动功能 0 = 禁用 bit[n] 去抖动功能</p> <p>去抖动功能对边沿触发中断有效，对于电平触发中断模式该使能位不起作用。</p>

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

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_IMD	GP_BA+0x018	R/W	GPIO 端口 A 中断模式控制	0xFFFF_0000
GPIOB_IMD	GP_BA+0x058	R/W	GPIO 端口 B 中断模式控制	0xFFFF_0000
GPIOC_IMD	GP_BA+0x098	R/W	GPIO 端口 C 中断模式控制	0xFFFF_0000
GPIOD_IMD	GP_BA+0x0D8	R/W	GPIO 端口 D 中断模式控制	0xFFFF_0000
GPIOE_IMD	GP_BA+0x118	R/W	GPIO 端口 E 中断模式控制	0xFFFF_0000



Bits	描述	
[31:16]	Reserved	保留
[n]	IMD[n]	<p>端口 [A/B/C/D/E] 边沿或电平检测中断控制</p> <p>IMD[n] 用于控制中断是电平触发或边沿触发。若为边沿触发中断，触发源可由去抖动控制；若为电平触发中断，输入源由一个 HCLK 时钟采样并产生中断。</p> <p>1 = 电平触发中断 0 = 边沿触发中断</p> <p>如果设置管脚为电平触发中断，则在寄存器 GPIOx_IEN 中，只能设置一个电平。若设置了两个电平去触发中断，则设置被忽略，不会产生中断。</p> <p>去抖动功能对于边沿触发有效，对于电平触发中断无效。</p>

GPIO 端口 [A/B/C/D] 中断使能控制 (GPIOx_IEN)

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_IEN	GP_BA+0x01C	R/W	GPIO 端口 A 中断使能	0x0000_0000
GPIOB_IEN	GP_BA+0x05C	R/W	GPIO端口 B 中断使能	0x0000_0000
GPIOC_IEN	GP_BA+0x09C	R/W	GPIO端口 C 中断使能	0x0000_0000
GPIOD_IEN	GP_BA+0x0DC	R/W	GPIO端口 D 中断使能	0x0000_0000
GPIOE_IEN	GP_BA+0x11C	R/W	GPIO端口 E 中断使能	0x0000_0000

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

Bits	描述
[n+16]	<p>IR_EN[n] 端口 [A/B/C/D/E] 输入上升沿或输入高电平中断使能 IR_EN[n] 用来使能相应 GPIO_PIN[n] 输入的中断。该位置 '1' 也能使能管脚的唤醒功能。 当设置 IR_EN[n] 位为 1: 如果中断是电平触发模式，输入 PIN[n] 的状态为高电平时将产生中断。 如果中断是边沿触发模式，输入 PIN[n] 的状态由低电平到高电平变化时将产生中断 1 = 使能 PIN[n] 高电平或由低电平到高电平变化的中断 0 = 禁用 PIN[n] 高电平或由低电平到高电平变化的中断</p>
[n]	<p>IF_EN[n] 端口 [A/B/C/D/E] 输入下降沿或输入低电平中断使能 IF_EN[n] 用于使能相应 GPIO_PIN[n] 输入的中断。该位置 '1' 也能使能管脚的唤醒功能。 当设置 IF_EN[n] 位为 1: 如果中断是电平触发模式，输入 PIN[n] 的状态为低电平时将产生中断。 如果中断是边沿触发模式，输入 PIN[n] 的状态由高电平到低电平变化时将产生中断。 1 = 使能 PIN[n] 低电平或由高电平到低电平变化的中断 0 = 禁用 PIN[n] 低电平或由高电平到低电平变化的中断</p>

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

寄存器	偏移量	R/W	描述	复位后的值
GPIOA_ISRC	GP_BA+0x020	R/W	GPIO 端口 A 中断触发源标志	0x0000_0000
GPIOB_ISRC	GP_BA+0x060	R/W	GPIO 端口 B 中断触发源标志	0x0000_0000
GPIOC_ISRC	GP_BA+0x0A0	R/W	GPIO 端口 C 中断触发源标志	0x0000_0000
GPIOD_ISRC	GP_BA+0x0E0	R/W	GPIO 端口 D 中断触发源标志	0x0000_0000
GPIOE_ISRC	GP_BA+0x120	R/W	GPIO 端口 E 中断触发源标志	0x0000_0000

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

Bits	描述	
[31:16]	Reserved	保留
[n]	ISRC[n]	<p>端口 [A/B/C/D/E] 中断触发源标志</p> <p>读:</p> <p>1 = GPIOx[n] 产生中断 0 = GPIOx[n] 没有中断</p> <p>写:</p> <p>1= 清除相应的未处理中断 0= 无动作</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	外部中断去抖动控制	0x0000_0020

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

Bits	描述																							
[5]	ICLK_ON	<p>中断时钟 On 模式 如果 GPIO pin[n] 的中断被禁用, 那么设置该位为 0 将禁用中断产生电路时钟。 1 = 中断产生电路时钟总是使能 0 = 若 GPIOA/B/C/D/E[n] 中断被禁用, 则关闭相应的时钟</p>																						
[4]	DBCLKSRC	<p>去抖动计数器时钟源选择 1 = 去抖动计数器时钟源为内部 10 kHz 振荡器 0 = 去抖动计数器时钟源为 HCLK</p>																						
[3:0]	DBCLKSEL	<p>去抖动采样周期选择</p> <table border="1"> <thead> <tr> <th>DBCLKSEL</th> <th>描述</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>采样中断输入每 1 clocks 1 次</td> </tr> <tr> <td>1</td> <td>采样中断输入每 2 clocks 1 次</td> </tr> <tr> <td>2</td> <td>采样中断输入每 4 clocks 1 次</td> </tr> <tr> <td>3</td> <td>采样中断输入每 8 clocks 1 次</td> </tr> <tr> <td>4</td> <td>采样中断输入每 16 clocks 1 次</td> </tr> <tr> <td>5</td> <td>采样中断输入每 32 clocks 1 次</td> </tr> <tr> <td>6</td> <td>采样中断输入每 64 clocks 1 次</td> </tr> <tr> <td>7</td> <td>采样中断输入每 128 clocks 1 次</td> </tr> <tr> <td>8</td> <td>采样中断输入每 256 clocks 1 次</td> </tr> <tr> <td>9</td> <td>采样中断输入每 2*256 clocks 1 次</td> </tr> </tbody> </table>	DBCLKSEL	描述	0	采样中断输入每 1 clocks 1 次	1	采样中断输入每 2 clocks 1 次	2	采样中断输入每 4 clocks 1 次	3	采样中断输入每 8 clocks 1 次	4	采样中断输入每 16 clocks 1 次	5	采样中断输入每 32 clocks 1 次	6	采样中断输入每 64 clocks 1 次	7	采样中断输入每 128 clocks 1 次	8	采样中断输入每 256 clocks 1 次	9	采样中断输入每 2*256 clocks 1 次
DBCLKSEL	描述																							
0	采样中断输入每 1 clocks 1 次																							
1	采样中断输入每 2 clocks 1 次																							
2	采样中断输入每 4 clocks 1 次																							
3	采样中断输入每 8 clocks 1 次																							
4	采样中断输入每 16 clocks 1 次																							
5	采样中断输入每 32 clocks 1 次																							
6	采样中断输入每 64 clocks 1 次																							
7	采样中断输入每 128 clocks 1 次																							
8	采样中断输入每 256 clocks 1 次																							
9	采样中断输入每 2*256 clocks 1 次																							

	10	采样中断输入每 4*256clocks 1 次	
	11	采样中断输入每 8*256 clocks 1 次	
	12	采样中断输入每 16*256 clocks 1 次	
	13	采样中断输入每 32*256 clocks 1 次	
	14	采样中断输入每 64*256 clocks 1 次	
	15	采样中断输入每 128*256 clocks 1 次	

GPIO 端口 [A/B/C/D/E] I/O 位输出/输入控制 (GPIOxx_DOUT)

寄存器	偏移量	R/W	描述	复位后的值
GPIOAx_DOUT	GP_BA+0x200	R/W	GPIO Port A Pin I/O 位输出/输入控制	0x0000_0001
	- GP_BA+0x23C			
GPIOBx_DOUT	GP_BA+0x240	R/W	GPIO Port B Pin I/O 位输出/输入控制	0x0000_0001
	- GP_BA+0x27C			
GPIOCx_DOUT	GP_BA+0x280	R/W	GPIO Port C Pin I/O 位输出/输入控制	0x0000_0001
	- GP_BA+0x2BC			
GPIO Dx_DOUT	GP_BA+0x2C0	R/W	GPIO Port D Pin I/O 位输出/输入控制	0x0000_0001
	- GP_BA+0x2FC			
GPIOEx_DOUT	GP_BA+0x300	R/W	GPIO Port E Pin I/O 位输出/输入控制	0x0000_0001
	- GP_BA+0x3FC			

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

Bits	描述
[0]	GPIOxx_DOUT GPIOxx I/O 管脚位输出/输入控制 写该位可以控制 GPIO 管脚的输出值 1 = 设置相应 GPIO 管脚为高 0 = 设置相应 GPIO 管脚为低 读该寄存器可以获得 IO 管脚的状态。 例如: 写 GPIOA0_DOUT 即将写入值写到 GPIOA_DOUT[0] 位, 读 GPIOA0_DOUT 即读取 GPIOA_PIN[0] 的值。

5.6 I²C 串行接口控制器 (Master/Slave) (I²C)

5.6.1 概述

I²C 为双线、双向串行总线，通过简单有效的连线方式实现设备间的数据交换。I²C 的标准是一个多主机总线，包括冲突检测和仲裁以防止两个或多个主机试图同时控制总线时发生的数据损坏。

数据在主机与从机之间通过 SCL 时钟线控制，在 SDA 数据线上按一字节一字节的同步传输。每个字节为 8 位长度，一个 SCL 时钟脉冲传输一个数据位，数据由最高位 MSB 开始传输，每个传输字节后跟随一个应答位，每个位在 SCL 为高时采样；因此，SDA 线只有在 SCL 为低时才可以改变，在 SCL 为高时 SDA 保持稳定。当 SCL 为高时，SDA 线上的跳变视为命令中断 (START or STOP)。更多关于 I²C 总线时序的细节请参考 图 5-18。

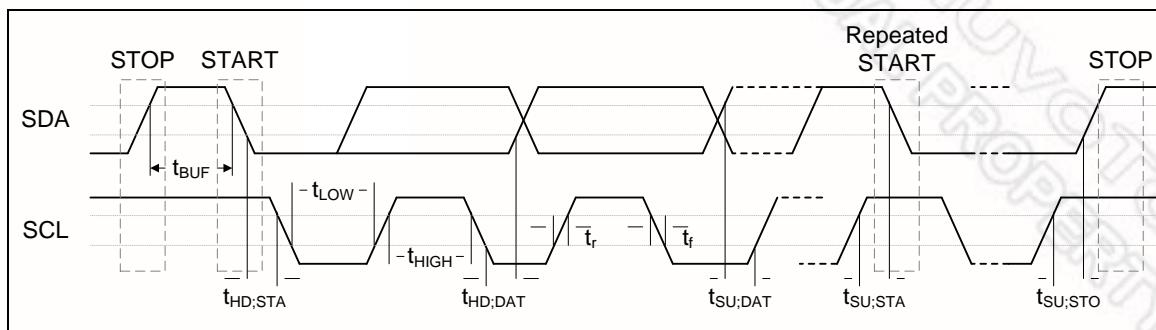


图 5-18 I²C 总线时序

设备的片上 I²C 逻辑提供符合 I²C 总线规范的串行接口。I²C 端口自动处理字节传输，将 I2CON 的 ENS1 设置为 '1'，即可使能该端口。I²C H/W 接口通过 SDA 与 SCL 两个引脚连到 I²C 总线。用于 I²C 操作的两个管脚需要上拉电阻，因为这两个管脚为开漏脚。当 I/O 管脚作为 I²C 端口使用时，用户必须事先设定 I/O 管脚功能为 I²C 功能。



5.6.2 特征

I²C 总线通过两条线 (SDA 和 SCL) 在连接到总线上的设备之间传输信息。总线的主要特征：

- 支持主机/从机 模式
- 主从机之间双向数据传输
- 多主机总线支持 (无中心主机)
- 多主机间同时传输数据仲裁，避免总线上串行数据损坏
- 总线采用串行同步时钟，可实现设备之间以不同的速率传输
- 串行同步时钟可作为握手方式控制总线上数据暂停及恢复传送
- 内建14位溢出计数器，当 I²C 总线中止且定时计数器溢出，产生 I²C 中断
- 需要外部上拉确保高电平输出
- 可编辑的时钟适用于不同速率控制
- 支持 7 位地址模式
- I²C 总线控制器支持多地址识别 (4组从机地址带 mask 选项)

5.6.3 功能描述

5.6.3.1 I²C 协议

通常标准的 I²C 通信包含四个部分：

- 1) 起始信号 (START) 或者重复起始信号 (Repeated START)
- 2) 从机地址传输和 R/W 位传输
- 3) 数据传输
- 4) 停止信号 (STOP)

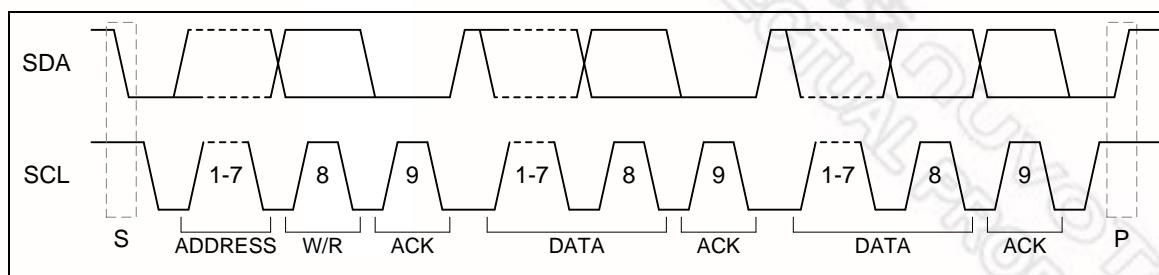


图 5-19 I²C 协议

5.6.3.2 I²C 总线上的数据传输

主机发出7位地址给从机接收者进行寻址

传输方向未改变

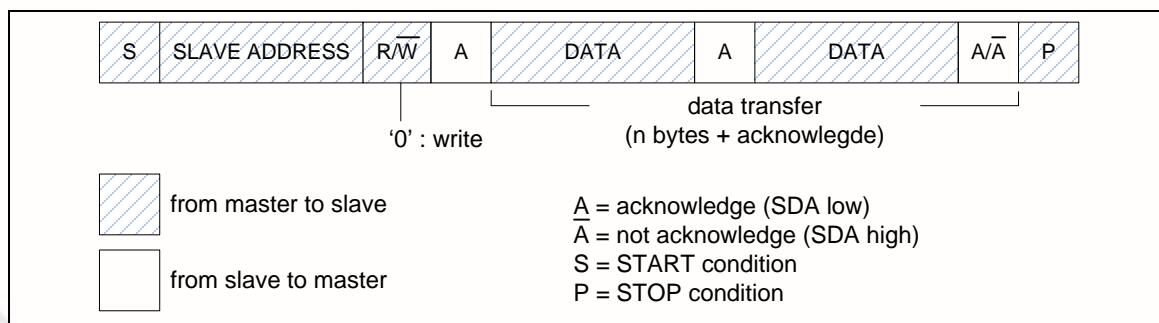


图 5-20 主机向从机传输数据

第一个字节（从机地址）后，主机紧接着从从机读取数据

传输方向改变

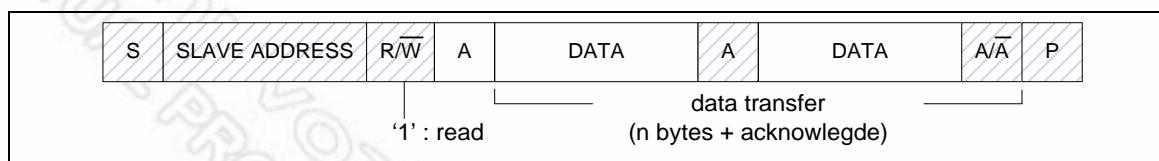


图 5-21 主机从从机读取数据

5.6.3.3 起始或重复起始信号

当总线处于空闲状态下，说明没有主机设备占用总线（SCL 和 SDA 线同时为高），主机可以通过发送一个起始 (START) 信号来发起传输请求。起始信号，通常表示为 S-bit，当 SCL 线为高时，SDA 线上信号由高至低变化，就被定义为起始信号。起始信号表示一个新的数据传输的开始。

重复起始信号 (Sr) 即在两个起始 (START) 信号之间没有停止 (STOP) 信号。主机采用这种方法与另一个从机或同一个从机以不同传输方向进行通信（例如：从写设备到读设备）而不释放总线。

停止 (STOP) 信号

主机可以通过产生一个停止信号来终止数据传送。停止信号，通常表示为 P-bit，当 SCL 线为高时，SDA 线上信号由低至高变化，就被定义为停止信号。

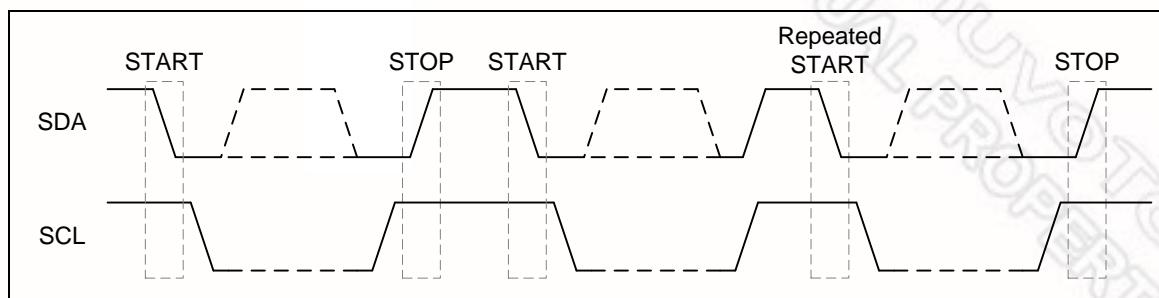


图 5-22 起始 (START) 和停止 (STOP) 条件

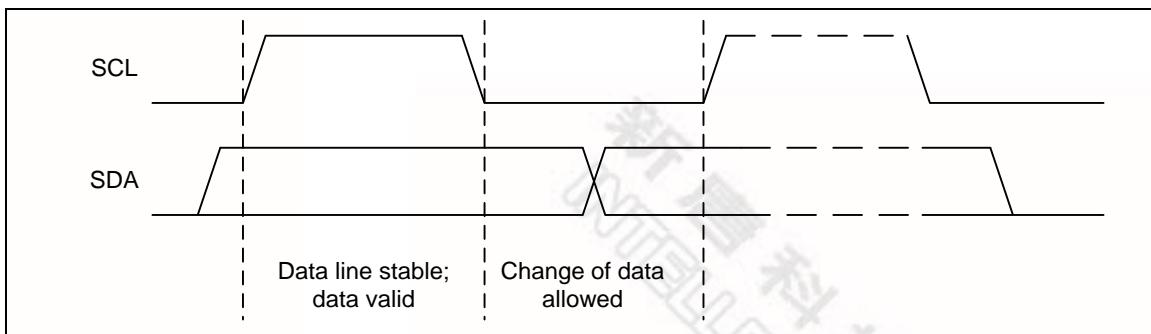
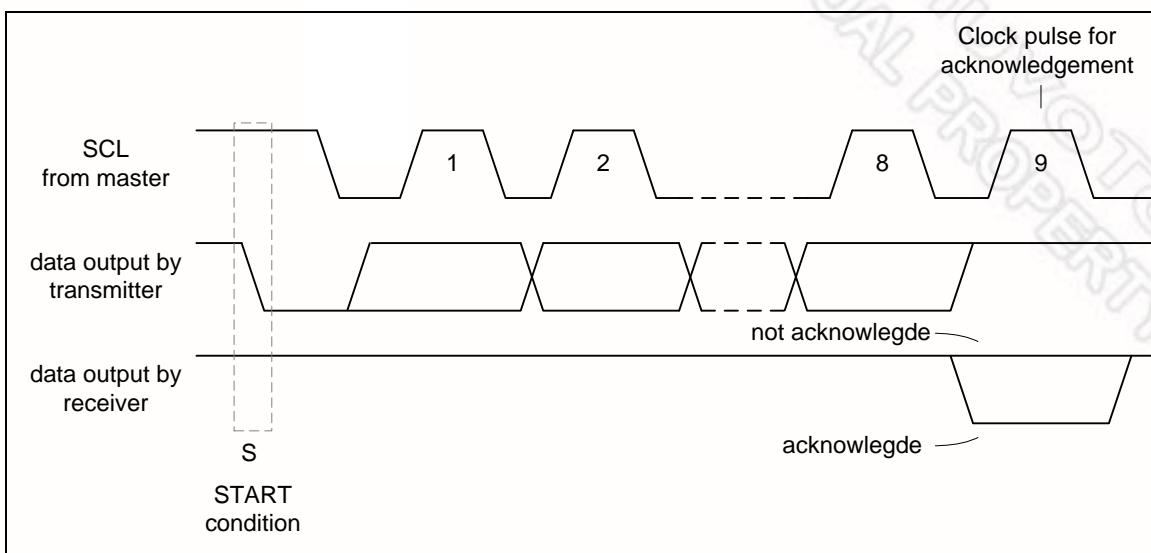
5.6.3.4 从机地址传输

起始信号后立即传输的第一个字节就是从机地址。这是一个7位设备地址跟随一个 RW 位。RW 位标志从机的信号的数据传输方向。系统中没有两个从机有相同的地址，只有被主机寻址的从机设备才会通过在第9个 SCL 时钟周期时将 SDA 拉低作为应答。

5.6.3.5 数据传输

当从机设备地址被成功识别，就可以根据 RW 位所决定的方向按一字节一字节方式进行数据传输。每个字节传输完后紧接着在第9个 SCL 时钟周期会有一个应答信号位。如果从机上产生无应答信号 (NACK)，主机可以产生停止信号来中止数据传输或者产生重复起始信号开始新一轮数据传输。

当主机作为接收设备时，发生无应答信号 (NACK)，则从机释放 SDA 线，让主机产生停止信号或重复起始信号。

图 5-23 I²C 总线上的位传输图 5-24 I²C 总线上的应答信号

5.6.4 协议寄存器

CPU 和 I²C 端口通过下列的 13 个特殊功能寄存器通讯：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 mask 寄存器，n=0~3），I2CLK（时钟速率寄存器）和 I2CTOC（定时计数器寄存器）。所有 I²C 的特殊功能寄存器的第8位至第31位都是保留的，不具备任何功能，读取返回值都为 0。

当 ENS1 (I2CON [6]) 置高，I²C 端口使能后，内部状态将由 I2CON 和 I²C 逻辑硬件控制。一旦有新的状态码产生后，会存储到 I2CSTATUS。I²C 中断标志位 SI (I2CON [3]) 也会自动置位。若此时使能中断位 EI (I2CON [7]) 为高，则会产生 I²C 中断。位字段 I2CSTATUS[7:3] 存储内部状态码，在 SI 被软件清除之前，I2CSTATUS 的最低三位始终保持低电平。寄存器基址为 0x4002_0000 和 0x4012_0000。

5.6.4.1 地址寄存器 (I2CADDR)

I²C 端口内建 4 个从机地址寄存器 I2CADDRn (n=0~3)。当 I²C 作为主机时，这四个寄存器的内容不相关。在从机模式下，位字段 I2CADDRn[7:1] 必须装载芯片自己的从机地址。当 I2CADDRn 地址与接收从机的地址符合时，I²C 硬件会作出反应。

I²C 端口支持“广播呼叫”功能。当 GC 位 (I2CADDRn [0]) 被置，I²C 端口硬件将应答广播呼叫的地址 (00H)。清 GC 位可禁用“广播呼叫”功能。

当 GC 位被置且 I²C 处于从机模式时，主机发出广播呼叫地址到 I²C 总线后，从机可以通过地址 00H 接收广播呼叫地址，然后它将跟随 GC 模式的状态。

I²C 总线控制器有 4 个地址掩码寄存器 I2CADMn (n=0~3) 支持多地址识别。当地址掩码寄存器的某位被置 1，表示可以忽略接收到的相应地址位。若该位被置 0，表示接收到的相应地址位和地址寄存器内的值完全吻合。

5.6.4.2 数据寄存器 (I2CDAT)

该寄存器包含一个准备发送或刚接收到的一个字节的串行数据。只要不在移位处理的过程，CPU 可以直接读写访问 I2CDAT[7:0]。当 I²C 处于一个确定的状态并且串行中断标志 (SI) 被置位，I2CDAT[7:0] 中的数据就一直是稳定的。在数据被移出的过程中，总线上的数据同时被移入。I2CDAT 的内容总是总线上出现的最后一个字节。因此，如果发生仲裁失败，在主发射机到从接收机传输的模式下，I2CDAT[7:0] 中的数据仍保持正确。

I2CDAT [7:0] 和应答位组成了一个 9 位的移位寄存器，应答位由 I²C 硬件控制，不能被 CPU 访问。串行数据因为应答位在 SCL 线上串行时钟脉冲的上升沿移入 I2CDAT [7:0] 时移位。当一个字节被移位到 I2CDAT [7:0] 后，则 I2CDAT [7:0] 中的数据是可用的，应答位 (ACK 或 NACK) 在第 9 个时钟脉冲由控制逻辑返回。串行数据在 SCL 时钟脉冲的下降沿从 I2CDAT [7:0] 移出，在 SCL 时钟脉冲的上升沿数据移入 I2CDAT [7:0]。

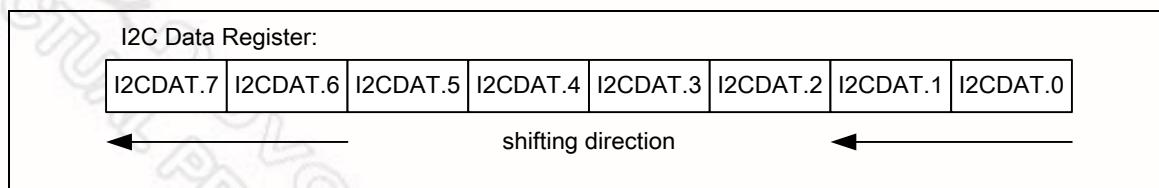


图 5-25 I²C 数据移位方向

5.6.4.3 控制寄存器 (I2CON)

CPU 可以直接读或写寄存器 I2CON [7:0]。两个受硬件影响的位是：在 I²C 硬件请求串行中断时 SI 位置位；在 STOP 条件出现在总线上时，STO 位被清除。当 ENS1 = 0 时，STO 位也会被清除。

EI 使能中断

ENS1 设置使能 I²C 串行功能控制器。当 ENS1=1，使能 I²C 串行功能。SDA 和 SCL 的多功能管脚必须设置成 I²C 功能。

STA I²C START 控制位。设置 STA 为逻辑 1 进入主机模式。如果总线空闲，则 I²C 硬件发送一个 START 或重复 START 条件到总线。

STO I²C STOP 控制位。在 I²C 主机模式，设置 STO 来传送一个 STOP 条件到总线，然后 I²C 硬件将会检查总线状况，如果检测到一个 STOP 状况，这个标志会被硬件自动清除。在 I²C 从机模式，设置 STO 复位 I²C 硬件来定义“无编址”从机模式，这表示在从机接收模式不再接收从主机传送装置发送的数据。

SI I²C 中断标志。当一个新的 I²C 状态出现在寄存器 I2CSTATUS 时，SI 标志由硬件置位，并且如果 EI (I2CON [7]) 位被置位，则产生 I²C 中断请求。SI 必须由软件通过向该位写 '1' 清零。所有的状态均列在章节 5.6.6。

AA 应答控制位。当 AA=1 先于地址或数据接收，在以下两种情况：1.) 从机正在应答主机发送的地址，2.) 接收设备正在应答发送设备发送的数据，在 SCL 线上的应答时钟脉冲期间将返回一个应答（SDA 上的低电平）。当 AA = 0 先于地址或数据接收，则在 SCL 线上的应答时钟脉冲期间将返回一个非应答（SDA 上的高电平）。

5.6.4.4 状态寄存器 (I2CSTATUS)

I2CSTATUS [7:0] 是一个 8-位只读寄存器。低三位一直为 0。I2CSTATUS [7:3] 是状态码，共有 26 个可能的状态码。所有状态均列在章节 5.6.6。当 I2CSTATUS [7:0] 的值为 F8H 时，没有串行中断请求。所有其他的 I2CSTATUS [7:3] 的值对应 I²C 的状态。当进入其中任一状态时，就会产生状态中断请求 (SI = 1)。在 SI 被硬件置位或 SI 被软件复位后一个机器周期，有效状态码出现在 I2CSTATUS[7:3] 中。

此外，00H 状态表示总线错误。总线错误发生在 START 或 STOP 条件出现在帧结构不正确的位置。不正确的位比如是在串行传输地址字节、数据字节或应答位期间。要从总线错误中恢复 I²C，STO 必须置位以及必须清除 SI 以进入无编址模式。然后清除 STO 用来释放总线，等待新的通信。当总线错误产生时，I²C 总线不能识别停止条件。

5.6.4.5 I²C 时钟波特率位 (I2CLK)

当 I²C 在主机模式下，I²C 数据的波特率由 I2CLK[7:0] 寄存器设定。其在从机模式下时是不重要的；在从机模式下，I²C 将自动与主机 I²C 设备时钟频率同步，可高达 1 MHz。

I²C 数据波特率的设定：I²C 数据波特率 = (system clock) / (4x (I2CLK [7:0] +1))。如果 system clock = 16 MHz, the I2CLK [7:0] = 40 (28H)，那么 I²C 数据波特率 = 16 MHz / (4x (40 +1)) = 97.5 Kbits/sec。

5.6.4.6 I²C 超时计数器寄存器 (I2CTOC)

MCU 提供一个 14 位超时的计数器来处理当 I²C 总线锁死时的情况。当计数功能使能后，计数器开始计数直至溢出 (TIF=1) 并要求 CPU 产生 I²C 中断或者清除 ENTI 为 0 关闭计数功能。当超时计数器使能，对 SI 标志置高会使计数器复位，再对 SI 清零会重新开始计数。如果 I²C 总线锁死，会使 I2CSTATUS 及 SI 标志不再更新，该14-位超时计数器会发生溢出从而产生 I²C 中断通知 CPU。

参考图 5-26 14-位超时计数器。用户可以写 1 清 TIF 为 0。

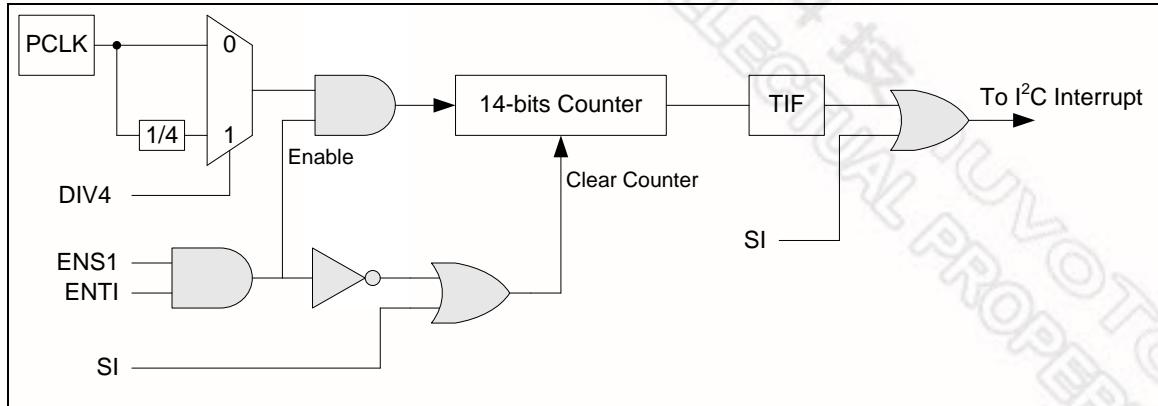


图 5-26: I²C 超时计数器框图

5.6.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
I2C0_BA = 0x4002_0000				
I2C1_BA = 0x4012_0000				
I2CON	I2Cx_BA+0x00	R/W	I ² C 控制寄存器	0x0000_0000
I2CADDR0	I2Cx_BA+0x04	R/W	I ² C 从机地址寄存器0	0x0000_0000
I2CDAT	I2Cx_BA+0x08	R/W	I ² C 数据寄存器	0x0000_0000
I2CSTATUS	I2Cx_BA+0x0C	R	I ² C 状态寄存器	0x0000_00F8
I2CLK	I2Cx_BA+0x10	R/W	I ² C 时钟分频寄存器	0x0000_0000
I2CTOC	I2Cx_BA+0x14	R/W	I ² C 超时控制寄存器	0x0000_0000
I2CADDR1	I2Cx_BA+0x18	R/W	I ² C 从机地址寄存器1	0x0000_0000
I2CADDR2	I2Cx_BA+0x1C	R/W	I ² C 从机地址寄存器2	0x0000_0000
I2CADDR3	I2Cx_BA+0x20	R/W	I ² C 从机地址寄存器3	0x0000_0000
I2CADM0	I2Cx_BA+0x24	R/W	I ² C 从机地址掩码寄存器0	0x0000_0000
I2CADM1	I2Cx_BA+0x28	R/W	I ² C 从机地址掩码寄存器1	0x0000_0000
I2CADM2	I2Cx_BA+0x2C	R/W	I ² C 从机地址掩码寄存器2	0x0000_0000
I2CADM3	I2Cx_BA+0x30	R/W	I ² C 从机地址掩码寄存器3	0x0000_0000



5.6.6 寄存器描述

I²C 控制寄存器 (I2CON)

寄存器	偏移量	R/W	描述	复位后的值
I2CON	I2C_BA+0x00	R/W	I ² C 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
EI	ENS1	STA	STO	SI	AA	Reserved	

Bits	描述	
[31:8]	Reserved	保留
[7]	EI	使能中断 1 = 使能 I ² C 中断 0 = 禁用 I ² C 中断
[6]	ENS1	I ² C 控制器使能位 1 = 使能 0 = 禁用 设置使能 I ² C 串行控制器功能。当 ENS1=1, I ² C 串行功能使能。SDA 和 SCL 对应的多功能管脚功能必须设置为 I ² C 功能。
[5]	STA	I ² C 起始控制位 设置 STA 为 1, 进入主机模式, 如果总线处于空闲状态, I ² C 硬件会送出 START 或 重复 START 条件。
[4]	STO	I ² C 停止控制位 在主机模式, 设置 STO 来传送一个 STOP 条件到总线, 然后 I ² C 硬件将会检查总线状况, 如果检测到一个 STOP 状况, 这个标志会被硬件自动清除。在 I ² C 从机模式, 设置 STO 复位 I ² C 硬件来定义“无编址”从机模式, 这表示在从机接收模式不再接收从主机传送装置发送的数据。
[3]	SI	I ² C 停止标志 当一个新的 I ² C 状态出现在寄存器 I2CSTATUS 时, SI 标志由硬件置位, 并且如果 EI (I2CON [7]) 位被置位, 则产生 I ² C 中断请求。SI 必须由软件通过向该位写 '1' 清零。
[2]	AA	应答控制位 当 AA=1 先于地址或数据接收, 在以下两种情况: 1.) 从机正在应答主机发送的地址, 2.) 接收设备正在应答发送设备发送的数据, 在 SCL 线上的应答时钟脉冲期间将返回一

		个应答（SDA 上的低电平）。 当 AA = 0 先于地址或数据接收，则在 SCL 线上的应答时钟脉冲期间将返回一个非应答（SDA 上的高电平）。
[1:0]	Reserved	保留

I²C 数据寄存器 (I2CDAT)

寄存器	偏移量	R/W	描述	复位后的值
I2CDAT	I2C_BA+0x08	R/W	I ² C 数据寄存器	0x0000_0000

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

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

I²C 状态寄存器 (I2CSTATUS)

寄存器	偏移量	R/W	描述	复位后的值
I2CSTATUS	I2C_BA+0x0C	R/W	I ² C 状态寄存器	0x0000_00F8

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
I2CSTATUS[7:3]					0	0	0

Bits	描述	
[31:8]	Reserved	保留
[7:0]	I2CSTATUS	<p>I²C 状态寄存器</p> <p>低三位一直为 0。高五位是状态码，共有 26 个可能的状态码。当 I2CSTATUS [7:0] 的值为 F8H 时，没有串行中断请求。所有其他的 I2CSTATUS [7:3] 的值对应 I²C 的状态。当进入其中任一状态时，就会产生状态中断请求 (SI = 1)。在 SI 被硬件置位或 SI 被软件复位后一个机器周期，有效状态码出现在 I2CSTATUS[7:3] 中。</p> <p>此外，00H 状态表示总线错误。总线错误发生在 START 或 STOP 条件出现在帧结构不正确的位置。不正确的位比如是在串行传输地址字节、数据字节或应答位期间。</p>

I²C 时钟分频寄存器 (I2CLK)

寄存器	偏移量	R/W	描述	复位后的值
I2CLK	I2C_BA+0x10	R/W	I ² C 时钟分频寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
I2CLK[7:0]							

Bits	描述	
[31:8]	Reserved	保留
[7:0]	I2CLK	I ² C 时钟分频寄存器 I ² C 数据波特率 = (system clock) / (4x (I2CLK+1)).

I²C 超时计数器寄存器 (I2CTOC)

寄存器	偏移量	R/W	描述	复位后的值
I2CTOC	I2C_BA+0x14	R/W	I ² C 超时计数器寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					ENTI	DIV4	TIF

Bits	描述	
[31:3]	Reserved	保留
[2]	ENTI	<p>超时计数器使能/禁用 1 = 使能 0 = 禁用</p> <p>当使能该位，则14位超时计数器将会在 SI 清零后开始计数。设置 SI 为高将会复位计数器，在 SI 清零之后，计数器会重新开始计数。</p>
[1]	DIV4	<p>超时计数器输入时钟除以 4 1 = 使能 0 = 禁用</p> <p>该位使能后，超时时间扩大 4 倍。</p>
[0]	TIF	<p>超时标志 当超时发生时，该位由 H/W 置位，如果此时 I²C 的中断使能位 (EI) 置为1，则可引发 CPU 的中断。 S/W 可写 1 清除该位。</p>

I²C 从机地址寄存器 (I2CADDRx)

寄存器	偏移量	R/W	描述	复位后的值
I2CADDR0	I2C_BA+0x04	R/W	I ² C 从机地址寄存器0	0x0000_0000
I2CADDR1	I2C_BA+0x18	R/W	I ² C 从机地址寄存器1	0x0000_0000
I2CADDR2	I2C_BA+0x1C	R/W	I ² C 从机地址寄存器2	0x0000_0000
I2CADDR3	I2C_BA+0x20	R/W	I ² C 从机地址寄存器3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
I2CADDR[7:1]							GC

Bits	描述	
[31:8]	Reserved	保留
[7:1]	I2CADDR	I²C 地址寄存器 主机模式下，该寄存器内容没有意义。从机模式下，高七位作为芯片本身地址。如果地址符合，I ² C 硬件将会自动应答。
[0]	GC	广播呼叫功能 0 = 禁用广播呼叫功能 1 = 使能广播呼叫功能

I²C 从机地址掩码寄存器 (I2CADMx)

寄存器	偏移量	R/W	描述	复位后的值
I2CADM0	I2C_BA+0x24	R/W	I ² C 从机地址掩码寄存器0	0x0000_0000
I2CADM1	I2C_BA+0x28	R/W	I ² C 从机地址掩码寄存器1	0x0000_0000
I2CADM2	I2C_BA+0x2C	R/W	I ² C 从机地址掩码寄存器2	0x0000_0000
I2CADM3	I2C_BA+0x30	R/W	I ² C 从机地址掩码寄存器3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
I2CADM[7:1]							Reserved

Bits	描述	
[31:8]	Reserved	保留
[7:1]	I2CADM	<p>I²C 地址掩码寄存器</p> <p>1 = 使能掩码（接收到的相应地址位不予辨识）</p> <p>0 = 禁用掩码（接收到的相应地址必须完全符合地址寄存器）</p> <p>I²C 总线控制器有4个地址掩码寄存器，支持多地址识别。当地址掩码寄存器的某位被置‘1’，表示接收到的地址的相应位可忽略。如果该位被置‘0’，则表示接收到的地址的相应位必须和地址寄存器中的完全一致。</p>
[0]	Reserved	保留



5.6.7 操作模式

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

在实际应用中，I²C 端口可以作为主机和从机。在从机模式，I²C 端口寻找自身从机地址和广播呼叫地址，如果任一地址被检测到，从机准备接收或发送数据（通过设置AA位），应答信号在第9个时钟脉冲时发送，此时，如果中断已经使能，则主从机设备发生中断请求。在主控芯片要成为总线主机时，在进入主机模式之前，硬件等待总线空闲以使从机动作不被中断，在主机模式，如果总线仲裁失败，I²C 立即切换到从机模式，并检测自身从机地址。

5.6.7.1 主机传送模式

当 SCL 线上输出时钟信号时，SDA 线上输出数据。传输的第一个字节包含接收设备的 7-位从机地址和数据方向位。在该模式下，数据方向位为 0，在流程图中用 'W' 表示。因此第一个传输的字节是 SLA+W。串行数据一次传送 8-位。每传送完一个字节后，会收到一个应答位。起始和停止条件的输出分别表示传送的开始和结束。

5.6.7.2 主机接收模式

在该模式下，数据方向为 (R/W) 将为 1，在流程图中用 'R' 表示。因此第一个传输的字节是 SLA+R。当 SCL 输出时钟信号时，串行数据通过 SDA 进行接收。串行数据每次接收 8-位。每接收到一个字节，发送一个响应位。起始和停止条件的输出分别表示串行传输的开始和结束。

5.6.7.3 从机接收模式

通过 SDA 和 SCL 接收串行数据和串行时钟。每接收到一个字节，发送一个响应位。起始和停止条件将会被分别识别为串行传输的开始和结束。地址识别由硬件在接收从机地址和数据方向后完成。

5.6.7.4 从机传送模式

第一个字节是在从机接收模式接收并处理的，然后传输方向位被反转。当 通过 SCL 线接收串行时钟时，I²C 通过 SDA 线传送数据。起始和停止条件将会被分别识别为串行传输的开始和结束。

5.6.8 五种操作模式中的数据传输

五种操作模式分别是：主机/传送，主机/接收，从机/传送，从机/接收和广播呼叫模式。在 SI 位清除后，I²C 中的 STA, STO 和 AA 位将决定 I²C 硬件的下一个状态。一个新的动作完成后，将更新一个新的状态码并置位 SI 标志。如果 I²C 中断控制位 EI (I²CON[7]) 被置位，可在中断服务子程序中根据新的状态码执行相应的动作。

数据传输的每种模式见 图 5-27 到 图 5-32。

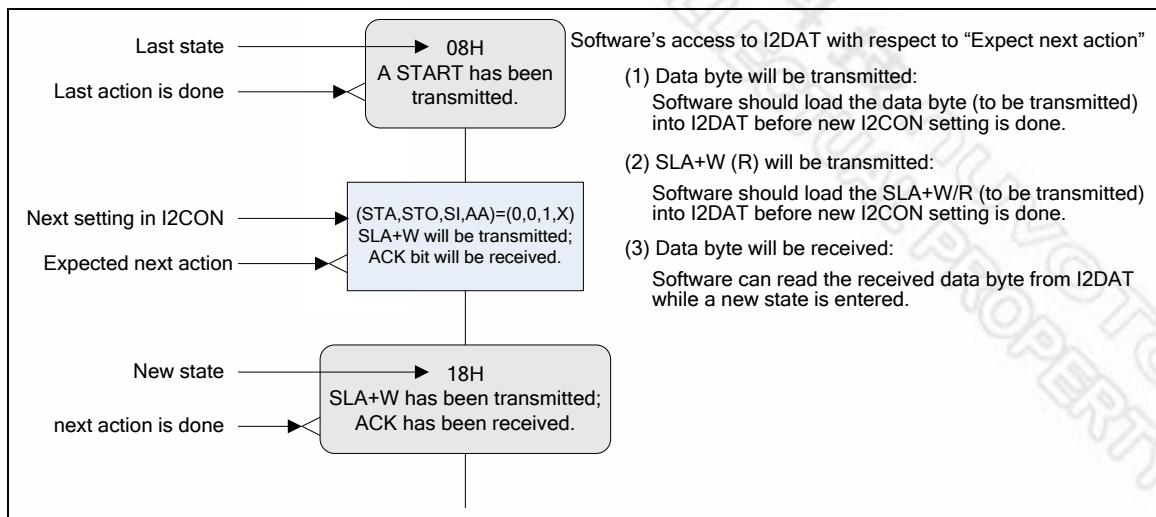


图 5-27 传输流程图

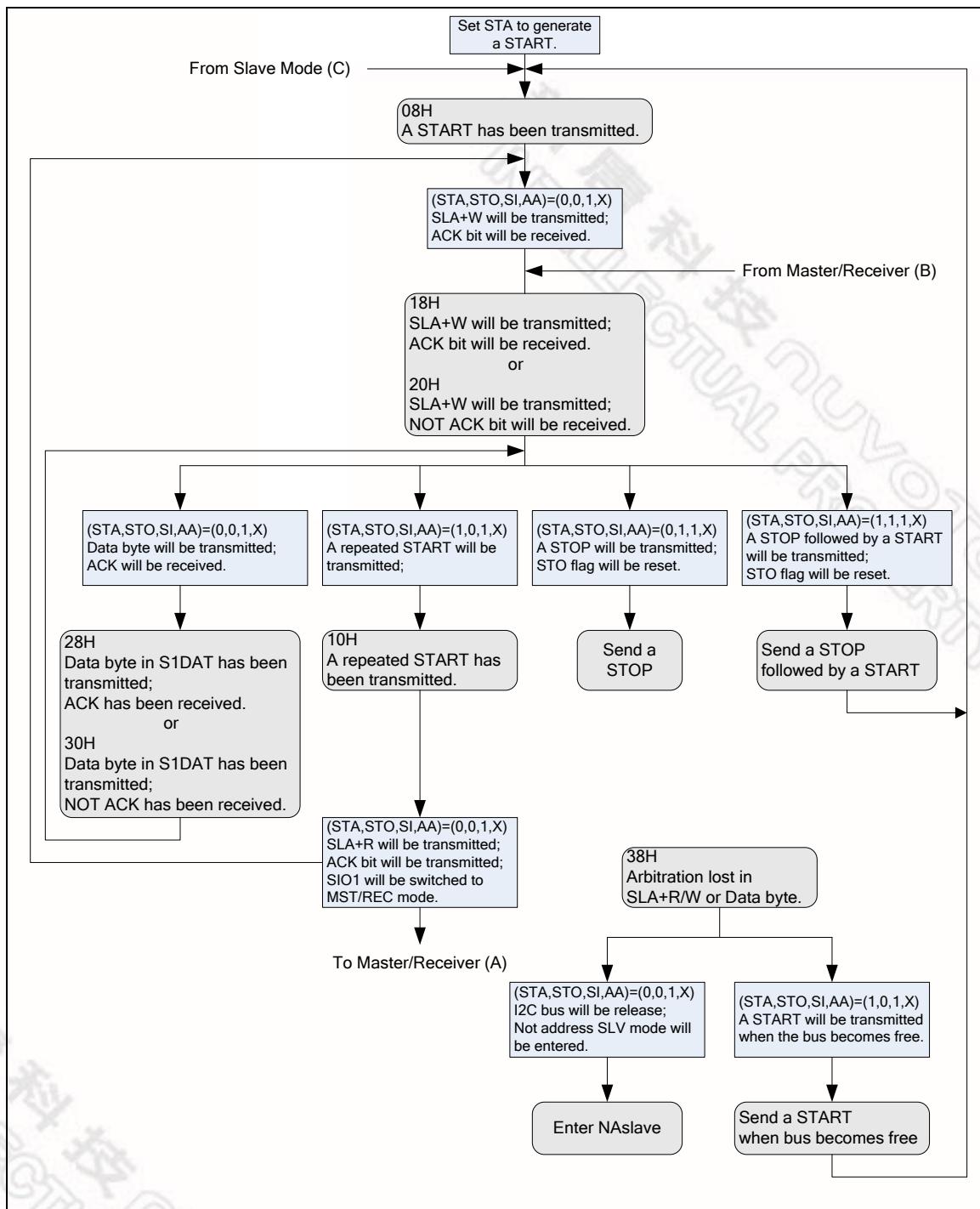


图 5-28 主机传送模式

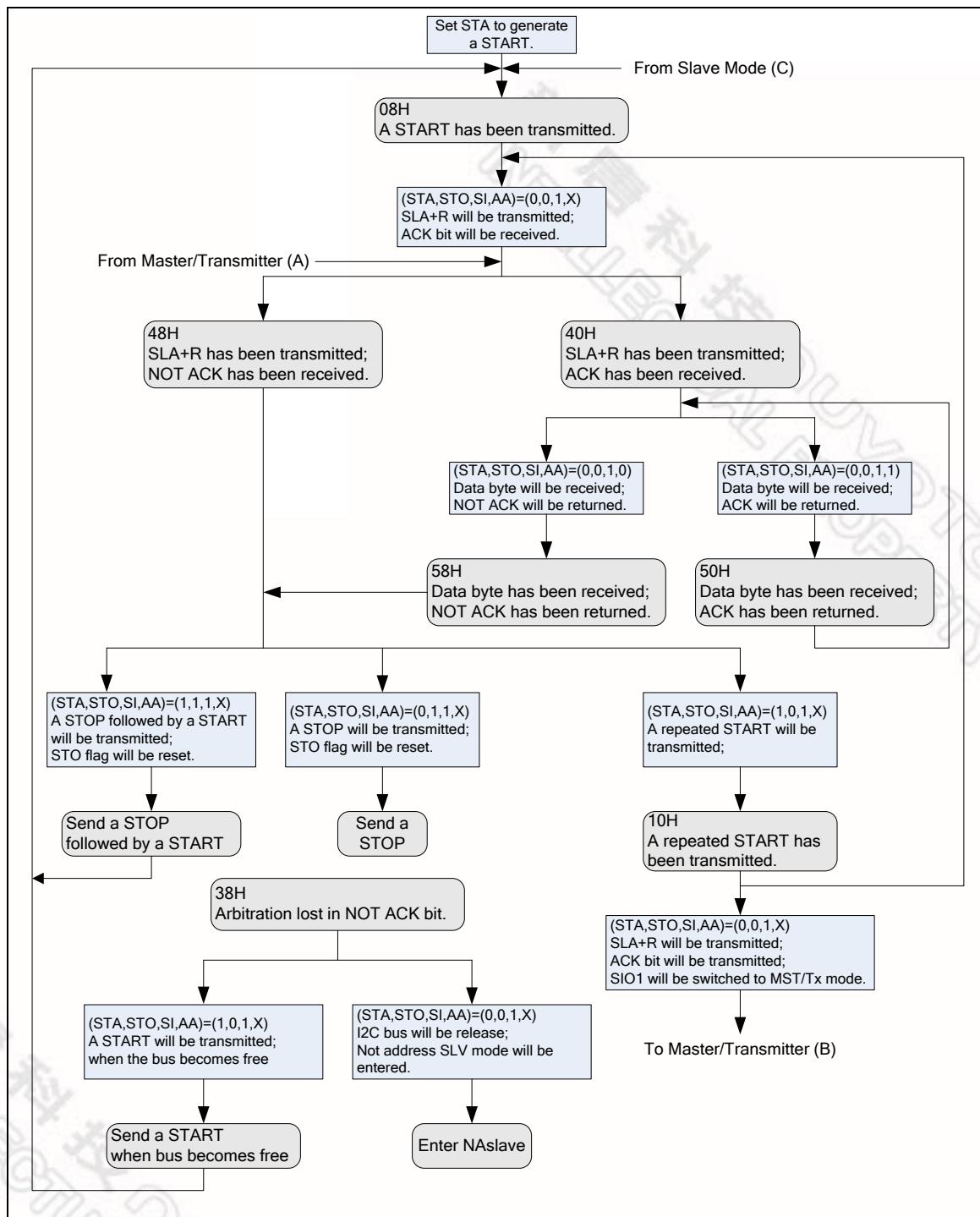


图 5-29 主机接收模式

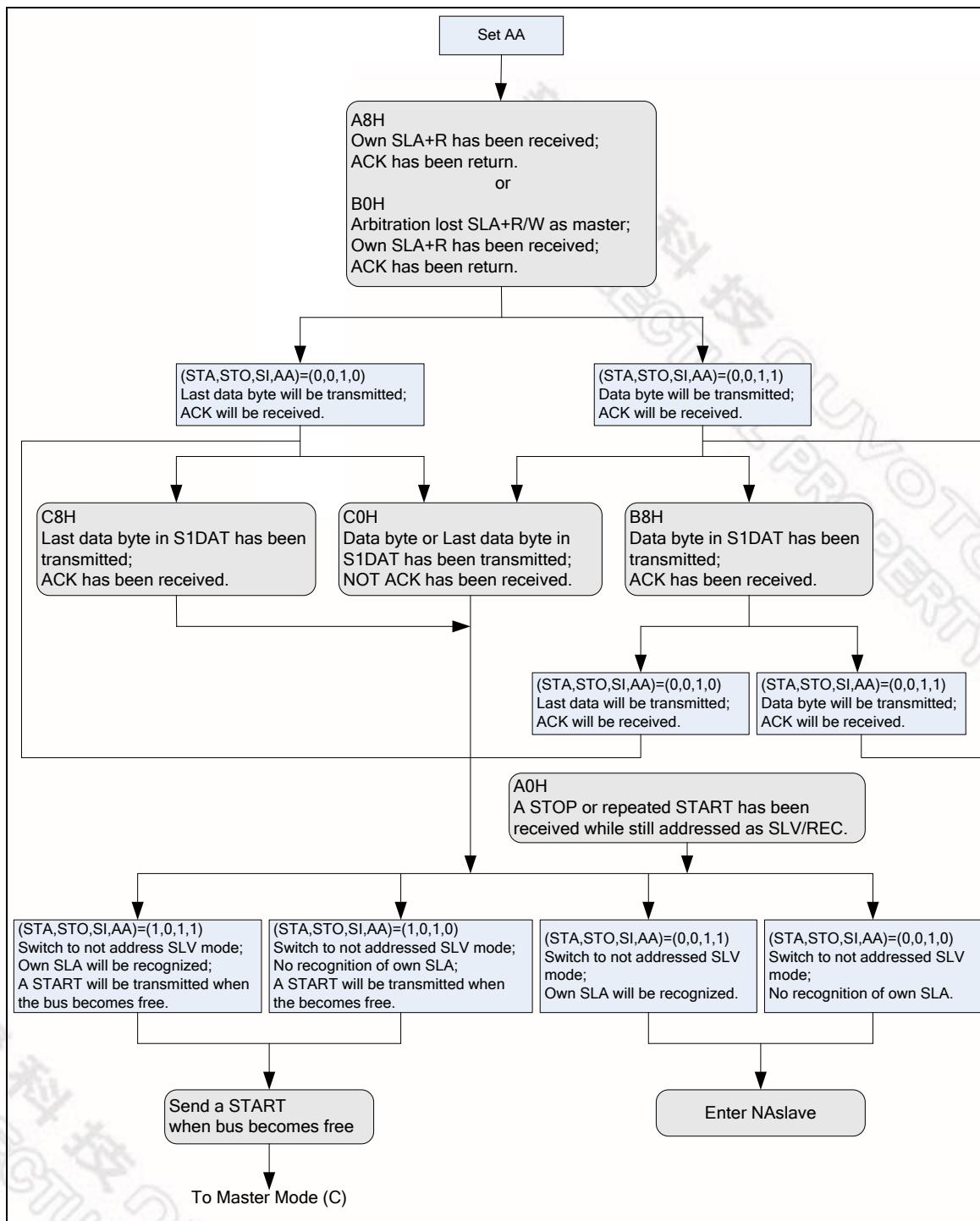


图 5-30 从机传送模式

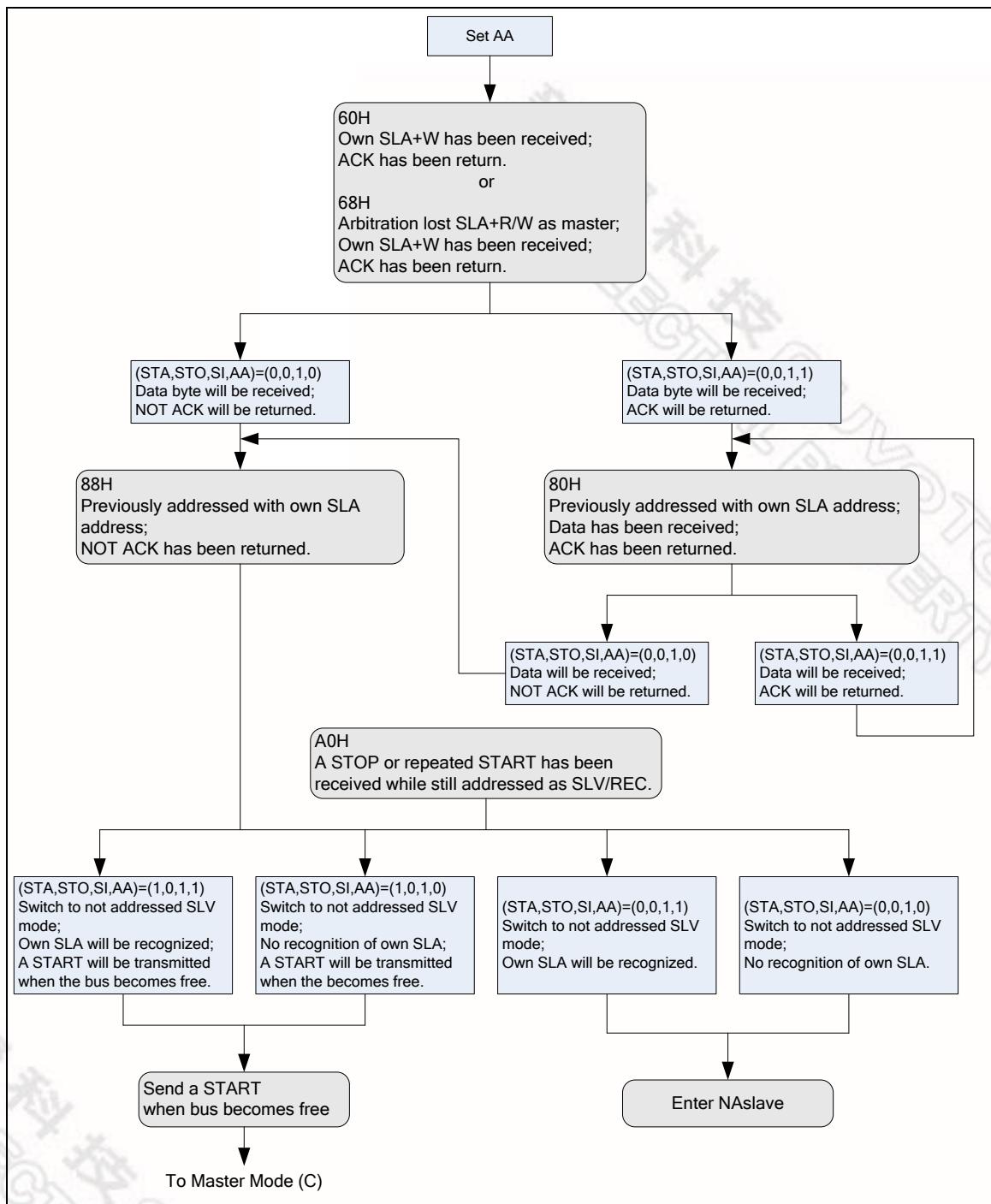


图 5-31 从机接收模式

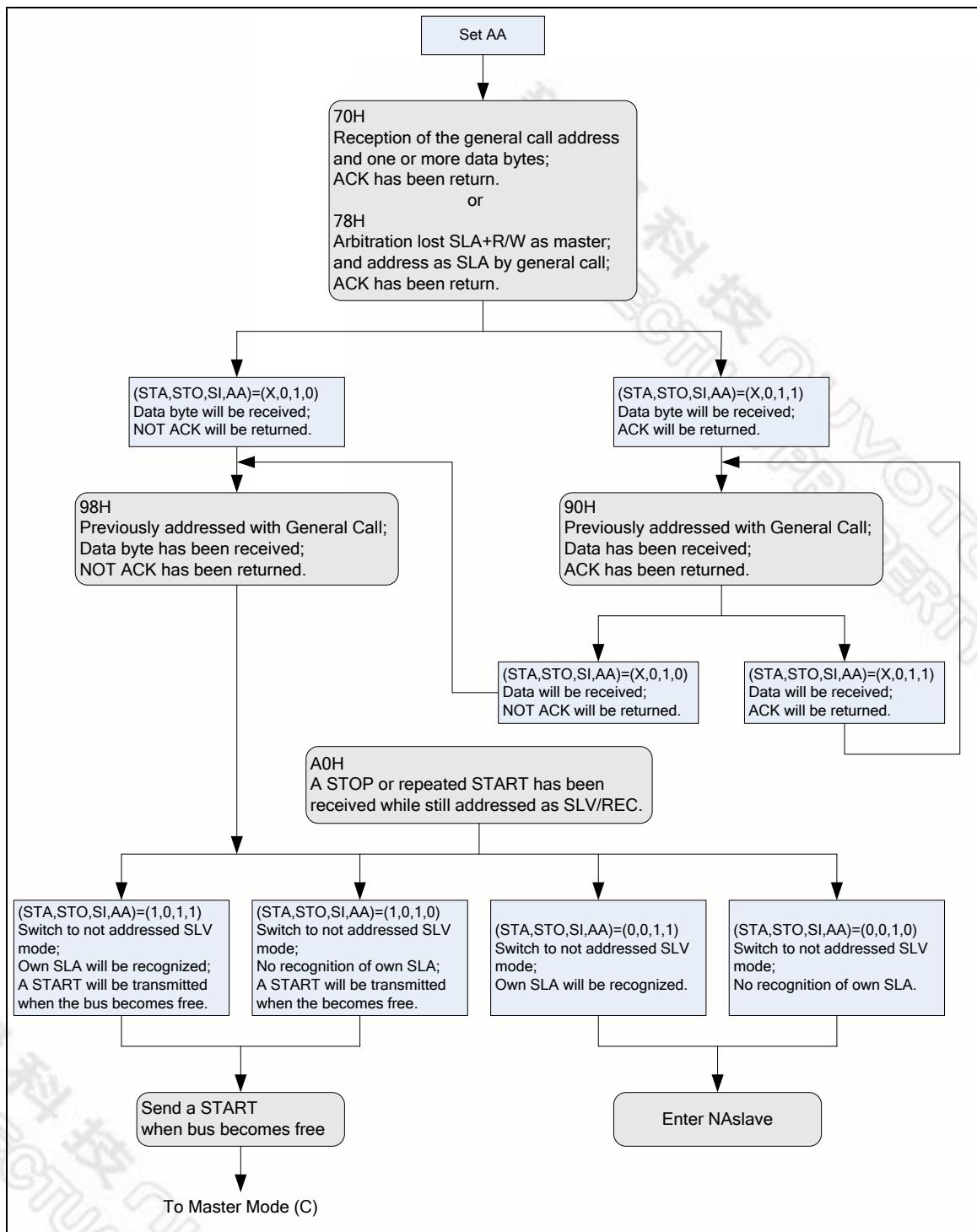


图 5-32 广播呼叫模式

5.7 PWM 发生器和捕捉定时器 (PWM)

5.7.1 概述

NuMicro™ NUC130/NUC140 有 2 组 PWM，支持 4 组 PWM 发生器，这些发生器可以配置成 8 个独立的 PWM 输出 PWM0~PWM7，或 4 组互补的 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 发生器带有一个 8-位预分频，一个提供 5 级时钟源 (1, 1/2, 1/4, 1/8, 1/16) 的时钟除频器，两个包含 2 个时钟选择的 PWM 定时器，两个用于 PWM 周期控制的 16 位 PWM 向下计数计数器，两个 16 位比较器用于 PWM 占空比控制及死区发生器。4 组 PWM 发生器提供 8 个独立的由硬件置位的 PWM 中断标志，当相应的 PWM 向下计数周期达到 0 时触发中断。每个 PWM 中断源通过相应的使能位允许 CPU 来请求 PWM 中断。PWM 发生器可以定义为产生单周期 PWM 周期信号的单触发模式或连续输出 PWM 波形的连续模式。

当 PCR.DZEN01 置位，PWM0 与 PWM1 形成互补的 PWM 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 定时器和死区发生器 2, 4, 6 控制。参考图 5-33 到图 5-40 查看 PWM 定时器的结构。

为防止 PWM 输出抖动不稳定波形，16 位向下计数计数器和 16 位比较器采用两组缓存。当用户向计数器/比较器寄存器内写入值时，只有当计数器/比较器的值计数到 0 后，写入值才会被重新写入计数器/比较器。该两组缓存可以避免 PWM 输出时产生干扰波形。

当 16 位向下计数计数器达到 0 时，中断请求产生。如果 PWM 定时器被定义为连续模式，当向下计数器达到 0 时，会自动重新导入 PWM 计数寄存器 (CNRx) 的设定值并重新开始运行下一个周期。如果定时器设为单触发模式，向下计数器达到 0 时停止计数，并产生中断请求。

比较器数据用于脉冲高宽度调节，计数控制逻辑在向下计数器计数到比较值时将 PWM 输出变高。

PWM-定时器其中之一特性是数字输入捕捉功能。若捕捉功能使能，则 PWM 的输出管脚会被切换成捕捉输入模式。捕捉器 0 和 PWM0 共用 PWM0 中的定时器；捕捉器 1 和 PWM1 共用 PWM1 定时器，以此类推。因此用户在使用捕捉功能之前，必须预先配置 PMW 定时器。使能捕捉器特性后，当输入通道有上升沿时，捕捉器总是将 PWM-计数器的值锁存到捕捉上升沿锁存寄存器 (CRLR)，当输入通道有下降沿时，捕捉器总是将 PWM-计数器的值锁存到捕捉下降沿锁存寄存器 (CFLR)。捕捉器通道 0 中断可以通过编程设定 CCR0.CRL_IE0[1]（上升沿触发中断有效）和 CCR0.CFL_IE0[2]（下降沿触发中断有效）来决定中断发生的条件。同样设定 CCR0.CRL_IE1[17] 和 CCR0.CFL_IE1[18]，可以设定捕捉器通道 1。类似的捕捉器每组的通道 2 到 3 通过设定 CCR2 的相应位也有同样的特性。对每组而言，每当捕捉器触发中断 0/1/2/3 时，PWM 计数器 0/1/2/3 也会同时被重置。

最大的捕捉频率由捕捉中断延迟决定。当捕捉中断发生时，软件至少执行以下三步：第一步，读 PIIR 获取中断源，第二步，读 CRLRx/CFLRx(x=0~3) 获取捕捉值，以及最后写 1 清 PIIR 为 0。如果中断延迟花 T0 完成，捕捉信号在 (T0) 间隔内必须不能改变。此条件下，最大捕捉频率为 1/T0。例如：

HCLK = 50 MHz, PWM_CLK = 25 MHz, 中断延时为 900 ns

因此最大捕捉频率为 $1/900\text{ns} \approx 1000 \text{ kHz}$



5.7.2 特征

5.7.2.1 PWM 功能特性:

- PWM 组有两个 PWM 发生器。每个 PWM 支持 8-位预分频器，一个时钟除频器，两个 PWM 定时器（向下计数），一个死区发生器和两个 PWM 输出。
- 最高 16-位 解析度
- PWM 中断请求与 PWM 周期同步
- One-shot 或 Auto-reload 模式
- 最高 2 个 PWM 组(PWMA/PWMB) 可支持 8 路 PWM 通道或 4 对 PWM 通道

5.7.2.2 捕捉功能特征:

- 与 PWM 发生器共用定时器模块
- 支持 8 个捕捉输入通道，共享 8 个 PWM 输出通道
- 每个通道支持 1 个上升沿锁存寄存器 (CRLR)，一个下降沿锁存寄存器 (CFLR) 和 捕捉中断标志 (CAPIFx)

5.7.3 框图

图 5-33 到 图 5-40 描述 PWM 对的结构（PWM-Timer 0/1 为一对，PWM-Timer 2/3 为另一对，依此类推）。

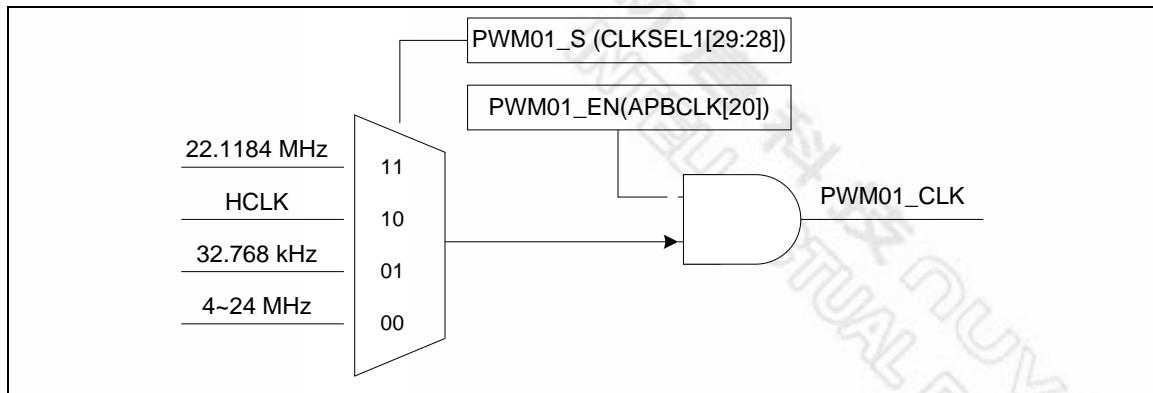


图 5-33 PWM 发生器 0 时钟源控制

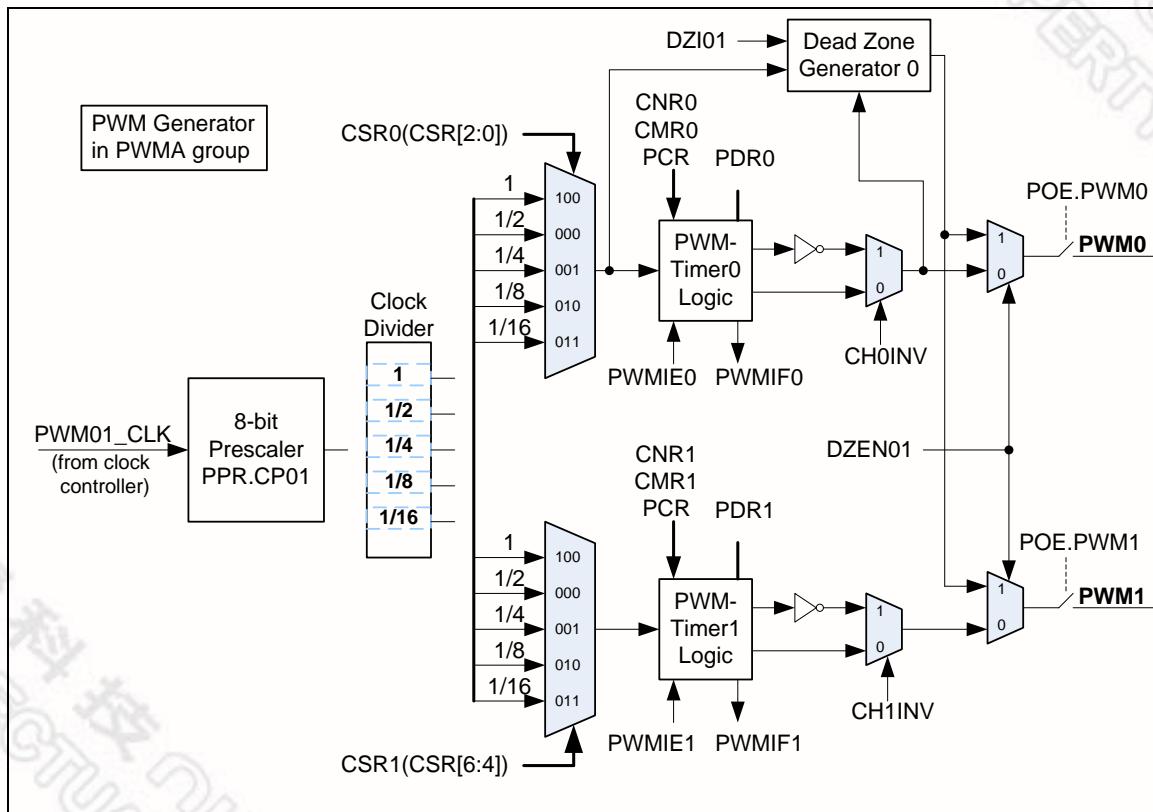


图 5-34 PWM 发生器 0 结构框图

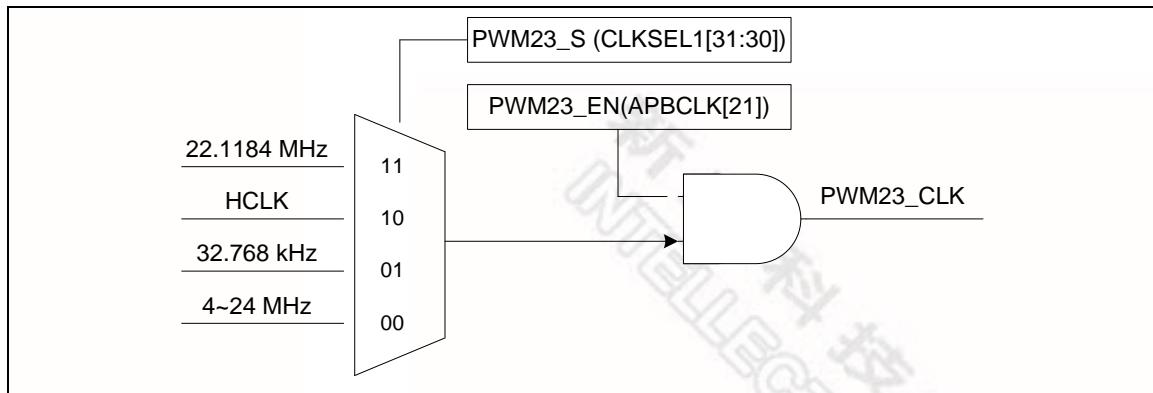


图 5-35 PWM 发生器 2 时钟源控制

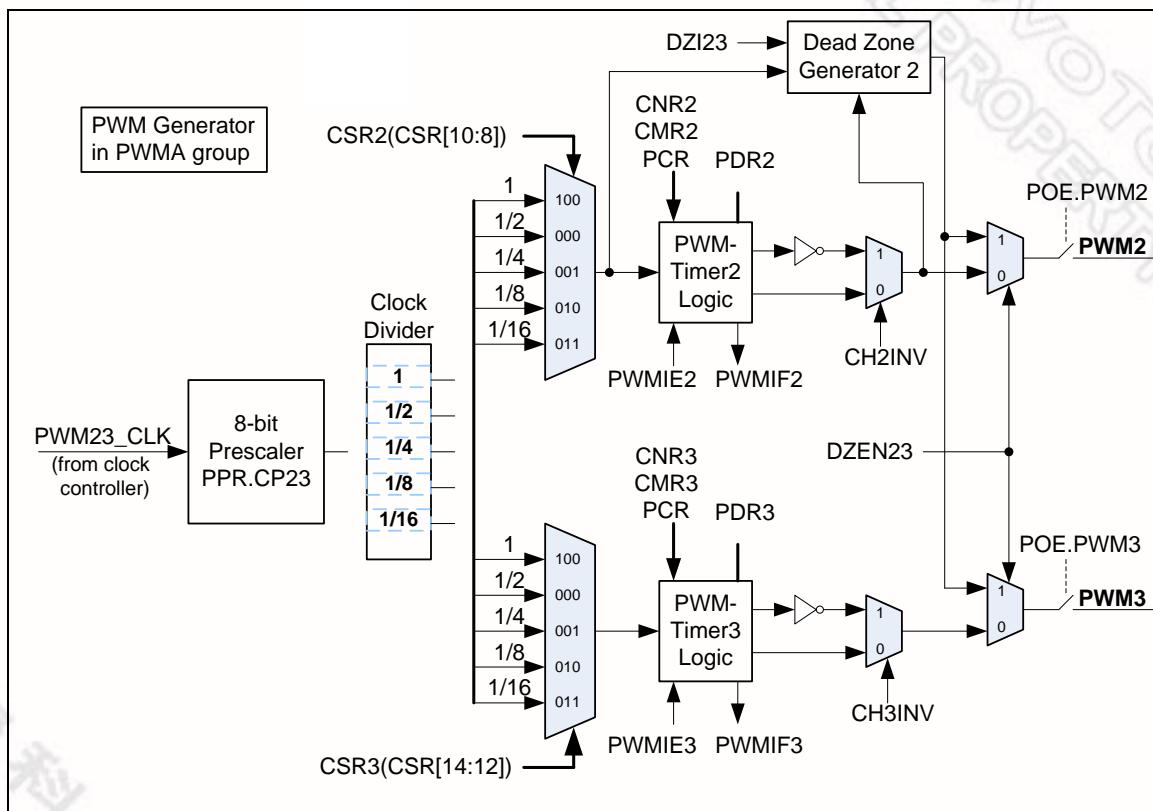


图 5-36 PWM 发生器 2 结构框图

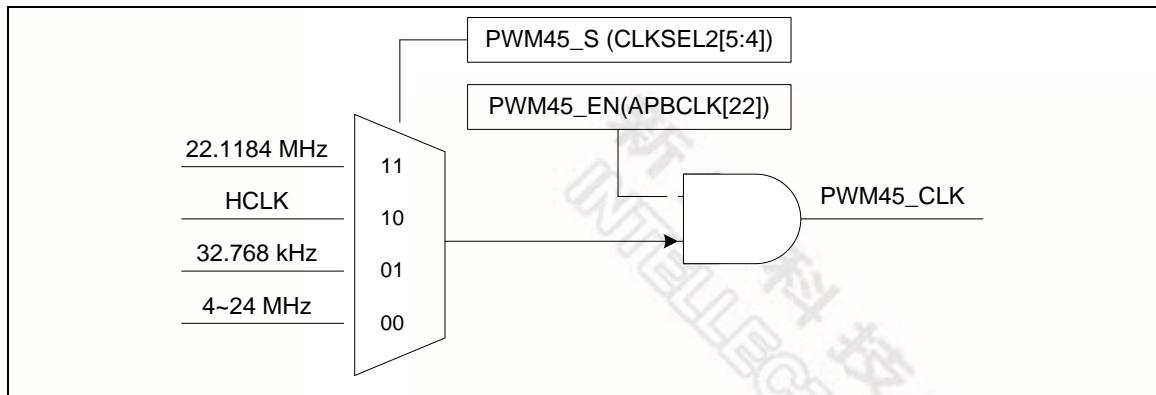


图 5-37 PWM 发生器 4 时钟源控制

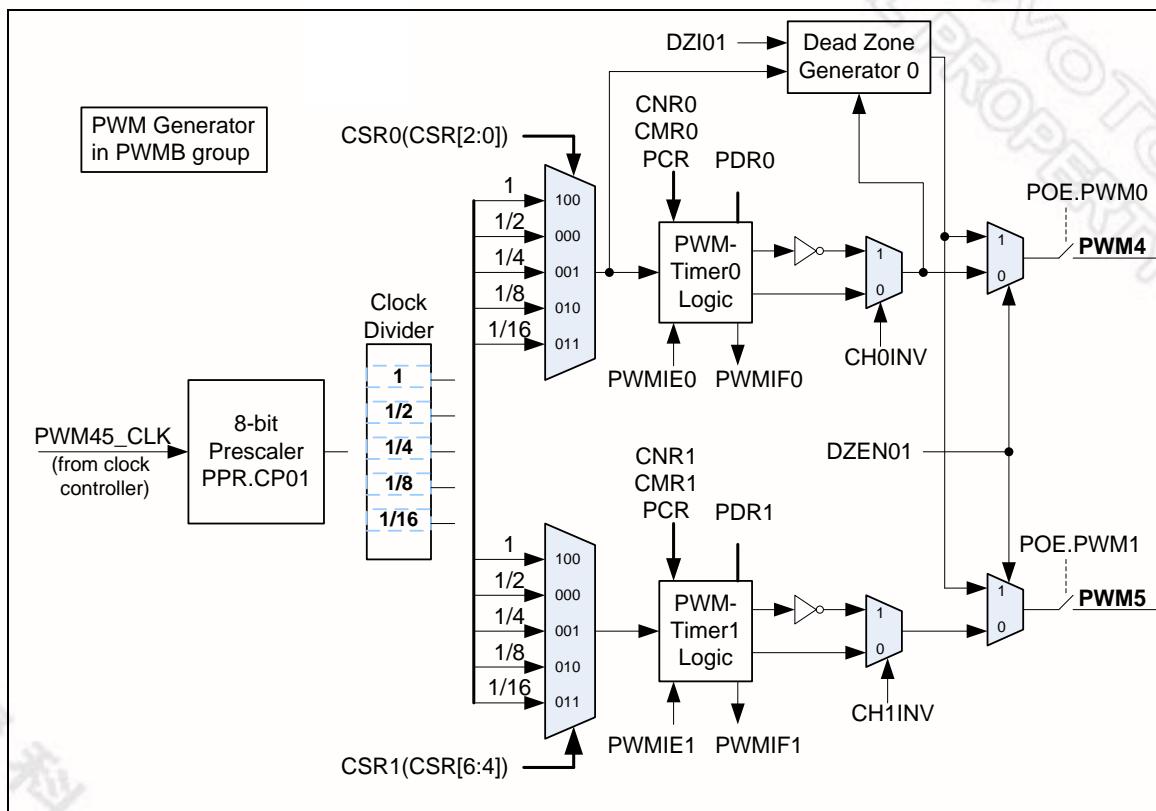


图 5-38 PWM 发生器 4 结构框图

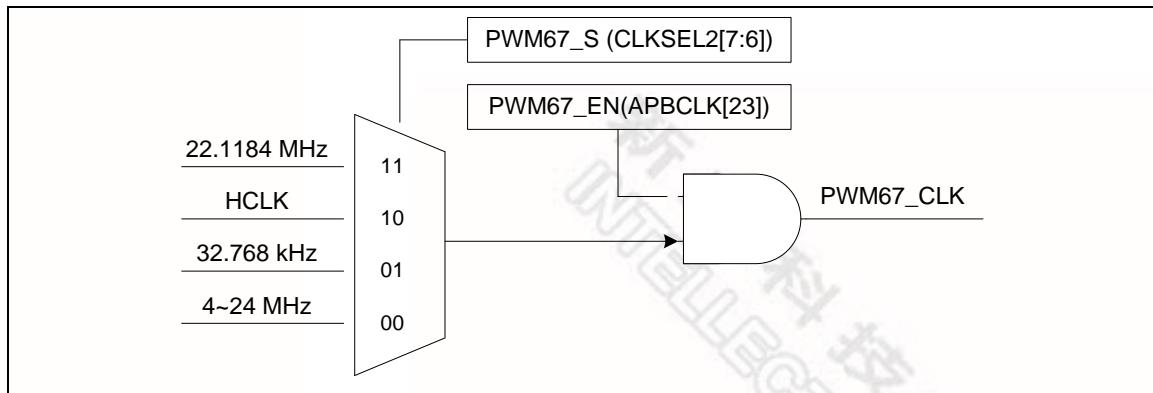


图 5-39 PWM 发生器 6 时钟源控制

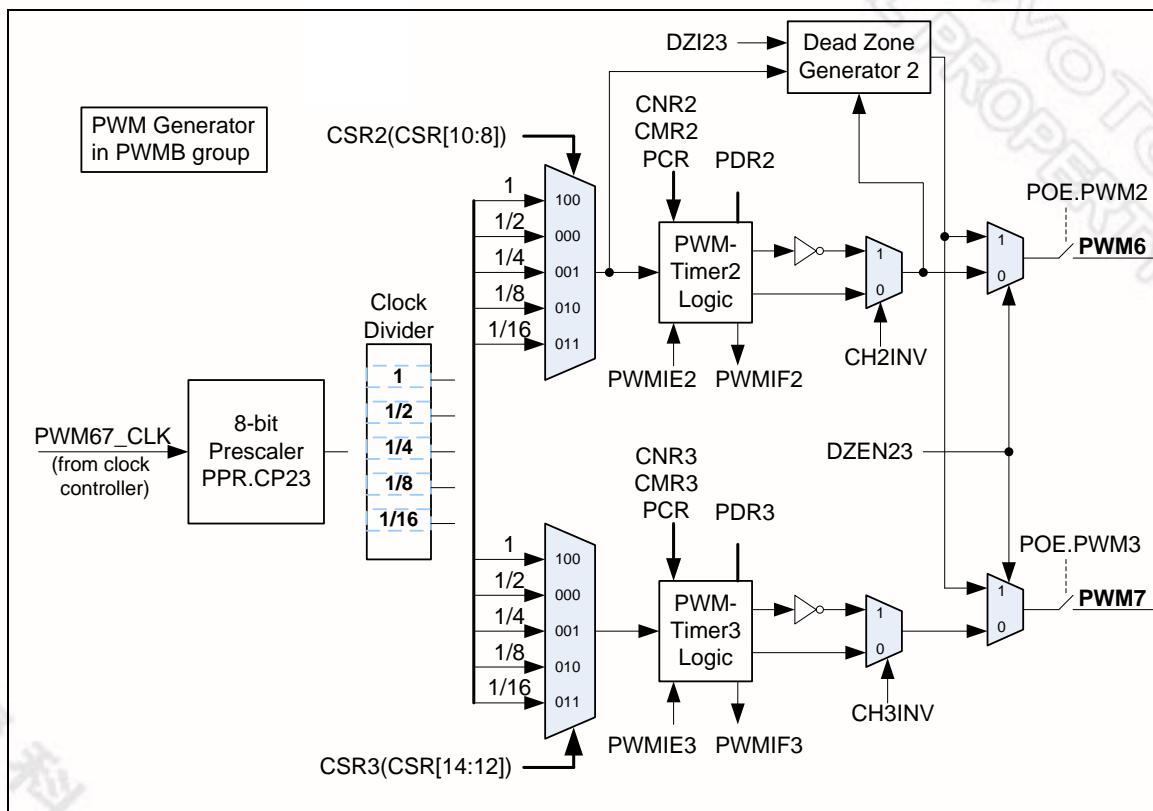


图 5-40 PWM 发生器 6 结构框图

5.7.4 功能描述

5.7.4.1 PWM-定时器操作

PWM 周期和占空比控制由向下计数的 PWM 寄存器 (CNR) 和 PWM 比较寄存器 (CMR) 配置。PWM-定时器工作时序如图 5-42。脉宽调制的公式如下, PWM-定时器的图例如图 5-41。注: 在 PWM 功能使能前, MCU 相应的 GPIO 管脚必须配置成 PWM 功能 (使能 POE 并禁用 CAPENR)。

- PWM 频率 = $\text{PWM}_{\text{xy_CLK}} / [(\text{prescale}+1) * (\text{clock divider}) * (\text{CNR}+1)]$; 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 输出为高
- $\text{CMR} < \text{CNR}$: PWM 低脉宽 = $(\text{CNR}-\text{CMR}) \text{ unit}[1]$; PWM 高脉宽 = $(\text{CMR}+1) \text{ unit}$
- $\text{CMR} = 0$: PWM 低脉宽 = $(\text{CNR}) \text{ unit}$; PWM 高脉宽 = 1 unit

Note: [1] Unit = 一个 PWM 时钟周期。

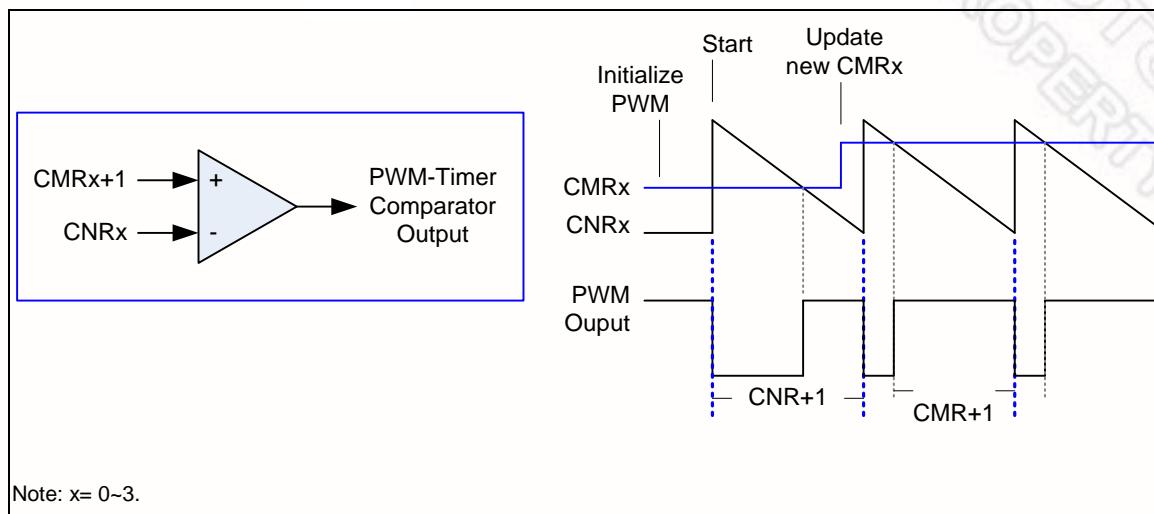


图 5-41 PWM-定时器内部比较器输出

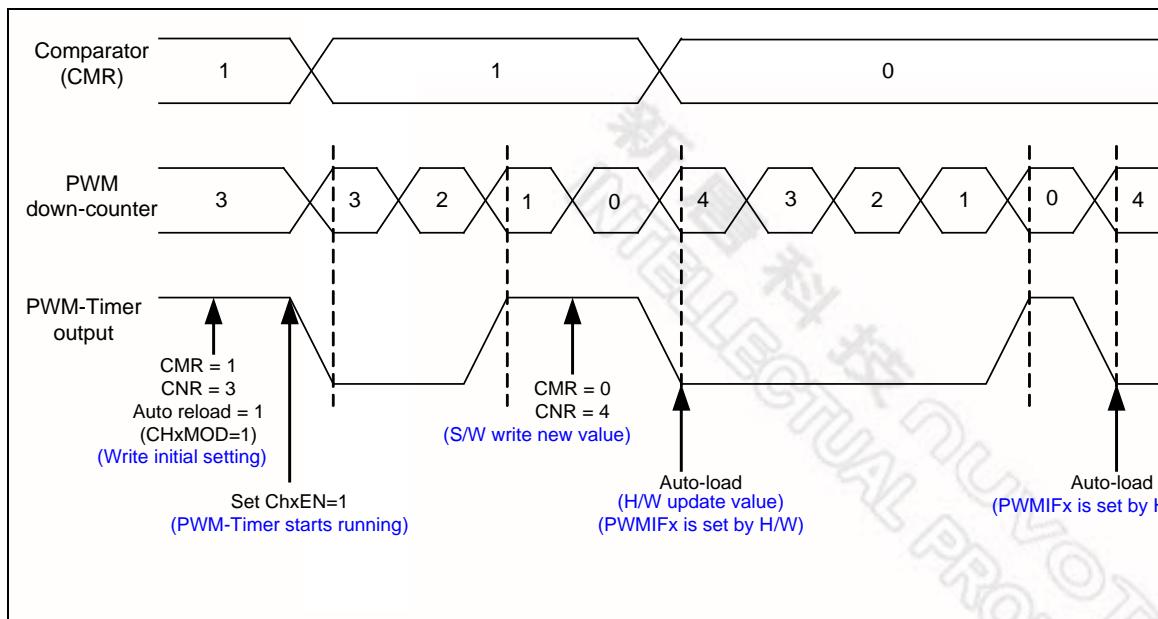


图 5-42 PWM-定时器操作时序

5.7.4.2 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。寄存器预先设定的值，在一个周期完成后，可以自动重载。PWM 计数器的值写入 CNRx，并可从 PDRx 读出当前 PWM 计数器的值。

PWM 控制寄存器 (PCR) 中 CH0MOD 位定义 PWM0 是自动重载模式或是单触发模式。如果 CH0MOD 设为 1，当 PWM 计数器计到 0，自动重载 CNR0 值到 PWM 计数器。如果 CNR0 设为 0，PWM 计数器计数到 0 后，将暂停运行。如果此时 CH0MOD 也设为 0，计数器会立即停止。PWM1~PWM7 运行状态与 PWM0 相同。

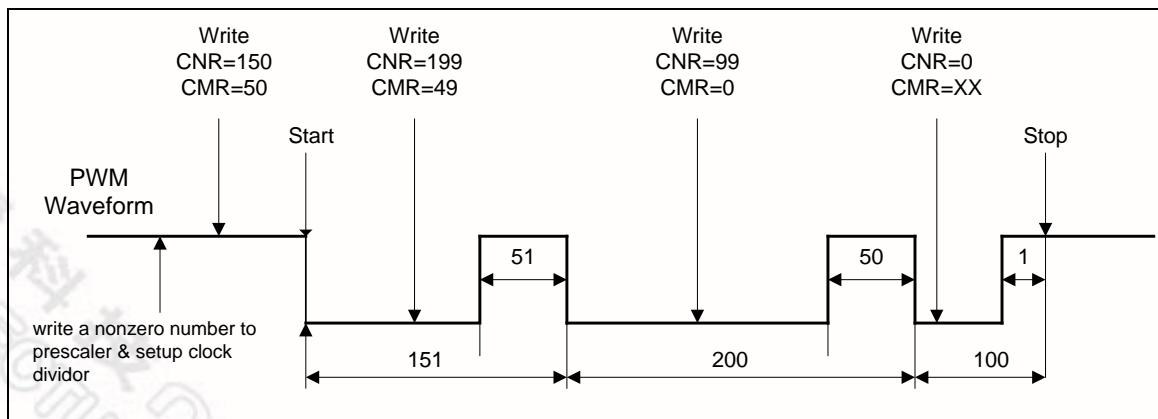


图 5-43 PWM 双缓存图解

5.7.4.3 可调占空比

双缓存允许 CMRx 字在当前周期的任意时刻改写。导入值会在下一个周期生效。

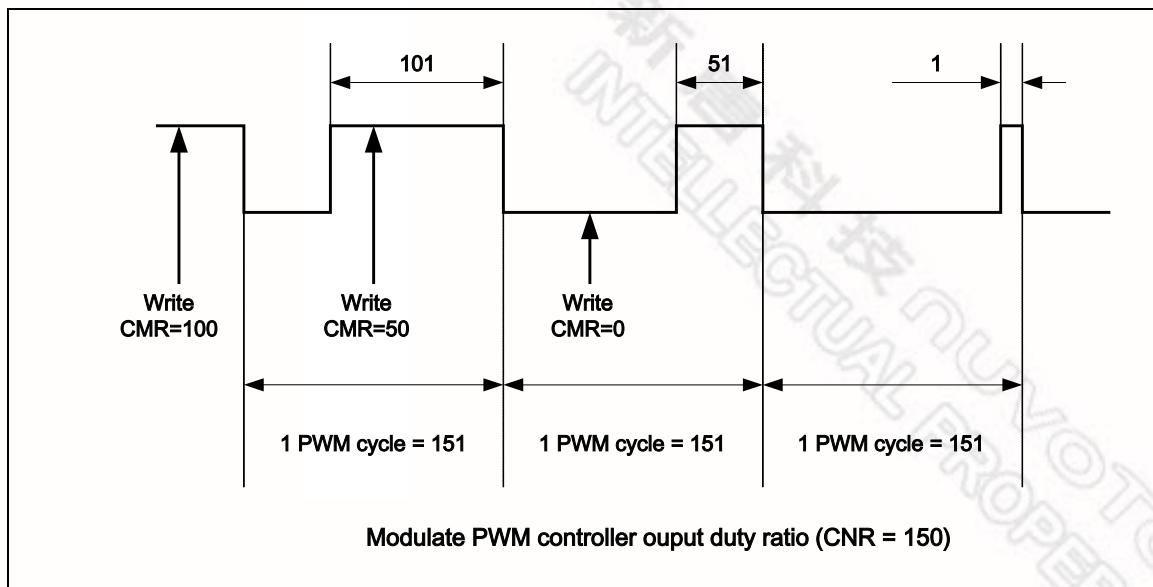


图 5-44 PWM 控制输出占空比

5.7.4.4 死区发生器

PWM 控制器提供死区发生器，用于电器器件保护。该功能在 PWM 上升沿输出时产生可编程的延迟时间。用户可通过编程 PPRx.DZI 确定死区间隔。

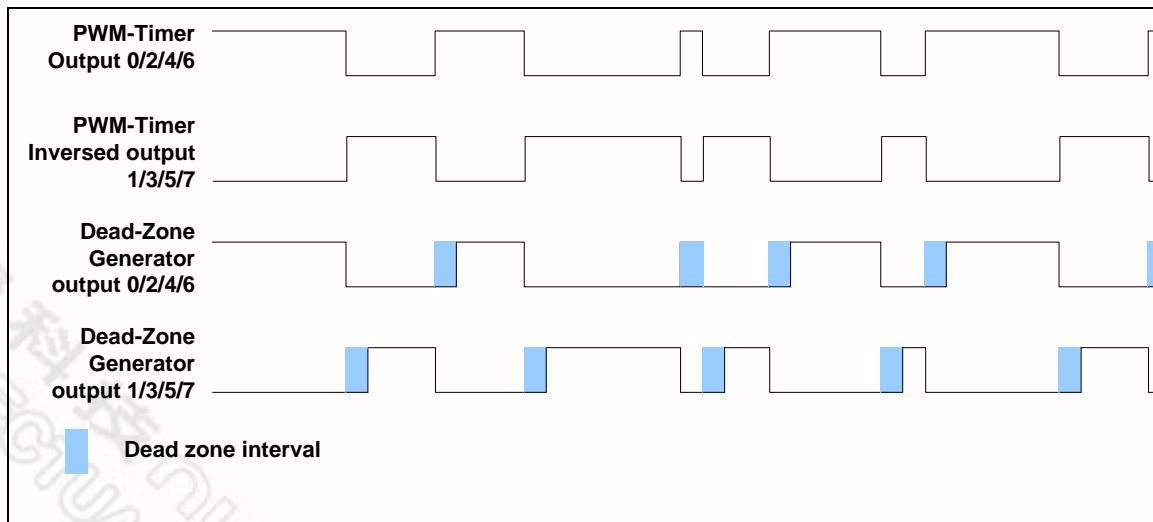


图 5-45 PWM-对输出带死区发生器操作

5.7.4.5 捕捉操作

捕捉器0和 PWM0 共享 PWM0 中的一个定时器，捕捉器1和 PWM1使用另一组定时器，以此类推。当输入信号有上升沿转变时，PWM 计数器的值将存入 CRLRx 寄存器；当输入信号有下降沿转变时，PWM 计数器的值将存入 CFLRx 寄存器。捕捉器通道 0 可以通过设定 CCR0[1]（上升沿触发中断有效）和 CCR0[2]（下降沿触发中断有效）进行编程来决定中断产生的条件。同样可以通过设定 CCR0[17] 和 CCR0[18] 来设定捕捉器通道 1，依此类推。每当捕捉控制器分发捕捉中断时，相应的 PWM 计数器会同时重新装载 CNRx 的值。注：对于捕捉通道，相应的 GPIO 管脚必须配置成捕捉功能（禁用 POE 并使能 CAPENx）。

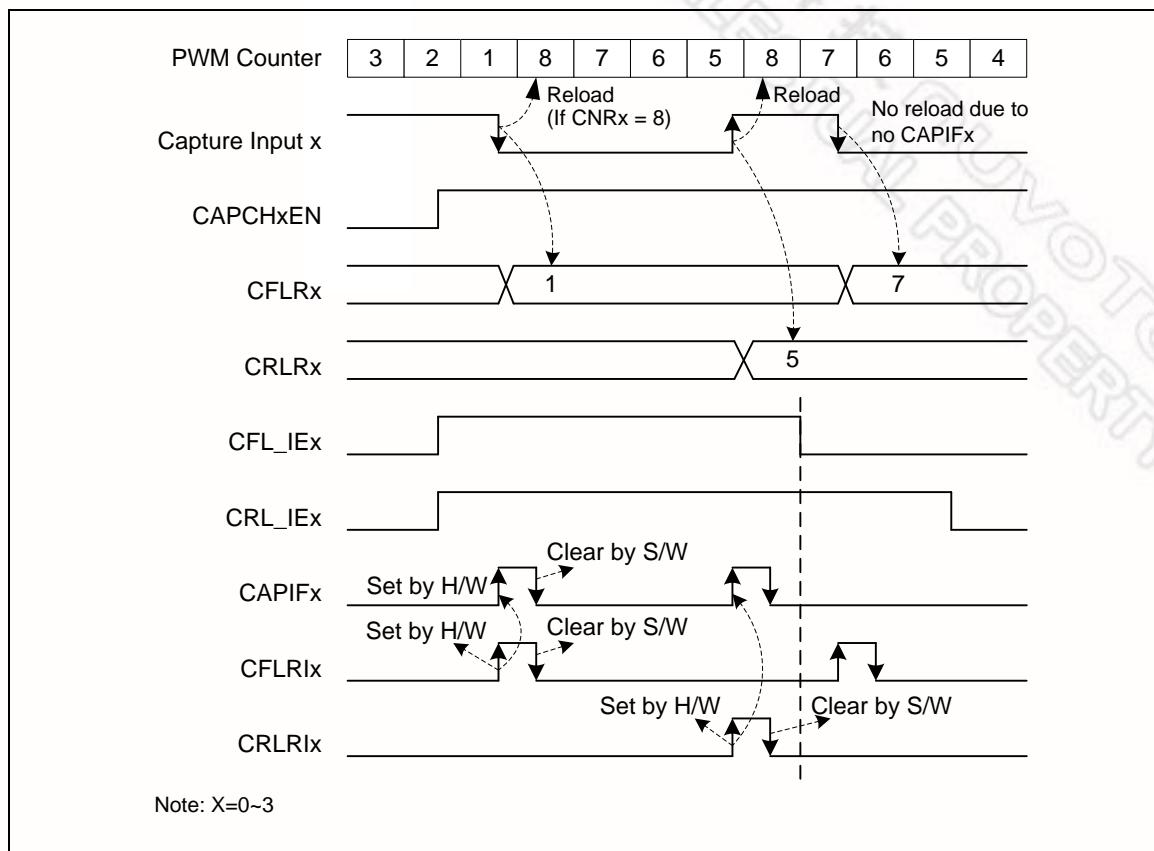


图 5-46 捕捉操作时序

在这个范例中，CNR 为 8：

1. 当捕捉中断标志 (CAPIFx) 被置位时，PWM 计数器将重载 CNRx.
2. 通道低脉宽为 (CNR + 1 - CRLR).
3. 通道高脉宽为 (CNR + 1 - CFLR).

5.7.4.6 PWM-定时器中断结构

提供八个 PWM 中断，PWM0_INT~PWM7_INT，对高级中断控制器 (AIC) 来说，这些中断被分成 PWMA_INT 和 PWMB_INT。PWM 0 和 捕捉器 0 共用一个中断，PWM1 和 捕捉器 1 共用一个中断，依此类推。因此，在同一个通道 PWM 功能和捕捉功能不能同时发生。图 5-47 和图 5-48 描述

文件发布时间：2014年7月3日

了 PWM-定时器中断的结构。

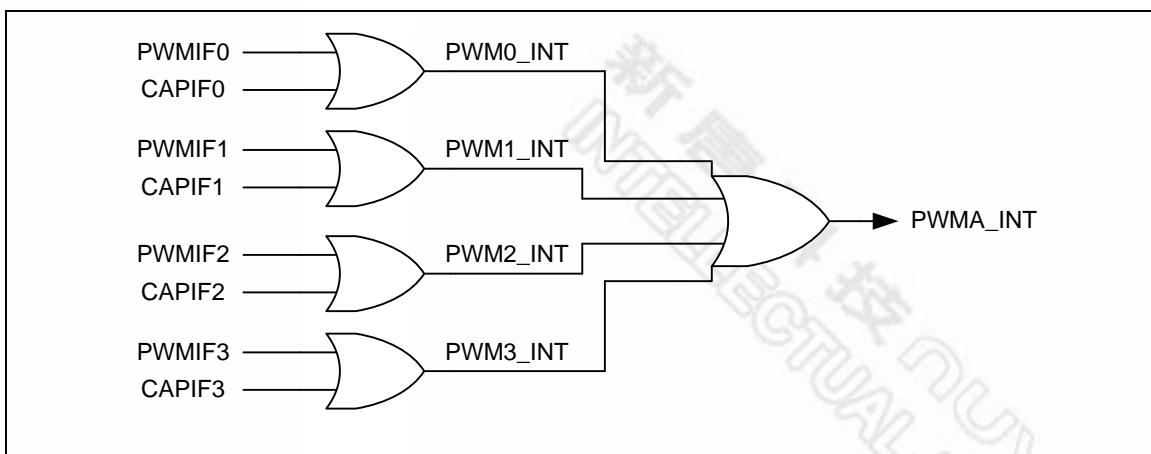


图 5-47 PWM A 组 PWM-定时器中断结构图

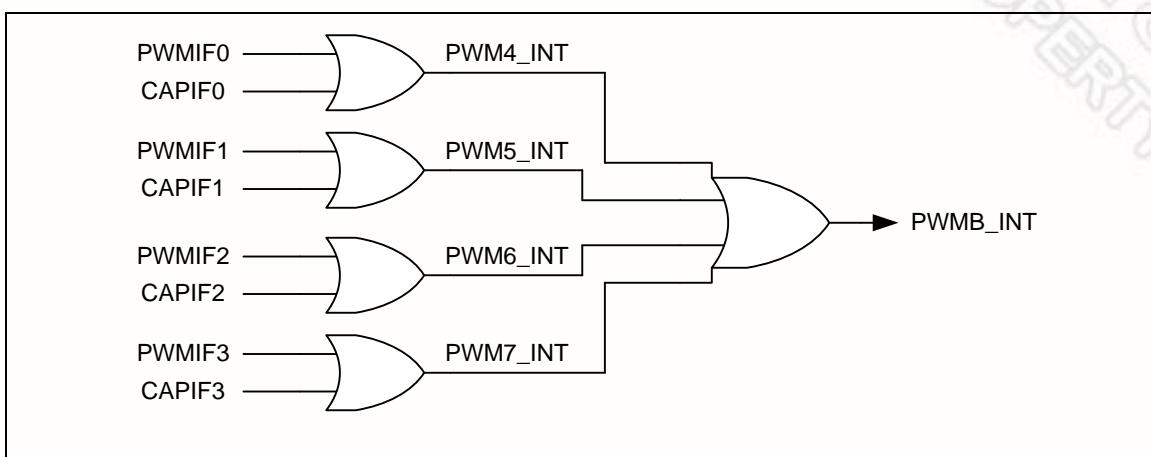


图 5-48 PWM B 组 PWM-定时器中断结构图



5.7.4.7 PWM-定时器开启过程

推荐使用下列步骤用于启动 PWM。

1. 设置时钟选择 (CSR)
2. 设置预分频器 (PPR)
3. 设置反转打开/关闭, 死区发生器打开/关闭, 自动重载/单触发模式以及停止 PWM-定时器 (PCR)
4. 设置比较器寄存器 (CMR) 来设定 PWM 占空比
5. 设置 PWM 向下计数计数器寄存器 (CNR) 来设定 PWM 周期。
6. 设置中断使能寄存器 (PIER)
7. 设置相应 GPIO 管脚为 PWM 功能 (使能 POE 并禁用 CAPENR) 用于 PWM 通道。
8. 使能 PWM 定时器开始运行 (设置 PCR 中的 CHxEN = 1)

5.7.4.8 PWM-定时器关闭过程

方式 1:

设置 16-位向下计数计数器 (CNR) 为 0, 并查看 PDR (16-位向下计数计数器的当前值)。当 PDR 达到 0, 禁用 PWM-定时器 (PCR 的 CHxEN 位)。**(推荐)**

方式 2:

设置 16-位向下计数计数器 (CNR) 为 0。当中断产生, 禁用 PWM-定时器 (PCR 的 CHxEN 位)。**(推荐)**

方式 3:

直接禁用 PWM-定时器 (PCR 的 CHxEN 位)。**(不推荐)**

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



5.7.4.9 捕捉开始过程

1. 设置时钟选择 (CSR)
2. 设置预分频器 (PPR)
3. 设置通道使能, 上升/下降沿中断使能以及输入信号反转打开/关闭 (CCR0, CCR2)
4. 设置 PWM 向下计数计数器 (CNR)
5. 设置相应的 GPIO 管脚为捕捉功能 (禁用POE 并使能 CAPENR) 用于相应的 PWM 通道。
6. 使能 PWM 定时器开始运行 (设置 PCR 中的 CHxEN 为 1)

5.7.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
PWMA_BA = 0x4004_0000 (PWM group A)				
PWMB_BA = 0x4014_0000 (PWM group B)				
PPR	PWMA_BA+0x00	R/W	PWM A组预分频寄存器	0x0000_0000
	PWMB_BA+0x00	R/W	PWM B组预分频寄存器	0x0000_0000
CSR	PWMA_BA+0x04	R/W	PWM A组时钟选择寄存器	0x0000_0000
	PWMB_BA+0x04	R/W	PWM B组时钟选择寄存器	0x0000_0000
PCR	PWMA_BA+0x08	R/W	PWM A组控制寄存器	0x0000_0000
	PWMB_BA+0x08	R/W	PWM B组控制寄存器	0x0000_0000
CNR0	PWMA_BA+0x0C	R/W	PWM A组计数寄存器0	0x0000_0000
	PWMB_BA+0x0C	R/W	PWM B组计数寄存器0	0x0000_0000
CMR0	PWMA_BA+0x10	R/W	PWM A组比较寄存器0	0x0000_0000
	PWMB_BA+0x10	R/W	PWM B组比较寄存器0	0x0000_0000
PDR0	PWMA_BA+0x14	R	PWM A组数据寄存器0	0x0000_0000
	PWMB_BA+0x14	R	PWM B组数据寄存器0	0x0000_0000
CNR1	PWMA_BA+0x18	R/W	PWM A组计数寄存器1	0x0000_0000
	PWMB_BA+0x18	R/W	PWM B组计数寄存器1	0x0000_0000
CMR1	PWMA_BA+0x1C	R/W	PWM A组比较寄存器1	0x0000_0000
	PWMB_BA+0x1C	R/W	PWM B组比较寄存器1	0x0000_0000
PDR1	PWMA_BA+0x20	R	PWM A组数据寄存器1	0x0000_0000
	PWMB_BA+0x20	R	PWM B组数据寄存器1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWM A组计数寄存器2	0x0000_0000
	PWMB_BA+0x24	R/W	PWM B组计数寄存器2	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWM A组比较寄存器2	0x0000_0000
	PWMB_BA+0x28	R/W	PWM B组比较寄存器2	0x0000_0000
PDR2	PWMA_BA+0x2C	R	PWM A组数据寄存器2	0x0000_0000
	PWMB_BA+0x2C	R	PWM B组数据寄存器2	0x0000_0000
CNR3	PWMA_BA+0x30	R/W	PWM A组计数寄存器3	0x0000_0000
	PWMB_BA+0x30	R/W	PWM B组计数寄存器3	0x0000_0000

CMR3	PWMA_BA+0x34	R/W	PWM A组比较寄存器3	0x0000_0000
	PWMB_BA+0x34	R/W	PWM B组比较寄存器3	0x0000_0000
PDR3	PWMA_BA+0x38	R	PWM A组数据寄存器3	0x0000_0000
	PWMB_BA+0x38	R	PWM B组数据寄存器3	0x0000_0000
PBCR	PWMA_BA+0x3C	R/W	PWM A组向后兼容寄存器	0x0000_0000
	PWMB_BA+0x3C	R/W	PWM B组向后兼容寄存器	0x0000_0000
PIER	PWMA_BA+0x40	R/W	PWM A组中断时能寄存器	0x0000_0000
	PWMB_BA+0x40	R/W	PWM B组中断时能寄存器	0x0000_0000
PIIR	PWMA_BA+0x44	R/W	PWM A组中断标志寄存器	0x0000_0000
	PWMB_BA+0x44	R/W	PWM B组中断标志寄存器	0x0000_0000
CCR0	PWMA_BA+0x50	R/W	PWM Group A组捕捉控制寄存器0	0x0000_0000
	PWMB_BA+0x50	R/W	PWM Group B组捕捉控制寄存器0	0x0000_0000
CCR2	PWMA_BA+0x54	R/W	PWM A组捕捉控制寄存器2	0x0000_0000
	PWMB_BA+0x54	R/W	PWM B组捕捉控制寄存器2	0x0000_0000
CRLR0	PWMA_BA+0x58	R	PWM A组捕捉上升沿锁存寄存器 (Channel 0)	0x0000_0000
	PWMB_BA+0x58	R	PWM B组捕捉上升沿锁存寄存器 (Channel 0)	0x0000_0000
CFLR0	PWMA_BA+0x5C	R	PWM A组捕捉下降沿锁存寄存器 (Channel 0)	0x0000_0000
	PWMB_BA+0x5C	R	PWM B组捕捉下降沿锁存寄存器 (Channel 0)	0x0000_0000
CRLR1	PWMA_BA+0x60	R	PWM A组捕捉上升沿锁存寄存器 (Channel 1)	0x0000_0000
	PWMB_BA+0x60	R	PWM B组捕捉上升沿锁存寄存器 (Channel 1)	0x0000_0000
CFLR1	PWMA_BA+0x64	R	PWM A组捕捉下降沿锁存寄存器 (Channel 1)	0x0000_0000
	PWMB_BA+0x64	R	PWM B组捕捉下降沿锁存寄存器 (Channel 1)	0x0000_0000
CRLR2	PWMA_BA+0x68	R	PWM A组捕捉上升沿锁存寄存器 (Channel 2)	0x0000_0000
	PWMB_BA+0x68	R	PWM B组捕捉上升沿锁存寄存器 (Channel 2)	0x0000_0000
CFLR2	PWMA_BA+0x6C	R	PWM A组捕捉下降沿锁存寄存器 (Channel 2)	0x0000_0000
	PWMB_BA+0x6C	R	PWM B组捕捉下降沿锁存寄存器 (Channel 2)	0x0000_0000
CRLR3	PWMA_BA+0x70	R	PWM A组捕捉上升沿锁存寄存器 (Channel 3)	0x0000_0000
	PWMB_BA+0x70	R	PWM B组捕捉上升沿锁存寄存器 (Channel 3)	0x0000_0000
CFLR3	PWMA_BA+0x74	R	PWM A组捕捉下降沿锁存寄存器 (Channel 3)	0x0000_0000
	PWMB_BA+0x74	R	PWM B组捕捉下降沿锁存寄存器 (Channel 3)	0x0000_0000

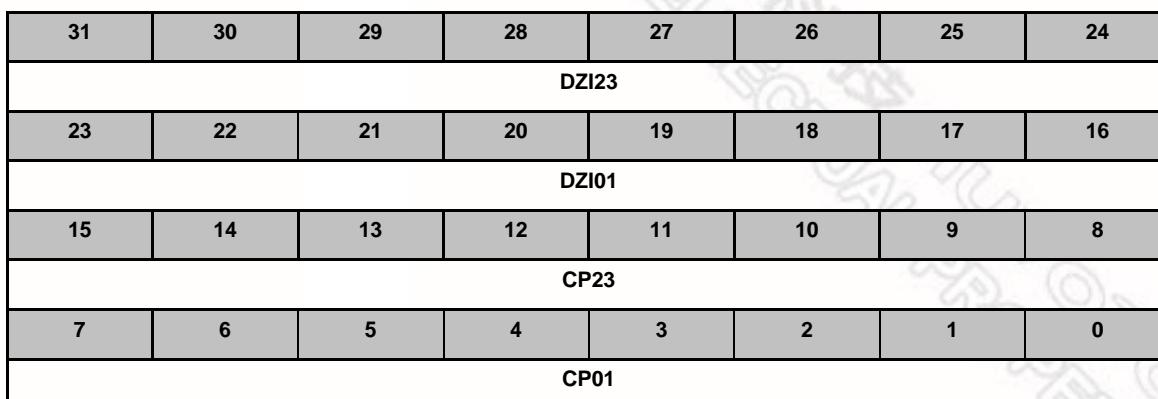
CAPENR	PWMA_BA+0x78	R/W	PWM A组捕捉输入 0~3 使能寄存器	0x0000_0000
	PWMB_BA+0x78	R/W	PWM B组捕捉输入 0~3 使能寄存器	0x0000_0000
POE	PWMA_BA+0x7C	R/W	PWM A组通道 0~3 输出使能	0x0000_0000
	PWMB_BA+0x7C	R/W	PWM B组通道 0~3 输出使能	0x0000_0000



5.7.6 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 A组预分频寄存器	0x0000_0000
	PWMB_BA+0x00	R/W	PWM B组预分频寄存器	0x0000_0000



Bits	描述
[31:24]	DZI23 通道2 和 通道3 的死区间隔 (PWM A组 PWM2 和 PWM3 配对, PWM B组 PWM6 和 PWM7 配对) 该8位寄存器决定死区长度。 单位死区时间长度由相关 CSR 位决定。
[23:16]	DZI01 通道0 和 通道1 的死区间隔 (PWM A组 PWM0 和 PWM1 配对, PWM B组 PWM4 和 PWM5 配对) 该8位寄存器决定死区长度。 单位死区时间长度由相关 CSR 位决定。
[15:8]	CP23 时钟预分频器 2 (PWM-定时器2 / 3 for A组, PWM-定时器 6 / 7 for B组) 在选定到相应的 PWM 定时器之前时钟输入被 (CP23 + 1) 除频。 如果 CP23 = 0, 预分频器2 输出时钟将会停止, 相应的 PWM-定时器也将停止。
[7:0]	CP01 时钟预分频器 0 (PWM-定时器0 / 1 for A组, PWM-定时器 4 / 5 for B组) 在选定到相应的 PWM 定时器之前时钟输入被 (CP01 + 1) 除频 如果 CP01 = 0, 预分频器0 输出时钟将会停止, 相应的 PWM-定时器也将停止。

PWM 时钟选择寄存器 (CSR)

寄存器	偏移量	R/W	描述	复位后的值
CSR	PWMA_BA+0x04	R/W	PWM A组时钟选择寄存器	0x0000_0000
	PWMB_BA+0x04	R/W	PWM B组时钟选择寄存器	0x0000_0000

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

Bits	描述													
[31:15]	Reserved	保留												
[14:12]	CSR3	PWM 定时器3 时钟源选择 (PWM 定时器3 for group A, PWM 定时器7 for group B) 选择 PWM 定时器的时钟输入: <table border="1" style="margin-left: 20px;"> <tr> <td>CSR3 [14:12]</td> <td>输入时钟除频</td> </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]	输入时钟除频	100	1	011	16	010	8	001	4	000	2
CSR3 [14:12]	输入时钟除频													
100	1													
011	16													
010	8													
001	4													
000	2													
[11]	Reserved	保留												
[10:8]	CSR2	PWM 定时器2 时钟源选择 (PWM 定时器2 for group A, PWM 定时器6 for group B) 选择 PWM 定时器的时钟输入。 (表与 CSR3 相同)												
[7]	Reserved	保留												
[6:4]	CSR1	PWM 定时器1 时钟源选择 (PWM 定时器1 for group A, PWM 定时器5 for group B) 选择 PWM 定时器的时钟输入。 (表与 CSR3 相同)												
[3]	Reserved	保留												



[2:0]	CSR0	PWM 定时器0 时钟源选择 (PWM 定时器0 for group A, PWM 定时器4 for group B) 选择 PWM 定时器的时钟输入。 (表与 CSR3 相同)
-------	-------------	--

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 A组控制寄存器 (PCR)	0x0000_0000
	PWMB_BA+0x08	R/W	PWM B组控制寄存器 (PCR)	0x0000_0000

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

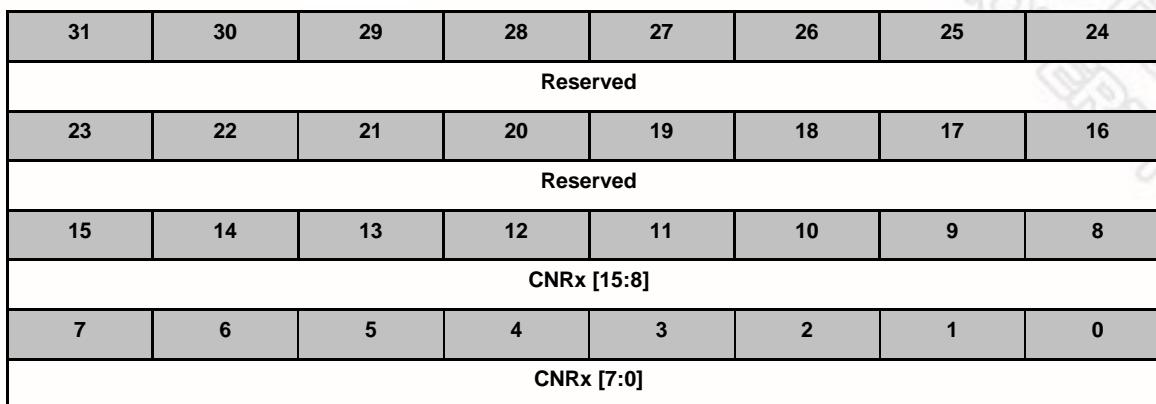
Bits	描述	
[31:28]	Reserved	保留
[27]	CH3MOD	<p>PWM-定时器 3 自动重载/单触发模式 (PWM 定时器3 for group A, PWM 定时器7 for group B) 1 = 自动重载模式 0 = 单触发模式 注：如果该位有状态转换，将会使 CNR3 和 CMR3 清位。</p>
[26]	CH3INV	<p>PWM-定时器 3 输出反向使能 (PWM 定时器3 for group A, PWM 定时器7 for group B) 1 = 反向使能 0 = 反向禁用</p>
[25]	Reserved	保留
[24]	CH3EN	<p>PWM-定时器 3 使能 (PWM 定时器3 for group A, PWM 定时器7 for group B) 1 = 使能相应的 PWM-定时器开始运行 0 = 停止相应的 PWM-定时器运行</p>
[23:20]	Reserved	保留
[19]	CH2MOD	<p>PWM-定时器 2 自动重载/单触发模式 (PWM定时器2 for group A, PWM定时器6 for group B) 1 = 自动重载模式 0 = 单触发模式 注：如果该位有状态转换，将会使 CNR2 和 CMR2 清位。</p>

[18]	CH2INV	PWM-定时器 2 输出反向使能 (PWM 定时器2 for group A, PWM 定时器6 for group B) 1 = 反向使能 0 = 反向禁用
[17]	Reserved	保留
[16]	CH2EN	PWM-定时器 2 使能 (PWM 定时器 2 for group A, PWM 定时器 6 for group B) 1 = 使能相应的 PWM-定时器开始运行 0 = 停止相应的 PWM-定时器运行
[15:12]	Reserved	保留
[11]	CH1MOD	PWM-定时器 1 自动重载/单触发模式 (PWM 定时器 1 for group A, PWM 定时器 5 for group B) 1 = 自动重载模式 0 = 单触发模式 注：如果该位有状态转换，将会使 CNR1 和 CMR1 清位。
[10]	CH1INV	PWM-Timer 1 输出反向使能 (PWM 定时器 1 for group A, PWM 定时器 5 for group B) 1 = 反向使能 0 = 反向禁用
[9]	Reserved	保留
[8]	CH1EN	PWM-定时器 1 使能 (PWM 定时器 1 for group A, PWM 定时器 5 for group B) 1 = 使能相应的 PWM-定时器开始运行 0 = 停止相应的 PWM-定时器运行
[7:6]	Reserved	保留
[5]	DZEN23	死区 2 发生器使能 (PWM A组 PWM2 和 PWM3 配对, PWM B组 PWM6 和 PWM7 配对) 1 = 使能 0 = 禁用 注：当死区发生器使能，PWM A组的 PWM2 和 PWM3 将成为互补的一对配对，， PWM B组的 PWM6 和 PWM7 将成为互补的一对配对。
[4]	DZEN01	死区 0 发生器使能 (PWM A组 PWM0 和 PWM1 配对, PWM B组 PWM4 和 PWM5 配对) 1 = 使能 0 = 禁用 注：当死区发生器使能，PWM A组的 PWM0 和 PWM1 将成为互补的一对配对，， PWM B组的 PWM4 和 PWM5 将成为互补的一对配对。
[3]	CH0MOD	PWM-Timer 0 Auto-reload/One-Shot Mode (PWM 定时器 0 for group A, PWM 定时器 4 for group B) 1 = 自动重载模式 0 = 单触发模式

		注：如果该位有状态转换，将会使 CNR0 和 CMR0 清位。
[2]	CH0INV	PWM-定时器 0 输出反向使能 (PWM定时器 0 for group A, PWM 定时器 4 for group B) 1 = 反向使能 0 = 反向禁用
[1]	Reserved	保留
[0]	CH0EN	PWM-定时器 0 使能 (PWM 定时器 0 for group A, PWM 定时器 4 for group B) 1 = 使能相应的 PWM-定时器开始运行 0 = 停止相应的 PWM-定时器运行

PWM 计数寄存器 3-0 (CNR3-0)

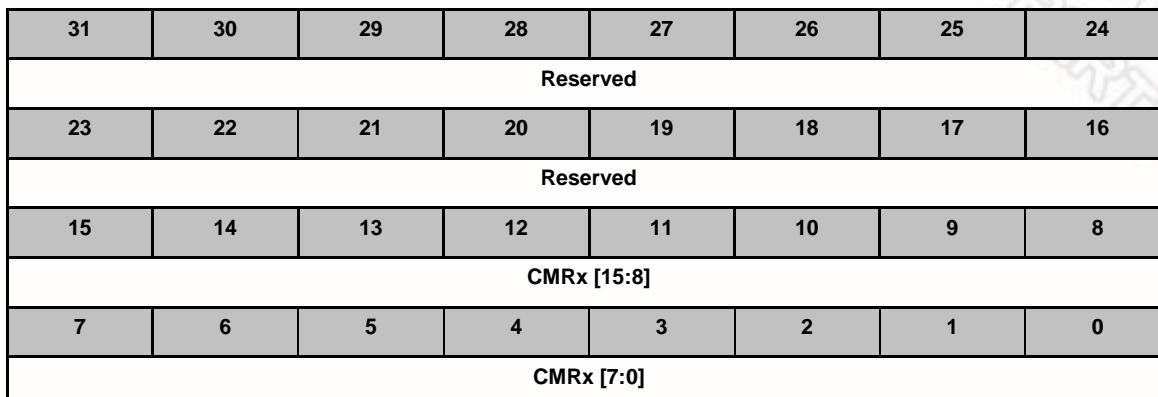
寄存器	偏移量	R/W	描述	复位后的值
CNR0	PWMA_BA+0x0C	R/W	PWM A组计数寄存器0	0x0000_0000
	PWMB_BA+0x0C	R/W	PWM B组计数寄存器0	0x0000_0000
CNR1	PWMA_BA+0x18	R/W	PWM A组计数寄存器1	0x0000_0000
	PWMB_BA+0x18	R/W	PWM B组计数寄存器1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWM A组计数寄存器2	0x0000_0000
	PWMB_BA+0x24	R/W	PWM B组计数寄存器2	0x0000_0000
CNR3	PWMA_BA+0x30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



Bits	描述	
[31:16]	Reserved	保留
[15:0]	CNRx	<p>PWM 定时器载入值 CNR 决定 PWM 的周期。</p> <ul style="list-style-type: none"> ● PWM 频率 = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; xy 代表 01, 23, 45 或 67, 取决于所选的 PWM 通道。 ● 占空比 = (CMR+1)/(CNR+1). ● CMR >= CNR: PWM 输出高. ● CMR < CNR: PWM 低脉宽 = (CNR-CMR) unit; PWM 高脉宽 = (CMR+1) unit. ● CMR = 0: PWM 低脉宽 = (CNR) unit; PWM 高脉宽 = 1 unit (Unit = 一个 PWM 时钟周期) <p>注: CNR 写入数据后将会在下一个 PWM 周期生效。</p>

PWM 比较寄存器 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 A组比较寄存器0	0x0000_0000
	PWMB_BA+0x10	R/W	PWM B组比较寄存器0	0x0000_0000
CMR1	PWMA_BA+0x1C	R/W	PWM A组比较寄存器1	0x0000_0000
	PWMB_BA+0x1C	R/W	PWM B组比较寄存器1	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWM A组比较寄存器2	0x0000_0000
	PWMB_BA+0x28	R/W	PWM B组比较寄存器2	0x0000_0000
CMR3	PWMA_BA+0x34	R/W	PWM A组比较寄存器3	0x0000_0000
	PWMB_BA+0x34	R/W	PWM B组比较寄存器3	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留
[15:0]	CMRx	<p>PWM 比较寄存器</p> <p>CMR 决定 PWM 的占空比。</p> <ul style="list-style-type: none"> ● 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 ● 占空比 = (CMR+1)/(CNR+1). ● CMR >= CNR: PWM 输出为高 ● CMR < CNR: PWM 低脉宽 = (CNR-CMR) unit; PWM 高脉宽 = (CMR+1) unit. ● CMR = 0: PWM 低脉宽 = (CNR) unit; PWM 高脉宽 = 1 unit (Unit = 一个 PWM 时钟周期) <p>注: CMR 写入数据后将会在下一个 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 A组数据寄存器0	0x0000_0000
	PWMB_BA0+0x14	R	PWM B组数据寄存器0	0x0000_0000
PDR1	PWMA_BA0+0x20	R	PWM A组数据寄存器1	0x0000_0000
	PWMB_BA0+0x20	R	PWM B组数据寄存器1	0x0000_0000
PDR2	PWMA_BA0+0x2C	R	PWM A组数据寄存器2	0x0000_0000
	PWMB_BA0+0x2C	R	PWM B组数据寄存器2	0x0000_0000
PDR3	PWMA_BA0+0x38	R	PWM A组数据寄存器3	0x0000_0000
	PWMB_BA0+0x38	R	PWM B组数据寄存器3	0x0000_0000

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

Bits	描述	
[31:16]	Reserved	保留
[15:0]	PDRx	PWM 数据寄存器 用户可以监测 PDR 来获取16-位向下计数计数器的当前值。

PWM 向后兼容寄存器

寄存器	偏移量	R/W	描述	复位后的值
PBCR	PWMA_BA+0x3C	R/W	PWM A组向后兼容寄存器	0x0000_0000
	PWMB_BA+0x3C	R/W	PWM B组向后兼容寄存器	0x0000_0000

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

Bits	描述	
[31:1]	Reserved	保留
[0]	BCn	<p>PWM 向后兼容寄存器</p> <p>0 = 配置写 0 清 CFLRI0~3 和 CRLRI0~3。</p> <p>1 = 配置写 1 清 CFLRI0~3 和 CRLRI0~3。</p> <p>请参考 CCR0/CCR2 寄存器第 6, 7, 22, 23 位的描述。</p>

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 A组中断使能寄存器	0x0000_0000
	PWMB_BA+0x40	R/W	PWM B组中断使能寄存器	0x0000_0000

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

Bits	描述	
[31:4]	Reserved	保留
[3]	PWMIE3	PWM 通道 3 中断使能 1 = 使能 0 = 禁用
[2]	PWMIE2	PWM 通道 2 中断使能 1 = 使能 0 = 禁用
[1]	PWMIE1	PWM 通道 1 中断使能 1 = 使能 0 = 禁用
[0]	PWMIE0	PWM 通道 0 中断使能 1 = 使能 0 = 禁用

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	R/W	PWM B组中断标志寄存器	0x0000_0000

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

Bits	描述	
[31:4]	Reserved	保留
[3]	PWMIF3	PWM 通道 3 中断状态 当 PWM 3 向下计数到 0，硬件将该位置 1。软件写 1 清除该位为0。
[2]	PWMIF2	PWM 通道 2 中断状态 当 PWM 2 向下计数到 0，硬件将该位置 1。软件写 1 清除该位为0。
[1]	PWMIF1	PWM 通道 1 中断状态 当 PWM 1 向下计数到 0，硬件将该位置 1。软件写 1 清除该位为0。
[0]	PWMIFO	PWM 通道 0 中断状态 当 PWM 0 向下计数到 0，硬件将该位置 1。软件写 1 清除该位为0。

注：用户可以通过向 PIIR 的相应位写 1 来清除各自的中断标志。

捕捉控制寄存器 (CCR0)

寄存器	偏移量	R/W	描述	复位后的值
CCR0	PWMA_BA+0x50	R/W	PWM A组捕捉控制寄存器	0x0000_0000
	PWMB_BA+0x50	R/W	PWM B组捕捉控制寄存器	0x0000_0000

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

Bits	描述	
[31:24]	Reserved	保留
[23]	CFLRI1	CFLR1 锁定标志位 当 PWM 组输入通道 1 发生下降沿变化时, CFLR1 锁定 PWM 向下计数器的值, 硬件设置该位。 如果 BCn 位为 0, 软件写 0 清该位; 如果 BCn 位为 1, 软件写 1 清该位。
[22]	CRLRI1	CRLR1 锁定标志位 当 PWM 组输入通道 1 发生上升沿变化时, CRLR1 锁定 PWM 向下计数器的值, 硬件设置该位。 如果 BCn 位为 0, 软件写 0 清该位; 如果 BCn 位为 1, 软件写 1 清该位。
[5]	Reserved	保留
[20]	CAPIF1	通道 1 捕捉中断标志位 如果 PWM 组通道1 上升沿锁定中断使能 (CRL_IE1=1), PWM 通道 1 发生上升沿转变会使得 CAPIF1 为高; 同样, 如果 PWM 组通道1 下降沿锁定中断使能 (CFL_IE1=1), 下降沿转变会使得 CAPIF1 为高。 写 1 清除该位为 0。
[19]	CAPCH1EN	通道 1 捕捉功能使能 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 当使能时, 捕捉功能将锁定 PWM 计数器, 并存储到 CRLR (上升锁存) 和 CFLR (下降锁存)。 当禁用时, 捕捉器不更新 CRLR 和 CFLR, 并禁用 PWM 组通道 1 中断。

[18]	CFL_IE1	通道 1 下降锁定中断使能 1 = 使能下降沿锁存中断 0 = 禁用下降沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 1 有下降沿转变，捕捉产生中断。
[17]	CRL_IE1	通道 1 上升锁定中断使能 1 = 使能上升沿锁存中断 0 = 禁用上升沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 1 有上升沿转变，捕捉产生中断。
[16]	INV1	通道 1 反向使能 1 = 反向使能。将 GPIO 的输入信号在用于捕捉定时器之前反向 0 = 反向禁用
[15:8]	Reserved	保留
[7]	CFLR0	CFLR0 锁定标志位 当 PWM 组输入通道 0 发生下降沿变化时，CFLR0 锁定 PWM 向下计数器的值，硬件设置该位。 如果 BCn 位为 0，软件写 0 清该位；如果 BCn 位为 1，软件写 1 清该位。
[6]	CRLR0	CRLR0 锁定标志位 当 PWM 组输入通道 0 发生上升沿变化时，CRLR0 锁定 PWM 向下计数器的值，硬件设置该位。 如果 BCn 位为 0，软件写 0 清该位；如果 BCn 位为 1，软件写 1 清该位。
[5]	Reserved	保留
[4]	CAPIF0	通道 0 捕捉中断标志位 如果 PWM 组通道 0 上升沿锁定中断使能 (CRL_IE0=1)，PWM 通道 0 发生上升沿转变会使得 CAPIF0 为高；同样，如果 PWM 组通道 0 下降沿锁定中断使能 (CFL_IE0=1)，下降沿转变会使得 CAPIF0 为高。 写 1 清除该位为 0。
[3]	CAPCH0EN	通道 0 捕捉功能使能 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。 当使能时，捕捉功能将锁定 PWM 计数器，并存储到 CRLR（上升锁存）和 CFLR（下降锁存）。 当禁用时，捕捉器不更新 CRLR 和 CFLR，并禁用 PWM 组通道 0 中断。
[2]	CFL_IE0	通道 0 下降锁定中断使能 1 = 使能下降沿锁存中断 0 = 禁用下降沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 0 有下降沿转变，捕捉产生中断。
[1]	CRL_IE0	通道 0 上升锁定中断使能 1 = 使能上升沿锁存中断

		0 = 禁用上升沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 0 有上升沿转变，捕捉产生中断。
[0]	INV0	通道 0 反向使能 1 = 反向使能。将 GPIO 的输入信号在用于捕捉定时器之前反向。 0 = 反向禁用

捕捉控制寄存器 (CCR2)

寄存器	偏移量	R/W	描述	复位后的值
CCR2	PWMA_BA+0x54	R/W	PWM A组捕捉控制寄存器	0x0000_0000
	PWMB_BA+0x54	R/W	PWM B组捕捉控制寄存器	0x0000_0000

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

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

[18]	CFL_IE3	通道 3 下降锁定中断使能 1 = 使能下降沿锁存中断 0 = 禁用下降沿锁存中断 当使能时, 如果捕捉器检测到 PWM 组通道 3 有下降沿转变, 捕捉产生中断
[17]	CRL_IE3	通道 3 上升锁定中断使能 1 = 使能上升沿锁存中断 0 = 禁用上升沿锁存中断 当使能时, 如果捕捉器检测到 PWM 组通道 3 有上升沿转变, 捕捉产生中断。
[16]	INV3	通道 3 反向使能 1 = 反向使能。将 GPIO 的输入信号在用于捕捉定时器之前反向。 0 = 反向禁用
[15:8]	Reserved	保留
[7]	CFLRI2	CFLR2 锁定标志位 当 PWM 组输入通道 2 发生下降沿变化时, CFLR2 锁定 PWM 向下计数器的值, 硬件设置该位。 如果 BCn 位为 0, 软件写 0 清该位; 如果 BCn 位为 1, 软件写 1 清该位。
[6]	CRLRI2	CRLR2 锁定标志位 当 PWM 组输入通道 2 发生上升沿变化时, CRLR2 锁定 PWM 向下计数器的值, 硬件设置该位。 如果 BCn 位为 0, 软件写 0 清该位; 如果 BCn 位为 1, 软件写 1 清该位。
[5]	Reserved	保留
[4]	CAPIF2	通道 2 捕捉中断标志位 如果 PWM 组通道2 上升沿锁定中断使能 (CRL_IE2=1), PWM 通道 2 发生上升沿转变会使得 CAPIF2 为高; 同样, 如果 PWM 组通道2 下降沿锁定中断使能 (CFLIE2=1), 下降沿转变会使得 CAPIF2 为高。 写 1 清除该位为 0。
[3]	CAPCH2EN	通道 2 捕捉功能使能 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 当使能时, 捕捉功能将锁定 PWM 计数器, 并存储到 CRLR (上升锁存) 和 CFLR (下降锁存)。 当禁用时, 捕捉器不更新 CRLR 和 CFLR, 并禁用 PWM 组通道 2 中断。
[2]	CFLIE2	通道 2 下降锁定中断使能 1 = 使能下降沿锁存中断 0 = 禁用下降沿锁存中断 当使能时, 如果捕捉器检测到 PWM 组通道 2 有下降沿转变, 捕捉产生中断
[1]	CRLIE2	通道 2 上升锁定中断使能 1 = 使能上升沿锁存中断

		0 = 禁用上升沿锁存中断 当使能时，如果捕捉器检测到 PWM 组通道 2 有上升沿转变，捕捉产生中断。
[0]	INV2	通道 2 反向使能 1 = 反向使能。将 GPIO 的输入信号在用于捕捉定时器之前反向。 0 = 反向禁用

捕捉上升沿锁存寄存器 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 A组捕捉上升沿锁存寄存器（通道 0）	0x0000_0000
	PWMB_BA+0x58	R	PWM B组捕捉上升沿锁存寄存器（通道 0）	0x0000_0000
CRLR1	PWMA_BA+0x60	R	PWM A组捕捉上升沿锁存寄存器（通道 1）	0x0000_0000
	PWMB_BA+0x60	R	PWM B组捕捉上升沿锁存寄存器（通道 1）	0x0000_0000
CRLR2	PWMA_BA+0x68	R	PWM A组捕捉上升沿锁存寄存器（通道 2）	0x0000_0000
	PWMB_BA+0x68	R	PWM B组捕捉上升沿锁存寄存器（通道 2）	0x0000_0000
CRLR3	PWMA_BA+0x70	R	PWM A组捕捉上升沿锁存寄存器（通道 3）	0x0000_0000
	PWMB_BA+0x70	R	PWM B组捕捉上升沿锁存寄存器（通道 3）	0x0000_0000

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

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

捕捉下降沿锁存寄存器 3-0 (CFLR3-0)

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

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

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

捕捉输入使能寄存器 (CAPENR)

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

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

Bits	描述	
[3:0]	CAPENR	<p>捕捉输入使能寄存器</p> <p>有4组捕捉输入。Bit0~Bit3 用于控制每个输入的使能或关闭。</p> <p>0 = 禁用 (PWMx 多功能管脚输入不会影响输入捕捉功能。)</p> <p>1 = 使能 (PWMx 多功能管脚输入将会影响输入捕捉功能。)</p> <p>CAPENR</p> <p><u>Bit 3210 for PWM A组</u></p> <p>Bit xxx1 → 捕捉通道 0 从 PA.12 输入</p> <p>Bit xx1x → 捕捉通道 1 从 PA.13 输入</p> <p>Bit x1xx → 捕捉通道 2 从 PA.14 输入</p> <p>Bit 1xxx → 捕捉通道 3 从 PA.15 输入</p> <p><u>Bit 3210 for PWM B组</u></p> <p>Bit xxx1 → 捕捉通道 0 从 PB.11 输入</p> <p>Bit xx1x → 捕捉通道 1 从 PE.5 输入</p> <p>Bit x1xx → 捕捉通道 2 从 PE.0 输入</p> <p>Bit 1xxx → 捕捉通道 3 从 PE.1 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 A组通道 0~3 输出使能寄存器	0x0000_0000
	PWMB_BA+0x7C	R/W	PWM B组通道 0~3 输出使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				PWM3	PWM2	PWM1	PWM0

Bits	描述	
[3]	PWM3	通道 3 输出使能寄存器 1 = 使能 PWM 通道 3 输出 0 = 禁用 PWM 通道 3 输出 注：相应的 GPIO 管脚必须切换到 PWM 功能
[2]	PWM2	通道 2 输出使能寄存器 1 = 使能 PWM 通道 2 输出 0 = 禁用 PWM 通道 2 输出 注：相应的 GPIO 管脚必须切换到 PWM 功能
[1]	PWM1	通道 1 输出使能寄存器 1 = 使能 PWM 通道 1 输出 0 = 禁用 PWM 通道 1 输出 注：相应的 GPIO 管脚必须切换到 PWM 功能
[0]	PWM0	通道 0 输出使能寄存器 1 = 使能 PWM 通道 0 输出 0 = 禁用 PWM 通道 0 输出 注：相应的 GPIO 管脚必须切换到 PWM 功能



5.8 实时时钟 (RTC)

5.8.1 概述

实时时钟 (RTC) 控制器用于记录实时时间及日历功能。RTC 的时钟源由外部 32.768 kHz 晶振提供，管脚为 X32I 和 X32O（请参考管脚描述）或者管脚 X32I 外接 32.768 kHz 振荡器输出信号源。RTC 控制器提供时间信息（秒、分、时）在时间载入寄存器 (TLR) 以及通过日历载入寄存器 (CLR) 提供日历信息（日、月、年）。时间信息由 BCD 码格式进行表示。该控制器也提供闹钟功能，用户可以预先在时间闹钟寄存器 (TAR) 中设置闹钟时间、日历闹钟寄存器 (CAR) 中设置闹钟日期来进行闹铃设置。

RTC 控制器支持周期时间节拍和闹钟匹配中断。通过设定 TTR (TTR[2:0])，周期中断有 8 个周期选项 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 及 1 秒。当闹钟中断使能 (RIER.AIER=1) 的情况下，RTC 计数器内的值 TLR 和 CLR 分别等于闹钟设定时间寄存器 TAR 和 CAR 时，中断标志 (RIIR.AIF) 将被置，并产生闹钟中断请求。如果唤醒芯片功能被使能 (TWKE (TTR[3])=1)，那么 RTC 时间与闹钟匹配时能够将芯片从掉电模式中唤醒。

5.8.2 特征

- 支持时间计数（秒，分，时）和日历计数（日，月，年），用户可以用来查看时间
- 闹钟寄存器（秒，分，时，日，月，年）
- 12-小时或 24-小时模式可选择
- 闰年自动识别
- 一周天数计数器
- 频率补偿寄存器 (FCR)
- 所有时间日期由 BCD 码表示
- 支持周期时间节拍中断，提供 8 个周期选项供选择 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 及 1 秒
- 支持 RTC 定时节拍和闹钟定时中断
- 支持从掉电模式下唤醒芯片

5.8.3 框图

RTC 模块框图如下：

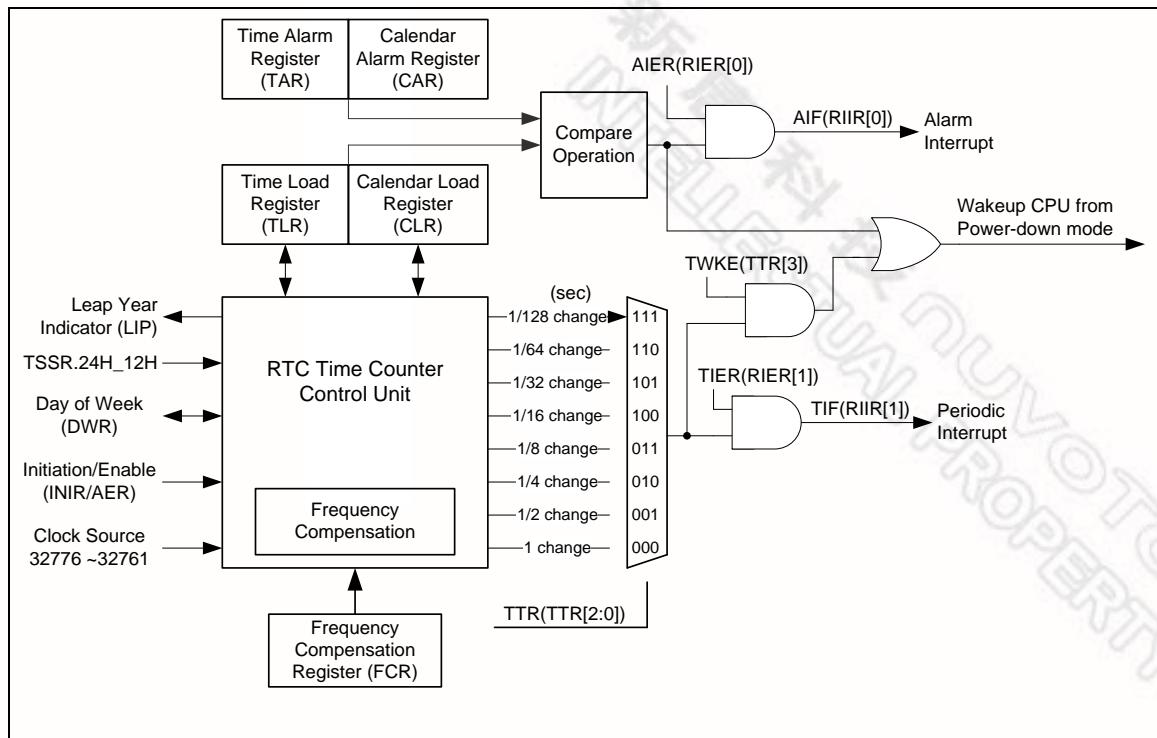


图 5-49 RTC 框图

5.8.4 功能描述

5.8.4.1 访问 RTC 寄存器

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

此外，用户必须保证RTC控制器载入数据的合理性。RTC不会对DWR和CLR内的合理性进行检查。

5.8.4.2 RTC 初始化

当RTC模块上电后，RTC处于reset状态。用户需要写入数据(0xa5eb1357)到INIR让RTC离开reset状态。一旦INIR被写入0xa5eb1357，RTC将会永远处于非复位状态。

5.8.4.3 RTC 读/写使能

寄存器AER位15~0作为保护RTC寄存器允许其读/写的密码。AER位15~0必须被设置为0xA965来使能访问的限制。一旦该部分被设置，将会至少有效1024个RTC时钟(大约30ms)。用户可以通过读取RTC使能标志位AER.ENF来确认RTC是否开始工作。

5.8.4.4 频率补偿

RTC FCR允许软件对时钟输入进行数字补偿。时钟输入的频率必须在32776 Hz到32761 Hz范围内。用户可以通过GPIO测量RTC时钟的频率，并存储在Flash中，当上电时用于数据检索。下列为例对时钟输入进行补偿的示例。

例1:

频率计数测量: 32773.65 Hz (> 32768 Hz)

整数部分: 32773 => 0x8005

FCR.Integer = 0x05 - 0x01 + 0x08 = 0x0c

分数部分: 0.65 x 60 = 39 => 0x27

FCR.Fraction = 0x27

例2

频率计数测量: 32765.27 Hz (\leq 32768 Hz)

整数部分: 32765 => 0x7FFD

FCR.Integer = 0x0A - 0x01 - 0x08 = 0x04

分数部分: 0.27 x 60 = 16.2 => 0x10

FCR.Fraction = 0x10

5.8.4.5 时间和日历计数

TLR和CLR用于载入时间和日历。TAR和CAR用于闹钟。由BCD格式呈现。

5.8.4.6 12/24小时时标选择

根据TSSR位0选择12/24小时时标格式。

5.8.4.7 一周日期计数器

RTC控制器提供一周计日格式寄存器DWR。内部的值由0至6，用于表示周日至周六。

5.8.4.8 时间周期中断

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

5.8.4.9 时间闹钟中断

当 TLR 和 CLR 中的 RTC 计数器等于 TAR 和 CAR 中的设定值，如果此时闹钟中断已设为使能 (RIER.AIERN=1)，则闹钟中断标志 (RIIR.AIF) 被置位同时发出闹钟中断请求。

5.8.4.10 应用指南：

1. TAR, CAR, TLR and CLR 寄存器为 BCD 格式。
2. 程序设计者必须保证载入值为有效合理的。例如，CLR = 201a (年), 13 (月), 00 (日)，或 CLR 与 DWR 不匹配，等等。
3. 复位状态：

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

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

5.8.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
RTC_BA = 0x4000_8000				
INIR	RTC_BA+0x00	R/W	RTC 初始寄存器	0x0000_0000
AER	RTC_BA+0x04	R/W	RTC 访问使能寄存器	0x0000_0000
FCR	RTC_BA+0x08	R/W	RTC 频率补偿寄存器	0x0000_0700
TLR	RTC_BA+0x0C	R/W	时间载入寄存器	0x0000_0000
CLR	RTC_BA+0x10	R/W	日历载入寄存器	0x0005_0101
TSSR	RTC_BA+0x14	R/W	时间格式（时标）选择寄存器	0x0000_0001
DWR	RTC_BA+0x18	R/W	一周日期寄存器	0x0000_0006
TAR	RTC_BA+0x1C	R/W	时间闹钟寄存器	0x0000_0000
CAR	RTC_BA+0x20	R/W	日历闹钟寄存器	0x0000_0000
LIR	RTC_BA+0x24	R	闰年指示寄存器	0x0000_0000
RIER	RTC_BA+0x28	R/W	RTC 中断使能寄存器	0x0000_0000
RIIR	RTC_BA+0x2C	R/W	RTC 中断指示寄存器	0x0000_0000
TTR	RTC_BA+0x30	R/W	RTC 时钟节拍寄存器	0x0000_0000



5.8.6 寄存器描述

RTC 初始寄存器 (INIR)

寄存器	偏移量	R/W	描述	复位后的值
INIR	RTC_BA+0x00	R/W	RTC 初始寄存器	0x0000_0000

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

Bits	描述	
[31:0]	INIR	RTC 初始化Initiation 当 RTC 模块上电后，RTC 处于复位状态。用户需要写入数据 (0xa5eb1357) 到 INIR 使得 RTC 离开复位状态。一旦 INIR 被写入 0xa5eb1357，RTC 将会一直处于非复位状态。
[0]	Active	RTC 活动状态 (只读) 0 = RTC 在复位状态 1 = RTC 正常运行

RTC 访问使能寄存器 (AER)

寄存器	偏移量	R/W	描述	复位后的值
AER	RTC_BA+0x04	R/W	RTC 访问使能寄存器	0x0000_0000

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

Bits	描述																																																										
[31:17]	Reserved		保留																																																								
[16]	ENF RTC 寄存器访问使能标志 (只读) 1 = RTC 寄存器读/写使能 0 = RTC 寄存器读/写禁用 对 AER[15:0] 载入 0xA965 后, 该位被置。1024个 RTC 时钟后, 或者当AER[15:0] 不等于 0xA965 时, 自动清除。		<table border="1"> <tr> <td>Register</td> <td>AER.ENF</td> <td>1</td> <td>0</td> </tr> <tr> <td>INIR</td> <td></td> <td>R/W</td> <td>R/W</td> </tr> <tr> <td>AER</td> <td></td> <td>R/W</td> <td>R/W</td> </tr> <tr> <td>FCR</td> <td></td> <td>R/W</td> <td>-</td> </tr> <tr> <td>TLR</td> <td></td> <td>R/W</td> <td>R</td> </tr> <tr> <td>CLR</td> <td></td> <td>R/W</td> <td>R</td> </tr> <tr> <td>TSSR</td> <td></td> <td>R/W</td> <td>R/W</td> </tr> <tr> <td>DWR</td> <td></td> <td>R/W</td> <td>R</td> </tr> <tr> <td>TAR</td> <td></td> <td>R/W</td> <td>-</td> </tr> <tr> <td>CAR</td> <td></td> <td>R/W</td> <td>-</td> </tr> <tr> <td>LIR</td> <td></td> <td>R</td> <td>R</td> </tr> <tr> <td>RIER</td> <td></td> <td>R/W</td> <td>R/W</td> </tr> <tr> <td>RIIR</td> <td></td> <td>R/W</td> <td>R/W</td> </tr> <tr> <td>TTR</td> <td></td> <td>R/W</td> <td>-</td> </tr> </table>	Register	AER.ENF	1	0	INIR		R/W	R/W	AER		R/W	R/W	FCR		R/W	-	TLR		R/W	R	CLR		R/W	R	TSSR		R/W	R/W	DWR		R/W	R	TAR		R/W	-	CAR		R/W	-	LIR		R	R	RIER		R/W	R/W	RIIR		R/W	R/W	TTR		R/W	-
Register	AER.ENF	1	0																																																								
INIR		R/W	R/W																																																								
AER		R/W	R/W																																																								
FCR		R/W	-																																																								
TLR		R/W	R																																																								
CLR		R/W	R																																																								
TSSR		R/W	R/W																																																								
DWR		R/W	R																																																								
TAR		R/W	-																																																								
CAR		R/W	-																																																								
LIR		R	R																																																								
RIER		R/W	R/W																																																								
RIIR		R/W	R/W																																																								
TTR		R/W	-																																																								



[15:0]	AER	RTC 寄存器访问使能密码（只写） 0xA965 = 使能 RTC 读写 Others = 禁用 RTC 读写
--------	-----	---

RTC 频率补偿寄存器 (FCR)

寄存器	偏移量	R/W	描述	复位后的值
FCR	RTC_BA+0x08	R/W	频率补偿寄存器	0x0000_0700

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

Bits	描述				
[31:12]	Reserved	保留			
		整数部分			
[11:8]	INTEGER	整数部分值	FCR[11:8]	整数部分值	FCR[11:8]
		32776	1111	32768	0111
		32775	1110	32767	0110
		32774	1101	32766	0101
		32773	1100	32765	0100
		32772	1011	32764	0011
		32771	1010	32763	0010
		32770	1001	32762	0001
		32769	1000	32761	0000
[5:0]	FRACTION	分数部分 公式 = (分数部分值) × 60 注：FCR 的值必须按照 Hex 格式表示。参考章节 5.8.4.4 范例。			

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

RTC 时间载入寄存器 (TLR)

寄存器	偏移量	R/W	描述	复位后的值
TLR	RTC_BA+0x0C	R/W	时间载入寄存器	0x0000_0000

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

Bits	描述	
[31:22]	Reserved	保留
[21:20]	10HR	10-Hour Time Digit (0~2)
[19:16]	1HR	1-Hour Time Digit (0~9)
[15]	Reserved	保留
[14:12]	10MIN	10-Min Time Digit (0~5)
[11:8]	1MIN	1-Min Time Digit (0~9)
[7]	Reserved	保留
[6:4]	10SEC	10-Sec Time Digit (0~5)
[3:0]	1SEC	1-Sec Time Digit (0~9)

注:

1. TLR 为 BCD 计数方式, RTC 不会对载入值进行检测。
2. 括号内列出的为可接受的值。

RTC 日历载入寄存器 (CLR)

寄存器	偏移量	R/W	描述	复位后的值
CLR	RTC_BA+0x10	R/W	日历载入寄存器	0x0005_0101

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

Bits	描述	
[31:24]	Reserved	保留
[23:20]	10YEAR	10-Year Calendar Digit (0~9)
[19:16]	1YEAR	1-Year Calendar Digit (0~9)
[15:13]	Reserved	保留
[12]	10MON	10-Month Calendar Digit (0~1)
[11:8]	1MON	1-Month Calendar Digit (0~9)
[7:6]	Reserved	保留
[5:4]	10DAY	10-Day Calendar Digit (0~3)
[3:0]	1DAY	1-Day Calendar Digit (0~9)

注:

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

RTC 时间格式选择寄存器 (TSSR)

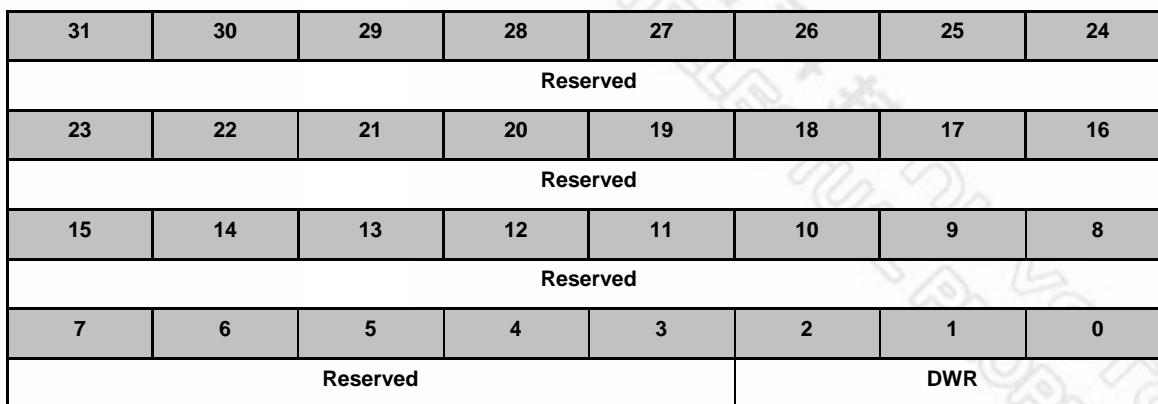
寄存器	偏移量	R/W	描述	复位后的值
TSSR	RTC_BA+0x14	R/W	时间格式选择寄存器	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							24H_12H

Bits	描述																																																							
[31:1]	Reserved	保留																																																						
[0]	24H_12H 24-小时 / 12-小时 模式选择 用于表示 TLR 和 TAR 为 24-小时 或 12-小时模式 1 = 选择 24-小时制 0 = 选择 12-小时制，带AM/PM 指示																																																							
	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>24-小时制</th> <th>12-小时制</th> <th>24-小时制</th> <th>12-小时制 (PM 时间 + 20)</th> </tr> </thead> <tbody> <tr><td>00</td><td>12(AM12)</td><td>12</td><td>32(PM12)</td></tr> <tr><td>01</td><td>01 (AM01)</td><td>13</td><td>21 (PM01)</td></tr> <tr><td>02</td><td>02(AM02)</td><td>14</td><td>22(PM02)</td></tr> <tr><td>03</td><td>03(AM03)</td><td>15</td><td>23(PM03)</td></tr> <tr><td>04</td><td>04 (AM04)</td><td>16</td><td>24 (PM04)</td></tr> <tr><td>05</td><td>05(AM05)</td><td>17</td><td>25(PM05)</td></tr> <tr><td>06</td><td>06(AM06)</td><td>18</td><td>26(PM06)</td></tr> <tr><td>07</td><td>07(AM07)</td><td>19</td><td>27(PM07)</td></tr> <tr><td>08</td><td>08(AM08)</td><td>20</td><td>28(PM08)</td></tr> <tr><td>09</td><td>09(AM09)</td><td>21</td><td>29(PM09)</td></tr> <tr><td>10</td><td>10 (AM10)</td><td>22</td><td>30 (PM10)</td></tr> <tr><td>11</td><td>11 (AM11)</td><td>23</td><td>31 (PM11)</td></tr> </tbody> </table>				24-小时制	12-小时制	24-小时制	12-小时制 (PM 时间 + 20)	00	12(AM12)	12	32(PM12)	01	01 (AM01)	13	21 (PM01)	02	02(AM02)	14	22(PM02)	03	03(AM03)	15	23(PM03)	04	04 (AM04)	16	24 (PM04)	05	05(AM05)	17	25(PM05)	06	06(AM06)	18	26(PM06)	07	07(AM07)	19	27(PM07)	08	08(AM08)	20	28(PM08)	09	09(AM09)	21	29(PM09)	10	10 (AM10)	22	30 (PM10)	11	11 (AM11)	23	31 (PM11)
24-小时制	12-小时制	24-小时制	12-小时制 (PM 时间 + 20)																																																					
00	12(AM12)	12	32(PM12)																																																					
01	01 (AM01)	13	21 (PM01)																																																					
02	02(AM02)	14	22(PM02)																																																					
03	03(AM03)	15	23(PM03)																																																					
04	04 (AM04)	16	24 (PM04)																																																					
05	05(AM05)	17	25(PM05)																																																					
06	06(AM06)	18	26(PM06)																																																					
07	07(AM07)	19	27(PM07)																																																					
08	08(AM08)	20	28(PM08)																																																					
09	09(AM09)	21	29(PM09)																																																					
10	10 (AM10)	22	30 (PM10)																																																					
11	11 (AM11)	23	31 (PM11)																																																					

RTC 一周日期寄存器 (DWR)

寄存器	偏移量	R/W	描述	复位后的值
DWR	RTC_BA+0x18	R/W	一周日期寄存器	0x0000_0006



Bits	描述																	
[31:3]	Reserved	保留																
[2:0]	DWR	<p>一周日期寄存器</p> <table border="1"> <tr> <td>值</td> <td>一周日期</td> </tr> <tr> <td>0</td> <td>星期日</td> </tr> <tr> <td>1</td> <td>星期一</td> </tr> <tr> <td>2</td> <td>星期二</td> </tr> <tr> <td>3</td> <td>星期三</td> </tr> <tr> <td>4</td> <td>星期四</td> </tr> <tr> <td>5</td> <td>星期五</td> </tr> <tr> <td>6</td> <td>星期六</td> </tr> </table>	值	一周日期	0	星期日	1	星期一	2	星期二	3	星期三	4	星期四	5	星期五	6	星期六
值	一周日期																	
0	星期日																	
1	星期一																	
2	星期二																	
3	星期三																	
4	星期四																	
5	星期五																	
6	星期六																	

RTC 时间闹钟寄存器 (TAR)

寄存器	偏移量	R/W	描述	复位后的值
TAR	RTC_BA+0x1C	R/W	时间闹钟寄存器	0x0000_0000

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

Bits	描述	
[31:22]	Reserved	保留
[21:20]	10HR	10-Hour Time Digit of Alarm Setting (0~2)
[19:16]	1HR	1-Hour Time Digit of Alarm Setting (0~9)
[15]	Reserved	保留
[14:12]	10MIN	10-Min Time Digit of Alarm Setting (0~5)
[11:8]	1MIN	1-Min Time Digit of Alarm Setting (0~9)
[7]	Reserved	保留
[6:4]	10SEC	10-Sec Time Digit of Alarm Setting (0~5)
[3:0]	1SEC	1-Sec Time Digit of Alarm Setting (0~9)

注:

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

RTC 日历闹钟寄存器 (CAR)

寄存器	偏移量	R/W	描述	复位后的值
CAR	RTC_BA+0x20	R/W	日历闹钟寄存器	0x0000_0000

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

Bits	描述	
[31:24]	Reserved	保留
[23:20]	10YEAR	10-Year Calendar Digit of Alarm Setting (0~9)
[19:16]	1YEAR	1-Year Calendar Digit of Alarm Setting (0~9)
[15:13]	Reserved	保留
[12]	10MON	10-Month Calendar Digit of Alarm Setting (0~1)
[11:8]	1MON	1-Month Calendar Digit of Alarm Setting (0~9)
[7:6]	Reserved	保留
[5:4]	10DAY	10-Day Calendar Digit of Alarm Setting (0~3)
[3:0]	1DAY	1-Day Calendar Digit of Alarm Setting (0~9)

注:

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



RTC 闰年指示寄存器 (LIR)

寄存器	偏移量	R/W	描述	复位后的值
LIR	RTC_BA+0x24	R	RTC 闰年指示寄存器	0x0000_0000

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

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

**RTC 中断使能寄存器 (RIER)**

寄存器	偏移量	R/W	描述	复位后的值
RIER	RTC_BA+0x28	R/W	RTC 中断使能寄存器	0x0000_0000

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

Bits	描述	
[31:2]	Reserved	保留
[1]	TIER	时钟节拍中断使能 1 = RTC 时钟节拍中断使能 0 = RTC 时钟节拍中断禁用
[0]	AIER	闹钟中断使能 1 = RTC 闹钟中断使能 0 = RTC 闹钟中断禁用

RTC 中断指示寄存器 (RIIR)

寄存器	偏移量	R/W	描述	复位后的值
RIIR	RTC_BA+0x2C	R/W	RTC 中断指示寄存器	0x0000_0000

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

Bits	描述	
[31:2]	Reserved	保留
[1]	TIF	<p>RTC 时钟节拍中断标志</p> <p>当 RTC 时钟节拍中断使能 (RIER.TIER=1), RTC 控制器根据 TTR[2:0] 设定的周期性使得 TIF 为高。该位通过软件写 1 清除。</p> <p>1= 如果 RIER.TIER = 1, 表示有 RTC 时钟节拍中断请求。 0= 没有 RCT 时钟节拍中断请求。</p>
[0]	AIF	<p>RTC Alarm Interrupt Flag</p> <p>当 RTC 闹钟中断使能 (RIER.AIER=1), 一旦 RTC TLR 和 CLR 的值达到了闹钟设定寄存器 TAR 和 CAR 的值, RTC 控制器将设置 AIF 为高。该位通过软件写 1 清除。</p> <p>1= 如果 RIER.AIER = 1, 表示有 RTC 闹钟中断请求 0= 没有 RTC 闹钟中断请求</p>

RTC 时钟节拍寄存器 (TTR)

寄存器	偏移量	R/W	描述	复位后的值
TTR	RTC_BA+0x30	R/W	RTC 时钟节拍寄存器	0x0000_0000

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

Bits	描述																			
[31:4]	Reserved	保留																		
[3]	TWKE	<p>RTC 定时器唤醒功能使能 如果芯片在进入掉电模式之前，TWKE 被置位，当 RTC 时钟节拍或闹钟匹配发生时，芯片会被 RTC 控制器唤醒。 1 = 使能 RTC 定时器通过时钟节拍或闹钟匹配将芯片从掉电模式唤醒的功能 0 = 禁用 RTC 定时器唤醒功能 注：时间计数根据 TTR[2:0] 的描述设定。</p>																		
[2:0]	TTR	<p>时钟节拍寄存器 设定产生时钟节拍中断的周期</p> <table border="1"> <thead> <tr> <th>TTR[2:0]</th> <th>时钟节拍 (秒)</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1/2</td> </tr> <tr> <td>2</td> <td>1/4</td> </tr> <tr> <td>3</td> <td>1/8</td> </tr> <tr> <td>4</td> <td>1/16</td> </tr> <tr> <td>5</td> <td>1/32</td> </tr> <tr> <td>6</td> <td>1/64</td> </tr> <tr> <td>7</td> <td>1/128</td> </tr> </tbody> </table> <p>注：在 RTC 访问使能位 ENF (AER[16]) 激活后，该寄存器的值可以被读回。</p>	TTR[2:0]	时钟节拍 (秒)	0	1	1	1/2	2	1/4	3	1/8	4	1/16	5	1/32	6	1/64	7	1/128
TTR[2:0]	时钟节拍 (秒)																			
0	1																			
1	1/2																			
2	1/4																			
3	1/8																			
4	1/16																			
5	1/32																			
6	1/64																			
7	1/128																			

5.9 串行外围设备接口 (SPI)

5.9.1 概述

SPI 接口是工作于全双工模式下的同步串行数据传输接口。共支持四组双向主/从模式传输。NuMicro™ NUC130/NUC140 系列包括4组 SPI 控制器，将从外设得到的数据进行串并转换，或将数据进行并串转换，发送到外设。每组 SPI 控制器可以被作为一个主机，还可以被设置为外围设备的从机。

该控制器支持不同串行时钟以适应不同应用，也支持 2 位传输模式，可同时连接两个片外从机设备，SPI 控制器也支持 PDMA 功能访问数据缓冲。

5.9.2 特征

- 最多支持四组 SPI 控制器
- 支持主机或从机模式操作
- 支持 1 位或 2 位传输模式
- 最大传输字可达32位，一次最多可传输2个字，即一次最多可传输64位数据
- 提供 burst 操作模式，在一次传输过程中，发送/接收可执行两次字传输
- 支持 MSB 或 LSB 为最先传输模式
- 主机模式下有 2 条设备/从机选择线，从机模式下有 1 条设备/从机选择线
- 支持数据寄存器字节重排序
- 支持字节或字休眠模式
- 主机模式下，支持不同输出串行时钟频率
- 主机模式下，支持两个可编程的串行时钟频率
- 支持两个通道的 PDMA 请求，一个用于发送，一个用于接收
- 支持三线，没有从机选择信号的双向接口
- SPI 时钟可以配置等于系统时钟

5.9.3 框图

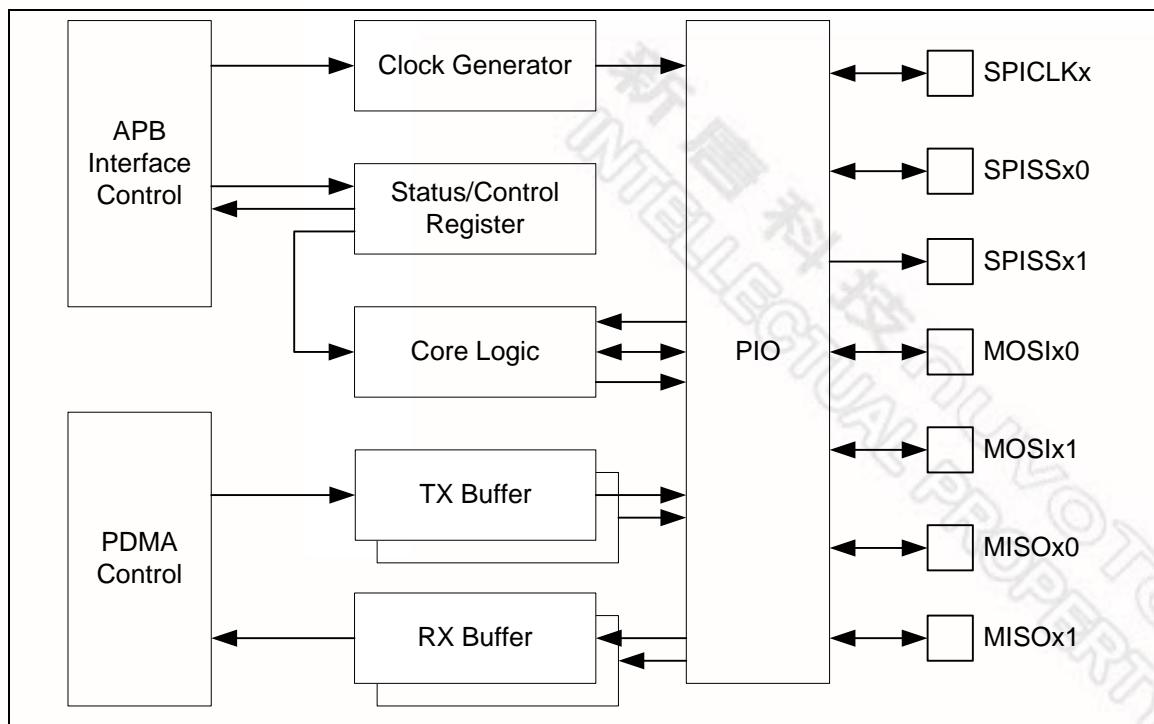


图 5-50 SPI 框图

5.9.4 功能描述

主机/从机模式

SPI 控制器可通过设置 **SLAVE** 位 (SPI_CNTRL[18]), 配置为主机或从机模式, 与外设从机或主机通信。主机模式与从机模式的应用框图如下:

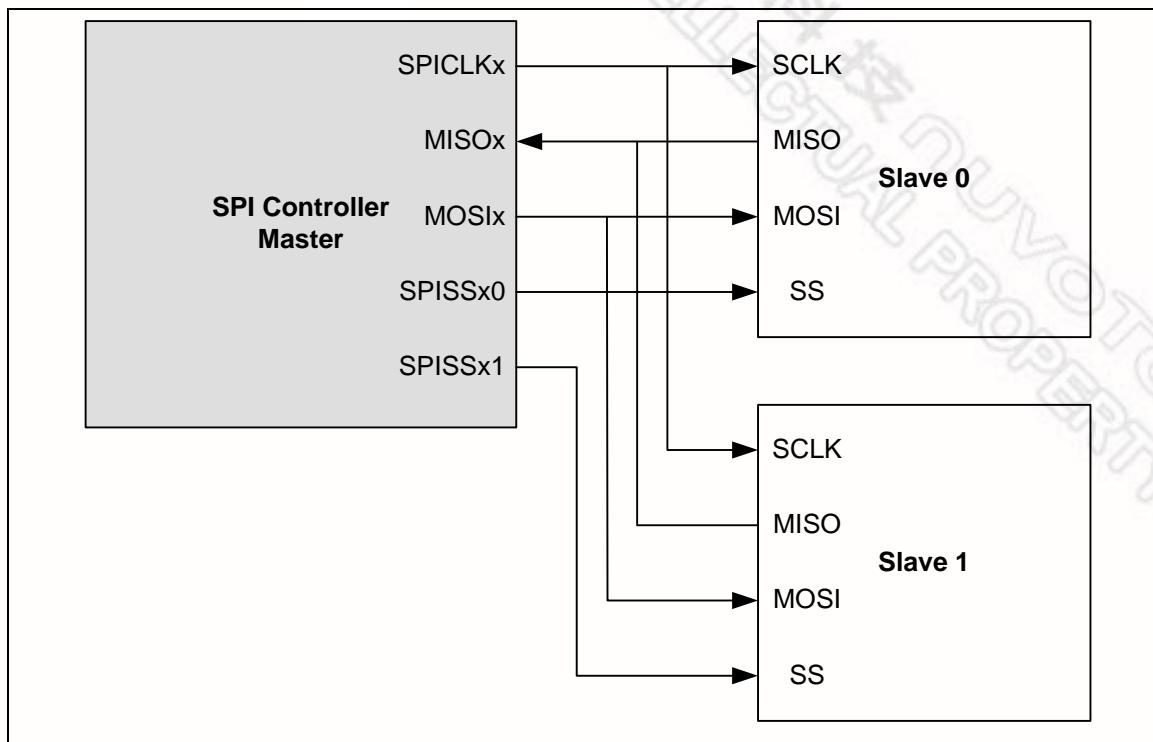


图 5-51 SPI 主机模式应用框图

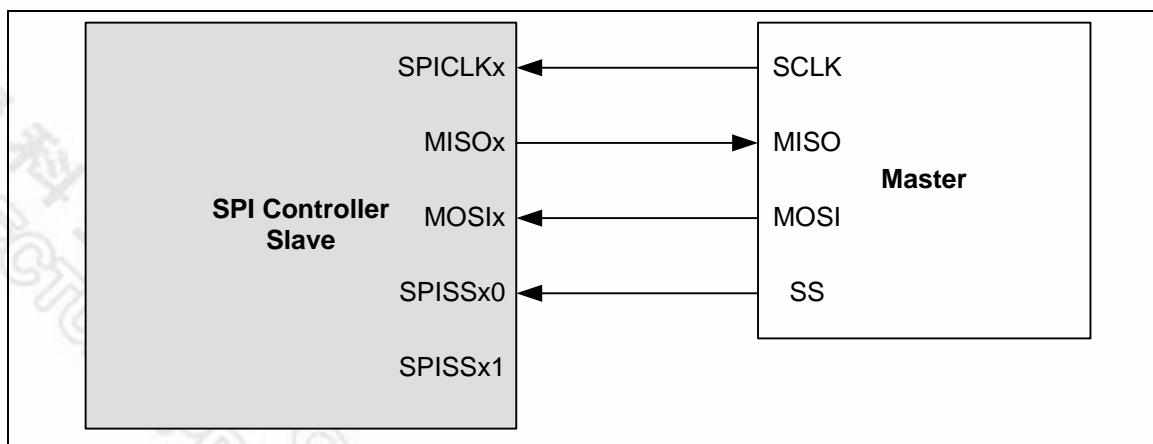


图 5-52 SPI 从机模式应用框图

从机选择

在主机模式下，SPI 控制器能通过从机选择输出脚 SPISSx0 与 SPISSx1 来选择驱动两个作为从机的外围设备。在从机模式下，片外主机设备通过 SPISSx0 输入端口驱动从机选择信号到 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通过编程 SS_LVL 位 (SPI_SSR[2]) 来设定低有效或高有效，SS_LTRIG 位 (SPI_SSR[4]) 定义从机选择信号 SPISSx0/1 为电平触发或边沿触发。触发条件的选择取决于所连接的从机/主机设备的类型。

从机模式下，如果 SS_LTRIG 位被配置成电平触发，则 LTRIG_FLAG 位 (SPI_SSR[5]) 用来表示事务完成后接收数目和接受位是否符合 TX_NUM and TX_BIT_LEN 的设定要求。（一个事务的完成意味着从机选择信号已撤销或 SPI 控制器已经完成了一个数据传输。）

电平触发/边沿触发

从机模式下，从机选择信号可配置成电平触发或边沿触发。当为边沿触发时，数据从检测到有效边沿时刻开始传输，在检测到无效边沿信号时结束。如果主机不发送无效边沿信号给从机，传输过程就不能完成，从机的中断标志位也不能置位。当为电平触发时，有两个条件可以中止传输，并置位中断标志位。第一个条件是如果传输位的数目与 TX_NUM 和 TX_BIT_LEN 的设定相匹配，从机中断将置位。第二个条件是在传输过程中，如果主机设定从机选择管脚为无效电平，这会强制从机设备终止当前传输，并置位中断标志位，这种情形下用户可以读 LTRIG_FLAG 位检查数据是否完全传输。

自动选择从机

在主机模式下，如果置位 AUTOSS (SPI_SSR[3])，将自动产生从机选择信号，并根据 SSR[0] 位 (SPI_SSR[0]) 与 SSR[1] 位 (SPI_SSR[1]) 是否使能，将从机选择信号输出到 SPISSx0 与 SPISSx1 管脚上。这意味着当通过设置 GO_BUSY 位 (SPI_CNTRL[0]) 后发送/接收开始时，在 SSR[1:0] 中选择的从机选择信号将由 SPI 控制器设置为激活状态，在数据传输结束后会被设为非激活状态。It means that the slave select signals, which are selected in SSR[1:0], will be asserted by the SPI controller when transmit/receive is started by setting the GO_BUSY bit (SPI_CNTRL[0]) and will be de-asserted after the data transfer is finished. 当 AUTOSS 位被清零时，从机选择输出信号将会被通过手工设置/清除 SPI_SSR[1:0] 相关位，从而进入激活或非激活状态。从机选择输出信号的激活状态定义在 SS_LVL 位 (SPI_SSR[2])。

串行时钟

在主机模式下，设置 DIVIDER1 位 (SPI_DIVIDER[15:0]) 来编程串行时钟的输出频率到 SPICLK 输出口。如果 VARCLK_EN 位 (SPI_CTL[23]) 使能，也支持可调串行时钟。此时，串行时钟的输出频率可依据 DIVIDER1 (SPI_DIVIDER[15:0]) 和 DIVIDER2 (SPI_DIVIDER[31:16]) 的值被编程为两种不同频率的一种。串行时钟的周期取决于寄存器 SPI_VARCLK 的设定。

在从机模式下，片外主机设备通过 SPICLK 输入端口驱动串行时钟到 SPI 控制器。

可调串行时钟频率

主机模式下，如果可调时钟使能位 VARCLK_EN (SPI_CNTRL[23]) 已使能，那么串行时钟的输出频率可以编程进行可调设定，频率模式格式定义在 VARCLK (SPI_VARCLK[31:0]) 寄存器中。如果 VARCLK 的该位为 '0'，则输出频率取决于 DIVIDER (SPI_DIVIDER[15:0])，如果 VARCLK 的该位 '1'，则输出频率取决于 DIVIDER2 (SPI_DIVIDER[31:16])。图 5-53 为串行时钟 (SPICLK), VARCLK, DIVIDER 和 DIVIDER2 寄存器间的时序关系。VARCLK 的两位决定一个时钟周期。VARCLK[31:30] 定义了 SPICLK 的第一个周期，VARCLK[29:28] 定义了 SPICLK 的第二个周期，依此类推。时钟源的选择定义在 VARCLK 中，它必须在切换到另一个时钟源的前一个周期进行设置。例如，如果 SPICLK 有 5 个 CLK1 周期，VARCLK 必须在 VARCLK 的 MSB 中设置 9 个 '0'，第 10 个位设置为 '1' 来切换下个时钟源为 CLK2。注意当使能 VARCLK_EN 位时，TX_BIT_LEN 必须设置成 0x10（仅 16 位模式）。

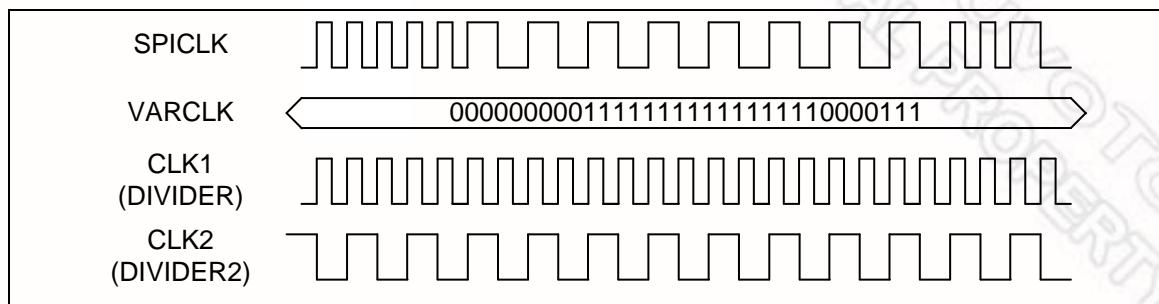


图 5-53 可调串行时钟频率

时钟极性

CLKP 位 (SPI_CTL[11]) 定义串行时钟的空闲状态。当 CLKP = 1，输出 SPICLK 为空闲高电平状态，否则 CLKP = 0 时，输出 SPICLK 为空闲低电平状态。

发送/接收位长度

传输字的长度在 Tx_BIT_LEN 位 (SPI_CNTRL[7:3]) 中定义。发送和接收时传输字可达 32 位长度。

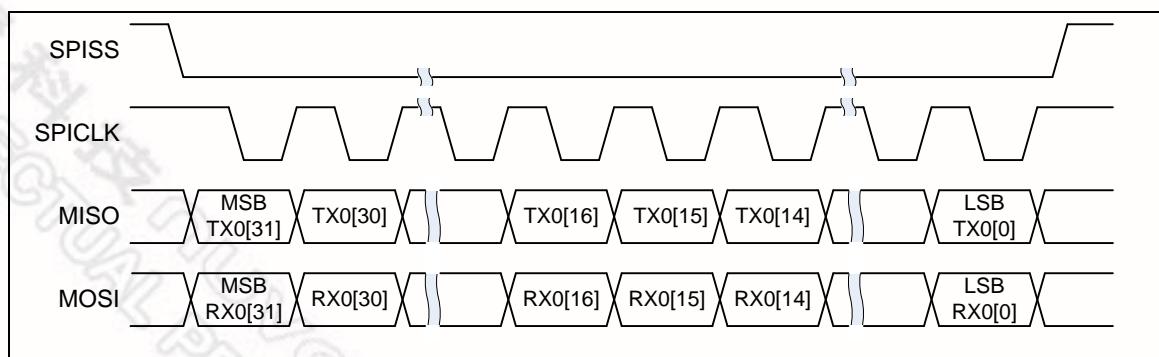


图 5-54 一次传输报文的 32-位

Burst 模式

SPI 控制器通过设置 TX_NUM 位域 (SPI_CNTRL[9:8]) 为 0x01 切换到突发模式。突发模式下，SPI 可以一次发送/接收两个报文。SPI 突发波形如下：

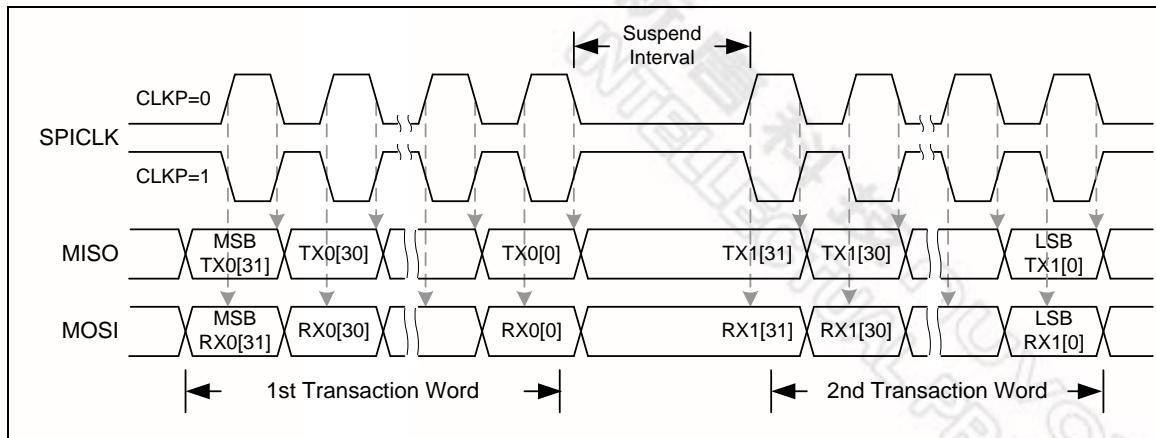


图 5-55 一次传输两个报文 (Burst 模式)

LSB First

LSB 位 (SPI_CNTRL[10]) 定义是从 LSB 还是从 MSB 开始发送/接收数据。

发送边沿

TX_NEG 位 (SPI_CNTRL[2]) 定义数据发送是在串行时钟 SPICLK 的负边沿还是正边沿。

接收边沿

Rx_NEG 位 (SPI_CNTRL[1]) 定义数据接收是在串行时钟 SPICLK 的负边沿还是正边沿。

注：TX_NEG 与 RX_NEG 不可设为相同。换句话说就是不要在同一个时钟边沿发送和接收数据。

字休眠

主机模式下，SP_CYCLE (SPI_CNTRL[15:12]) 4位提供了一个可配置为 2 ~ 17 个串行时钟周期的在两个连续的传输字之间休眠间隔。如果 CLKP = 0，休眠间隔指从前一次传输字的最后一个下降时钟沿到下一次传输字的第一个上升时钟沿。如果 CLKP = 1，间隔时间为前一次传输字的最后一个上升沿到下一次传输字的第一个下降沿。SP_CYCLE 的缺省值为 0x0 (2 个串行时钟周期)，如果 TX_NUM = 0x00，设置这些位不会影响数据传输过程。

字节重排序

当设置为 MSB 优先 (LSB = 0) 且使能 REORDER，在 TX_BIT_LEN = 32-位模式时，存储在 TX 缓存与 RX 缓存的数据将重新按 [BYTE0, BYTE1, BYTE2, BYTE3] 的顺序排列，数据将以 BYTE0, BYTE1, BYTE2, 和 BYTE3 的顺序进行发送/接收。如果 TX_BIT_LEN 设为 24-位模式，存储在 TX 缓存与 RX 缓存的数据将重新按 [unknown byte, BYTE0, BYTE1, BYTE2] 的顺序排列。SPI 控制器将按照 BYTE0, BYTE1, BYTE2 的顺序发送/接收数据，每个字节 MSB 优先发送/接收。16-位模式的规则与上面相同。字节重排序功能只在 TX_BIT_LEN 为16, 24, and 32 位时适用。

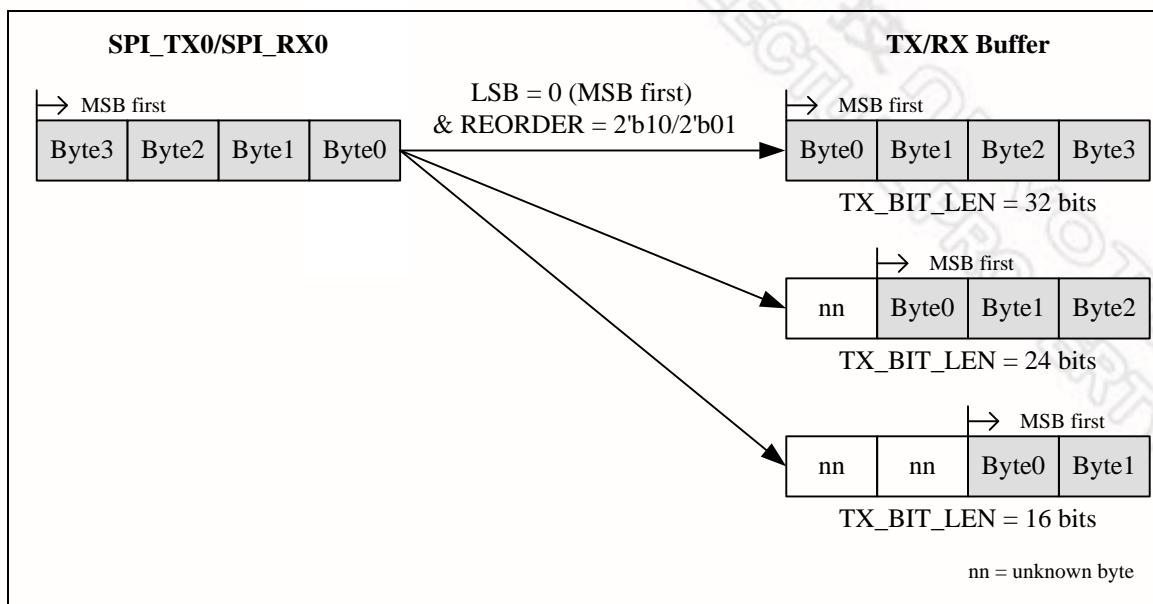


图 5-56 字节重排序

字节休眠

主机模式下，如果 SPI_CNTRL[19] 被设为 1，硬件将会在两个连续传输字节之间插入一个2 ~ 17个串行时钟周期的休眠间隔。字节休眠的设置与字休眠的设置一样都是用 SP_CYCLE。注意当使能字节休眠功能时，TX_BIT_LEN 必须设置为 0x00（每个字传输报文为32-位）。

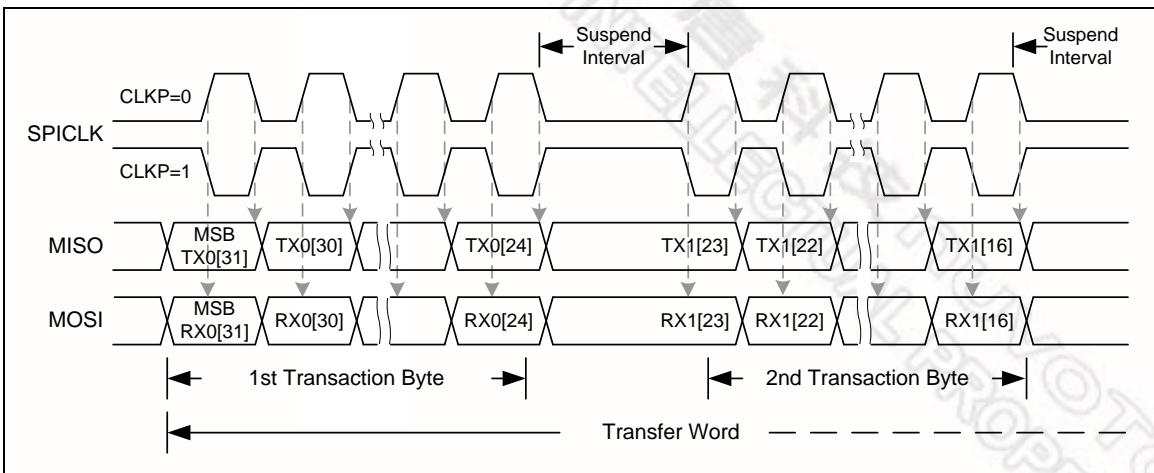


图 5-57 字节休眠的时序波形

REORDER	描述
00	禁用字节重排序功能和字节休眠间隔
01	使能字节重排序功能并在每个字节之间插入一个 (2~17 SPICLK) 的休眠间隔。 TX_BIT_LEN 必须设为 0x00 (32 bits/ word)
10	使能字节重新排序，禁用字节休眠功能。
11	禁用字节重排序功能，但是在每个字节之间插入休眠间隔 (2~17 SPICLK)。TX_BIT_LEN 必须设为 0x00 (32 bits/ word)

表 5-6 字节排序和字节休眠条件

非从机选择模式 (3-线模式)

该模式用于在从机模式下忽略从机选择信号。当被设为从机设备时，SPI 控制器可以工作在非从机选择模式 (3-线模式) 接口 (包括 SPICLK, SPI_MISO, 和 SPI_MOSI)。当 NOSLVSEL 位被设为 1 时，控制器将在 GO_BUSY 位被置为 1 和 串行时钟出现后开始发送/接收数据。在非从机选择模式，SS_LTRIG, SPI_SSR[4]，必须设为 1。

中断

当数据传输完毕时，每个 SPI 控制器可以产生独立的中断，相应的中断事件标志 IF (SPI_CNTRL[16]) 将会被置位。如果中断使能位 IE (SPI_CNTRL[17]) 被置位，则中断事件标志将会向 CPU 提出中断请求。中断标志位 IF 只能写 1 清零。

在 3 线模式下，当传输开始时，SLV_START_INTSTS 中的中断标志会被置位，同样当接收到的数据和 TX_BIT_LEN、TX_NUM 中定义的匹配时，也会有中断事件产生。如果接收到的位数少于要求设定值而且在非从机选择模式时间周期内再没有串行时钟输入，用户可以设置 SLV_ABORT 位来强制结束当前传输，然后用户获得一个传输结束的中断事件。

两位传输模式

当设置 TWOB 位 (SPI_CNTRL[22]) 为 1 时，SPI 控制器支持两-位 (two-bit) 传输模式。当 TWOB 位被使能，可以同时发送和接收两-位串行数据。

例如，在主机模式下，保存在寄存器 SPI_TX0 和寄存器 SPI_RX0 中的数据将会分别从 MOSIx0 管脚和 MOSIx1 管脚发送。同时，寄存器 SPI_RX0 和寄存器 SPI_RX1 将会分别保存从 MISOx0 管脚和 MISOx1 管脚接收到数据。

在从机模式下，保存在寄存器 SPI_TX0 和寄存器 SPI_RX1 中的数据将会分别从 MISOx0 管脚和 MISOx1 管脚发送。同时，寄存器 SPI_RX0 和寄存器 SPI_RX1 将会分别保存从 MOSIx0 管脚和 MOSIx1 管脚接收到数据。

注意当使能 TWOB 位时，TX_NUM 必须编程设置为 0x00。

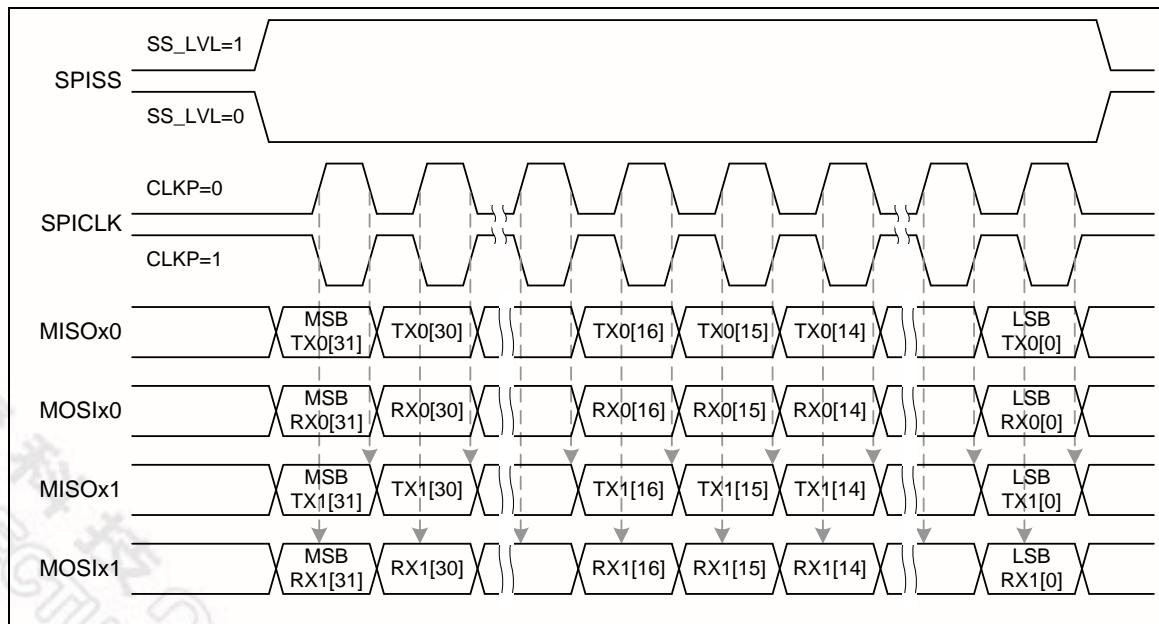


图 5-58 两位传输模式（从机模式）

5.9.5 时序图

从机选择信号的有效状态可以由 SS_LVL 位 (SPI_SSR[2]) 和 SS_LTRIG 位 (SPI_SSR[4]) 的设定定义。串行时钟 (SPICLK) 的空闲状态可以通过 CLKP 位 (SPI_CNTRL[11]) 配置为高电平或低电平。传输字段长度在 TX_BIT_LEN (SPI_CNTRL[7:3]) 中定义，传输的数目在 TX_NUM (SPI_CNTRL[8]) 中定义，发送/接收数据是以 MSB 或 LSB 优先由 LSB 位 (SPI_CNTRL[10]) 定义。用户可以通过设置 TX_NEG/RX_NEG (SPI_CNTRL[2:1]) 寄存器来选择发送/接收数据时串行时钟的边沿。四个 SPI 发送/接收及相关设置如下。

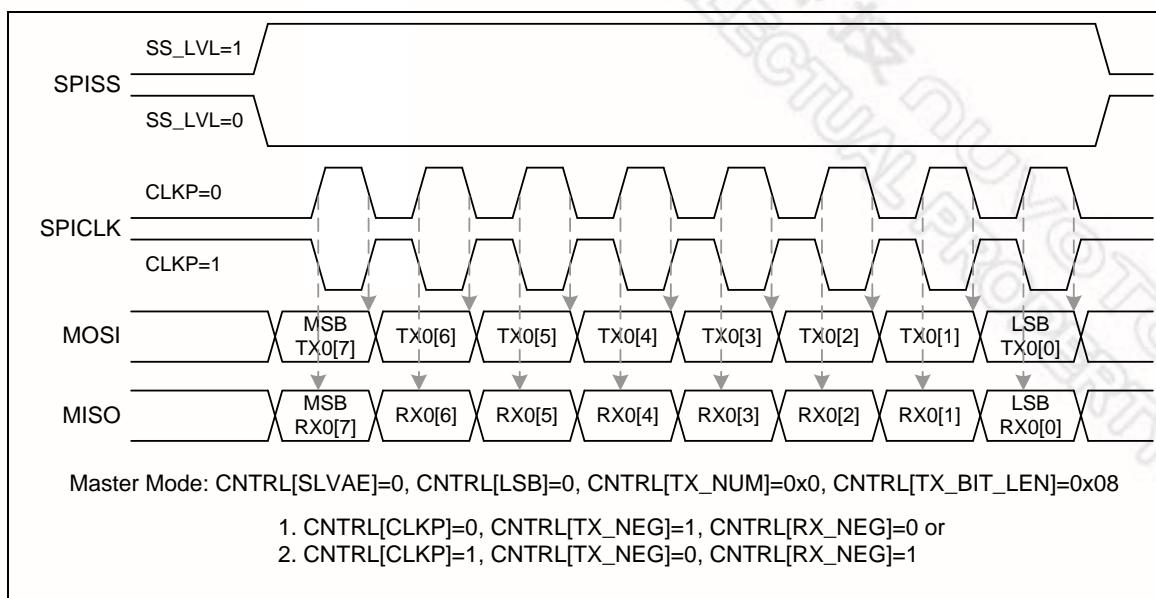


图 5-59 SPI 主机模式下的时序

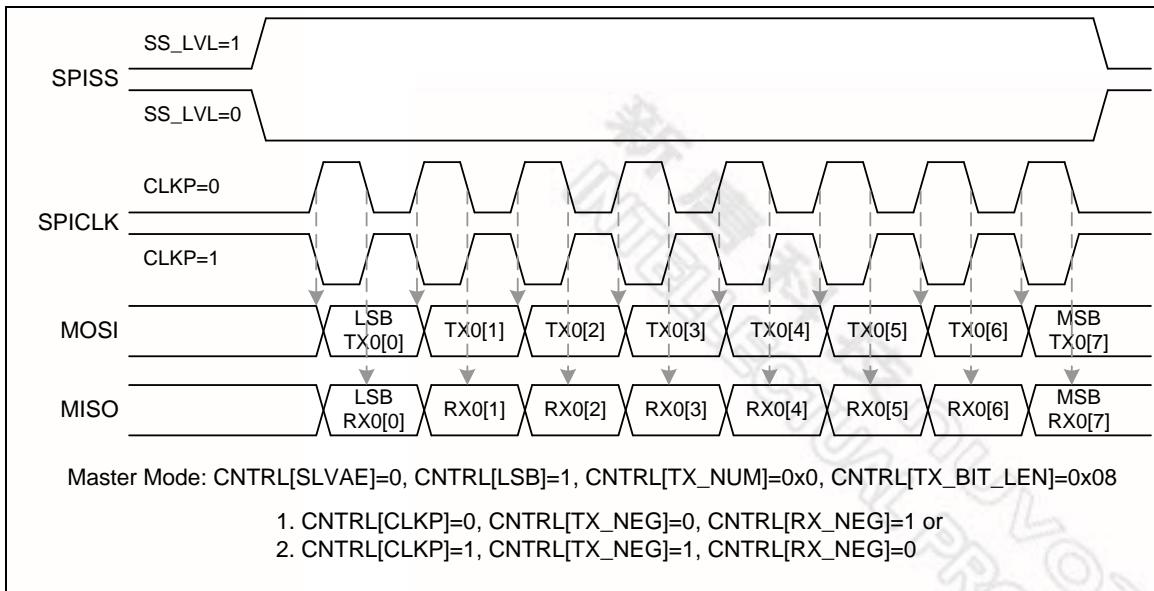


图 5-60 SPI 主机模式下的时序 (Alternate Phase of SPICLK)

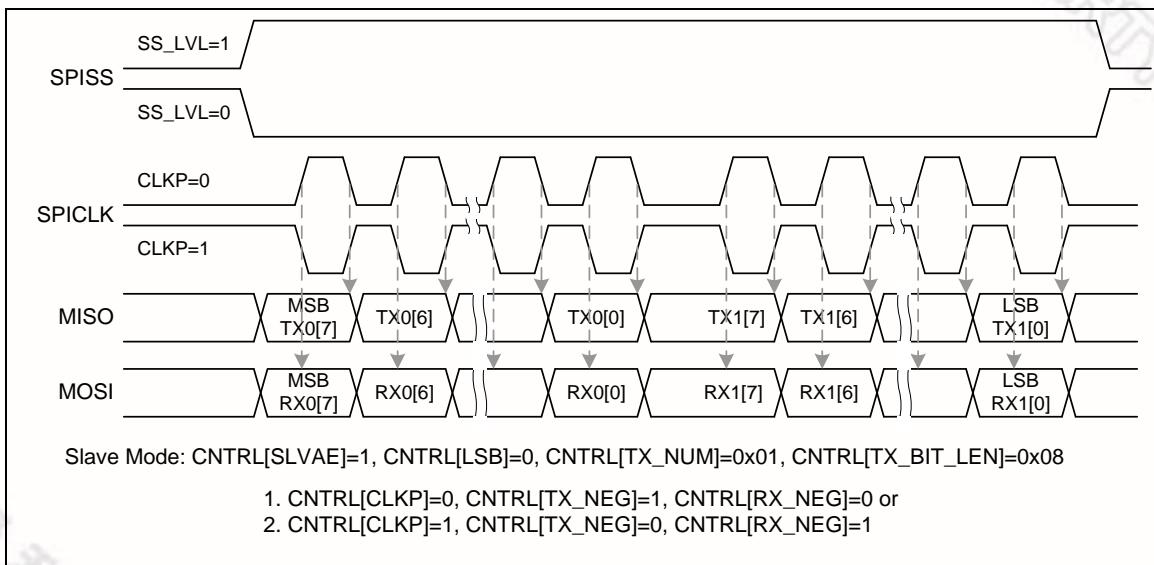


图 5-61 SPI 从机模式下的时序

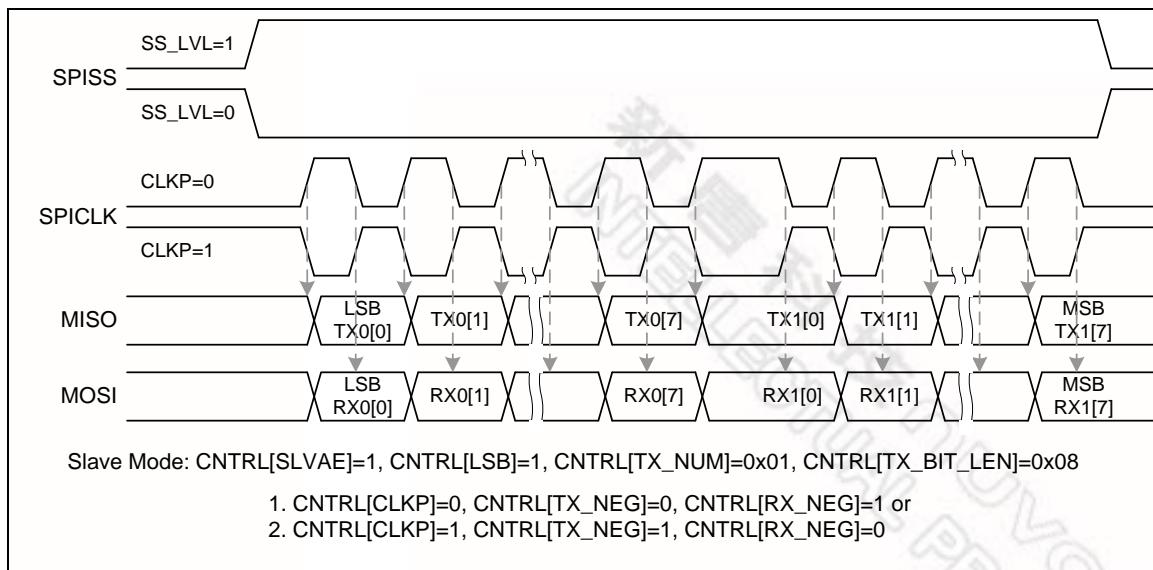


图 5-62 SPI 从机模式下的时序 (Alternate Phase of SPICLK)

5.9.6 编程例程

例 1, SPI 控制器作为主机去访问一个片外从机设备，过程如下：

- 数据在串行时钟正边沿锁存
- 数据在串行时钟负边沿传输
- MSB 先传输
- SPICLK 空闲模式为低电平状态
- 每次发送/接收只有一个字节
- 使用第一个 SPI 从机选择管脚和一个片外从机相连。从机选择信号为低电平有效

操作流程如下：

- 1) 设定寄存器 DIVIDER (SPI_DIVIDER[15:0]) 来决定串行时钟输出频率
- 2) 将主机模式的相应配置值写入 SPI_SSR 寄存器
 1. 禁用自动从机选择位 AUTOSS(SPI_SSR[3] = 0)
 2. 设置从机选择有效电平位 SS_LVL (SPI_SSR[2] = 0)，使从机选择信号为低电平触发输出。
 3. 设置相关从机选择寄存器位 SSR[0] (SPI_SSR[0]) 来选择从机选择信号的输出 IO 管脚，从而激活相关的片外从机设备。
- 3) 通过设置 SPI_CNTRL 寄存器来控制 SPI 主机的行为。
 1. 通过设置 SLAVE 位 (SPI_CNTRL[18] = 0) 将 SPI 控制器设为主机设备。
 2. 通过设置 CLKP 位 (SPI_CNTRL[11] = 0) 将串行时钟的空闲状态设为低电平。

3. 通过设置 TX_NEG 位 (`SPI_CNTRL[2] = 1`) 选择数据在串行时钟的负边沿传输。
4. 通过设置 RX_NEG 位 (`SPI_CNTRL[1] = 0`) 选择数据锁存在串行时钟的正边沿。
5. 通过设置 TX_BIT_LEN 位域 (`SPI_CNTRL[7:3] = 0x08`) 设定一次字传输的长度为8-位。
6. 通过设置 TX_NUM (`SPI_CNTRL[9:8] = 0x0`) 设定为一次字传输
7. 通过设置 MSB 位 (`SPI_CNTRL[10] = 0`) 设定为 MSB 传输优先，本例中不必理会 SP_CYCLE 位域 (`SPI_CNTRL[15:12]`)，因为没有使用 burst 模式。
- 4) 如果 SPI 主机要发送（写）一个字节的数据到片外从机设备，则将所要发送到数据写入寄存器 `TX0[7:0]` (`SPI_TX0[7:0]`)。
- 5) 如果 SPI 主机只是要从片外从机设备接收（读）一个字节的数据，不必管被传输出去的数据是什么，只需要向寄存器 `SPI_RX0[7:0]` 写入 `0xFF`。
- 6) 使能 GO_BUSY 位 (`SPI_CNTRL[0] = 1`) 开始 SPI 接口的数据传输。
- 7) 等待 SPI 中断发生（如果中断使能位 IE 使能）或轮询检测 GO_BUSY 位直到被硬件自动清零。
- 8) 从寄存器 `RX0 [7:0]` (`SPI_RX0[7:0]`) 中读取接收到的一个字节数据。
- 9) 重复步骤 4) 继续其他数据传输或设置 SSR [0] 为 0 来停止片外从机设备。

例 2, SPI 控制器作为从机设备，和一片外主机设备相连，外设通过 SPI 接口与片上 SPI 从机通信。过程如下：

- 数据在串行时钟正边沿锁存
- 数据在串行时钟负边沿传输
- LSB 先传输
- SPICLK 空闲状态为高电平
- 每次发送/接收一个字节
- 从机选择信号为高电平触发

操作流程如下：

- 1) 将从机模式的相应配置值写入 SPI_SSR 寄存器
设置从机选择有效电位 SS_LVL (`SPI_SSR[2] = 1`) 和从机选择触发电位 SS_LTRIG (`SPI_SSR[4] = 1`) 来选择高电平触发作为从机选择信号。
- 2) 通过设置 SPI_CNTRL 寄存器来控制 SPI 从机的行为。
 1. 通过设置 SLAVE 位 (`SPI_CNTRL[18] = 1`) 将 SPI 控制器设为从机设备
 2. 通过设置 CLKP 位 (`SPI_CNTRL[11] = 1`) 将串行时钟的空闲状态设为高电平
 3. 通过设置 TX_NEG 位 (`SPI_CNTRL[2] = 1`) 选择数据在串行时钟的负边沿传输
 4. 通过设置 RX_NEG 位 (`SPI_CNTRL[1] = 0`) 选择数据锁存在串行时钟的正边沿。
 5. 通过设置 TX_BIT_LEN 位域 (`SPI_CNTRL[7:3] = 0x08`) 设定一次字传输的长度为8-位。

6. 通过设置 TX_NUM (**SPI_CNTRL[9:8] = 0x0**) 设定为一次字传输
7. 通过设置 LSB 位 (**SPI_CNTRL[10] = 1**) 设定为 LSB 传输优先, 本例中不必理会 SP_CYCLE 位域 (**SPI_CNTRL[15:12]**), 因为没有使用 burst 模式。
- 3) 如果 SPI 从机要发送一个字节的数据到外设主机, 则将所要发送的数据写入到寄存器 **TX0[7:0]** (**SPI_TX0[7:0]**)
- 4) 如果 SPI 从机只是要从外设主机接收一字节数据, 用户不必关心什么数据将被传输, 只需要向寄存器 **SPI_RX0[7:0]** 写入 **0xFF**。
- 5) 使能 GO_BUSY 位 (**SPI_CNTRL[0] = 1**) 来等待片外主机设备的从机选择触发输入和串行时钟输入, 以便开始在 SPI 接口的数据传输。
- 6) 等待 SPI 中断发生 (如果中断使能位 IE 使能) 或轮询检测 GO_BUSY 位直到被硬件自动清零。
- 7) 从寄存器 **RX0 [7:0]** (**SPI_RX0[7:0]**) 中读取接收到的一个字节数据。
- 8) 重复步骤 3) 继续其他数据传输或禁用 GO_BUSY 位来停止数据传输。



5.9.7 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
SPI0_BA = 0x4003_0000				
SPI1_BA = 0x4003_4000				
SPI2_BA = 0x4013_0000				
SPI3_BA = 0x4013_4000				
SPI_CTRL	SPIx_BA+0x00	R/W	控制与状态寄存器	0x0500_0004
SPI_DIVIDER	SPIx_BA+0x04	R/W	时钟分频寄存器	0x0000_0000
SPI_SSR	SPIx_BA+0x08	R/W	从机选择寄存器	0x0000_0000
SPI_RX0	SPIx_BA+0x10	R	数据接收寄存器0	0x0000_0000
SPI_RX1	SPIx_BA+0x14	R	数据接收寄存器1	0x0000_0000
SPI_TX0	SPIx_BA+0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1	SPIx_BA+0x24	W	数据发送寄存器1	0x0000_0000
SPI_VARCLK	SPIx_BA+0x34	R/W	可调时钟类型寄存器	0x007F_FF87
SPI_DMA	SPIx_BA+0x38	R/W	SPI DMA 控制寄存器	0x0000_0000
SPI_CTRL2	SPIx_BA+0x3C	R/W	控制与状态寄存器2	0x0000_0000

Note: 当软件编程 CNTRL, GO_BUSY 位必须最后写。



5.9.8 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	控制及状态寄存器	0x0500_0004

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
VARCLK_EN	TWOB	Reserved	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	
[31:24]	Reserved	保留
[23]	VARCLK_EN	<p>可调时钟使能 (Master Only) 1 = 串行时钟输出频率可调。输出频率由 VARCLK, DIVIDER, 和 DIVIDER2 的值决定。 0 = 串行时钟输出频率固定，只由 DIVIDER 的值决定。 注：当使能 VARCLK_EN 位， TX_BIT_LEN 必须设置为 0x10 (16-bit mode)</p>
[22]	TWOB	<p>两位 (Two Bits) 传输模式有效 1 = 使能 2-bit 传输模式 0 = 禁用 2-bit 传输模式 注意当使能 TWOB，从 SPI_TX1/0 发送两-位数据，接收两-位数据到 SPI_RX1/0。 当使能 TWOB， TX_NUM 必须设置为 0x00</p>
[21]	Reserved	保留
[20:19]	REORDER	<p>重排序模式选择 00 = 禁用字节重排序和字节休眠功能 01 = 使能字节重排序功能，在每个字节之间插入一个字节休眠间隔（2~17 SPICLK 周期）。TX_BIT_LEN 必须设为 0x00。 (32 bits/word) 10 = 使能字节重排序功能，禁用字节休眠功能。 11 = 禁用字节重排序功能，在每个字节之间插入一个字节的休眠间隔（2~17 SPICLK 周期）。TX_BIT_LEN 必须设为 0x00。 (32 bits/word) 注： 1. 字节重排功能可用时， TX_BIT_LEN 必须设为 16, 24 和 32 位。 2. 在从机电平触发模式下，若使能字节休眠功能，主机输出的从机选择管脚信号在连</p>

		续的四个字节传输过程中，必须保持有效状态。
[18]	SLAVE	从机模式指示 1 = 从机模式 0 = 主机模式
[17]	IE	中断使能 1 = 使能 SPI 中断 0 = 禁用 SPI 中断
[16]	IF	中断标志位 1 = 表示传输完成 0 = 表示传输还没有完成 注：该位写 1 清零。
[15:12]	SP_CYCLE	休眠间隔 (Master Only) 这四位用于配置两个连续发送/接收报文间的休眠间隔。如果 CLKP = 0，休眠间隔从当前传输的最后一次下降时钟沿到连续传输的第一次上升时钟沿。如果 CLKP = 1，休眠间隔则指从上升时钟沿到下降时钟沿。默认值为0x0。当 TX_NUM = 00b，设置该位对传输没有影响。休眠间隔见如下方程： <ul style="list-style-type: none"> ■ 字节休眠间隔和 burst 模式休眠间隔： $(SP_CYCLE[3:0] + 2) * \text{SPICLK 周期} + 1 \text{ 系统时钟周期}$ <p>例如：</p> $\begin{aligned} SP_CYCLE &= 0x0 \dots 2 \text{ SPICLK 时钟周期} + 1 \text{ 系统时钟周期} \\ SP_CYCLE &= 0x1 \dots 3 \text{ SPICLK 时钟周期} + 1 \text{ 系统时钟周期} \\ &\dots \\ SP_CYCLE &= 0xE \dots 16 \text{ SPICLK 时钟周期} + 1 \text{ 系统时钟周期} \\ SP_CYCLE &= 0xF \dots 17 \text{ SPICLK 时钟周期} + 1 \text{ 系统时钟周期} \end{aligned}$ <p>如果 SPI 时钟速率等于系统时钟速率，这就是说，使能 DIV_ONE，burst 模式休眠间隔周期为</p> $(SP_CYCLE[3:0] * 2 + 3.5) * \text{系统时钟周期}$
[11]	CLKP	时钟极性 1 = SPICLK 空闲高电平 0 = SPICLK 空闲低电平
[10]	LSB	优先传输 LSB 1 = 优先发送 LSB (SPI_TX0/1 的比特 0 位)，并且接收到的第一个比特位将放到 RX 寄存器的 LSB 位置 (SPI_RX0/1 的比特 0 位)。 0 = 优先发送/接收 MSB (根据 TX_BIT_LEN 的值决定 SPI_TX0/1 和 SPI_RX0/1 寄存器的第一个位的编号)。
[9:8]	TX_NUM	发送/接收的字数目 该域用于标示一次传输中，发送/接收的字数目。 00 = 一次传输仅发送/接收一个字

		01 = 一次传输发送/接收两个连续的字 (burst mode) 10 = 保留 11 = 保留 注：从机电平触发模式时，若 TX_NUM 设置为01，从机选择管教信号在连续的数据传输过程中，必须保持有效状态。
[7:3]	TX_BIT_LEN	传输位长度 该域用于标示一次传输中，完成的传输比特位，最高为32位 TX_BIT_LEN = 0x01 ... 1 bit TX_BIT_LEN = 0x02 ... 2 bits TX_BIT_LEN = 0x1F ... 31 bits TX_BIT_LEN = 0x00 ... 32 bits
[2]	TX_NEG	发送数据边沿反向位 1 = 在 SPICLK 下降沿改变发送数据输出信号 0 = 在 SPICLK 上升沿改变发送数据输出信号
[1]	RX_NEG	负边沿接收 1 = 在 SPICLK 的下降沿锁存接收数据输入信号 0 = 在 SPICLK 的上升沿锁存接收数据输入信号
[0]	GO_BUSY	通讯或忙状态标志 1 = 在主机模式下，写 1 到该位开始 SPI 数据传输；在从机模式下，写 1 到位表示从机准备好与主机进行通信。 0 = 当 SPI 在传输时，写 0 到该位停止数据传输。 数据传输过程中，该位值保持为 1，当传输完成后，该位自动清零。 注： 1. 在写 1 到 GO_BUSY 位之前，所有的寄存器必须设置完成。

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 (Master Only)	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]	DIVIDER2	<p>时钟分频 2 寄存器 (master only)</p> <p>这些位是设置第二个频率分频器，用于产生 SPICLK 输出的串行时钟。可根据下列公式获得所期望的频率：</p> $f_{sclk} = \frac{f_{pclk}}{(DIVIDER2 + 1) * 2}$ <p>当 VARCLK_EN 被清为零时，该设置无意义。</p>
[15:0]	DIVIDER	<p>时钟分频寄存器 (master only)</p> <p>这些位是设置频率分频器，用于产生 SPICLK 输出的串行时钟。可根据下列公式获得所期望的频率：</p> $f_{sclk} = \frac{f_{pclk}}{(DIVIDER + 1) * 2}$ <p>从机模式下，驱动 SPI 的主机时钟周期必须等于或大于 PCLK 周期的5倍。也就是 SPI 所能接受的最高时钟频率是从机 PCLK 的1/5。</p>

SPI 从机选择寄存器 (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	从机选择寄存器	0x0000_0000

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

Bits	描述	
[31:6]	Reserved	保留
[5]	LTRIG_FLAG	<p>电平触发标志 当 SS_LTRIG 位在从机模式下被置位，读该位的值可用来表示接收位的数量是否达到要求。 1 = 接收数量和接收位达到 TX_NUM 及 TX_BIT_LEN 内的设置值 0 = 接收数量或接收位没有符合设置值 注：该位只读。</p>
[4]	SS_LTRIG	<p>从机选择电平触发 (Slave only) 1 = 从机选择信号将是电平触发。根据 SS_LVL 来决定是高电平/低电平触发。 0 = 输入从机选择信号是边沿触发，该值为默认值。根据 SS_LVL 来决定是下降沿/上升沿触发。</p>
[3]	AUTOSS	<p>自动从机选择 (Master only) 1 = 该位置位，SPISSx0/1 信号自动产生。这表示当通过设置 GO_BUSY 位开始发送/接收时，SSR[1:0] 中设定的设备/从机选择信号将由 SPI 控制器设为有效状态，而当每次发送/接收结束时，设备/从机选择信号又会设为无效状态。 0 = 如果该位清零，从机选择信号将由 SSR[1:0] 相关位的设定值来决定激活或失效。</p>
[2]	SS_LVL	<p>从机选择触发电平 定义从机选择信号 (SPISSx0/1) 的有效状态。 1 = 从机选择信号 SPISSx0/1 在高电平/上升沿有效 0 = 从机选择信号 SPISSx0/1 在低电平/下降沿有效</p>
[1:0]	SSR	<p>从机选择寄存器 (Master only) 如果 AUTOSS 位被清零，写 1 到 SSR[1:0] 任一位将会激活所对应的 SPISSx0/1</p>



	<p>线, 写 0 则相应的 SPISSx0/1 线为非激活状态。</p> <p>如果 AUTOSS 位置 1, 写 0 到 SSR[1:0] 任一位将会保持相应的 SPISSx0/1 线为非激活状态; 写 1 到 SSR[1:0] 任一位将会选择相应的 SPISSx0/1 线在发送/接收的时间内被自动驱动到激活状态, 而其他时间为非激活状态。SPISSx0/1 的激活状态类型由 SS_LVL 指定。</p> <p>注: SPISSx0 在从机模式下被定义为从机选择输入。</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
SPI_RX0	SPIx_BA+0x10	R	数据接收寄存器0	0x0000_0000
SPI_RX1	SPIx_BA+0x14	R	数据接收寄存器1	0x0000_0000

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

Bits	描述	
[31:0]	RX	<p>数据接收寄存器</p> <p>数据接收寄存器保存了上一次执行的传输所接收到的数据。数据的有效长度根据寄存器 SPI_CNTRL 定义的传输长度决定。</p> <p>例如，如果 TX_BIT_LEN 为 0x08，TX_NUM 为 0x0，RX0[7:0] 位保存接收到的数据，其他位的值为不确定值。</p> <p>注：数据接收寄存器为只读寄存器。</p>

SPI 数据发送寄存器 (SPI_TX)

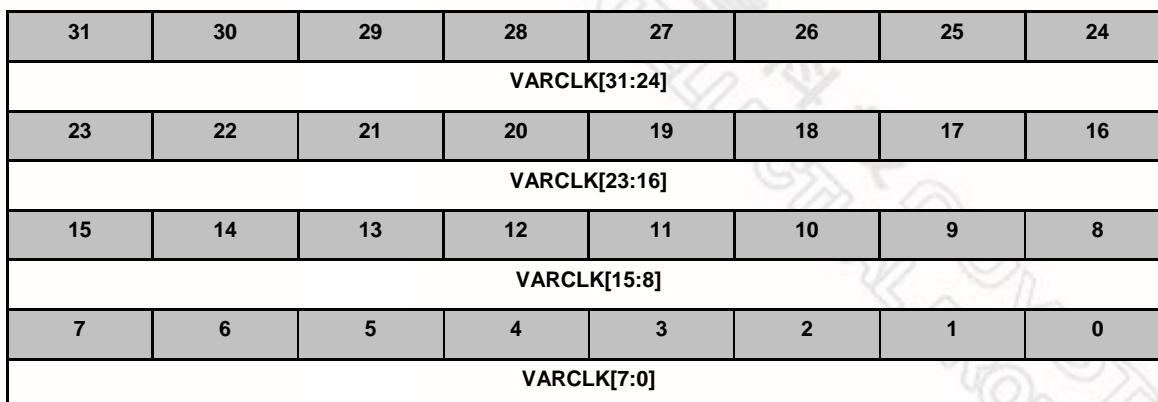
寄存器	偏移量	R/W	描述	复位后的值
SPI_TX0	SPIx_BA+0x20	W	数据发送寄存器0	0x0000_0000
SPI_TX1	SPIx_BA+0x24	W	数据发送寄存器1	0x0000_0000

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

Bits	描述	
[31:0]	TX	<p>数据发送寄存器</p> <p>数据发送寄存器内存储下一次被发送的数据。数据的有效长度根据 CNTRL 寄存器内定义的长度决定。</p> <p>例如，如果 TX_BIT_LEN 为 0x08，TX_NUM 为 0x0，则 TX0[7:0] 位将会被发送。如果 TX_BIT_LEN 为 0x00，TX_NUM 为 0x1，则 SPI 控制器将以相同的设置连续发送/接收 两个 32-位数据。传送序列是先 TX0[31:0] 然后 TX1[31:0]。</p>

SPI 可调时钟类型寄存器 (SPI_VARCLK)

寄存器	偏移量	R/W	描述	复位后的值
SPI_VARCLK	SPIx_BA+0x34	R/W	可调时钟类型寄存器	0x007F_FF87



Bits	描述	
[31:0]	VARCLK	<p>可调时钟类型</p> <p>该寄存器的值表示 SPI 时钟的频率类型。如果 VARCLK 的类型为'0'，则 SPICLK 的输出频率取决于 DIVIDER 的值。如果 VARCLK 的类型为'1'，则 SPICLK 的输出频率取决于 DIVIDER2 的值。参考寄存器 SPI_DIVIDER。</p> <p>更多详细细节请参考可调串行时钟频率章节。</p>

DMA 控制寄存器 (DMACTL)

寄存器	偏移量	R/W	描述	复位后的值
SPI_DMA	SPIx_BA+0x38	R/W	SPI DMA 模式控制寄存器	0x0000_0000

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

Bits	描述	
[31:2]	Reserved	保留
[1]	RX_DMA_GO	<p>接收 DMA 开始</p> <p>对该位置 1 将开始 PDMA 接收过程。SPI 控制器将自动向 PDMA 控制器发出请求。</p> <p>在 PDMA 传输完成后，硬件将自动清该位为 0。</p>
[0]	TX_DMA_GO	<p>发送 DMA 开始</p> <p>对该位置 1 将开始 PDMA 发送过程。SPI 控制器将自动向 PDMA 控制器发出请求。</p> <p>如果使用 PDMA 模式来传输数据，不需要置位 SPI_CNTRL 寄存器的 GO_BUSY 位。SPI 控制器内的 DMA 控制器将会在需要的时候自动设置该位。</p> <p>在 PDMA 传输完成后，硬件将自动清该位为 0。</p>

SPI 控制与状态寄存器 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	SPI 控制与状态寄存器 2	0x0000_0000

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

Bits	描述	
[31:12]	Reserved	保留
[11]	SLV_START_INTSTS	<p>从机开始中断状态 该位用来表示在没有从机选择的从机模式下，传输已经开始。 1 = 表示在没有从机选择的从机模式下，传输开始。该位在传输完成后自动清位或写 1 清位。 0 = 表示从机开始传输没激活。</p>
[10]	SSTA_INTEN	<p>从机开始中断使能 当在没有从机选择的从机模式下传输开始时，该位用于使能中断。如果在传输开始后的（用户定义）时间后，没有传输完成的中断，则用户可以置位 SLV_ABORT 位强制完成传输。 1 = 使能传输开始中断。该位在传输完成后清位或 SLV_START_INTSTS 位被清除（写 1 清除） 0 = 禁用传输开始中断。</p>
[9]	SLV_ABORT	<p>无从机选择的从机模式中的异常中止 在正常操作中，当接收到数据符合 TX_BIT_LEN 和 TX_NUM 的要求位的值时，会有中断事件。 如果接收到的位数少于要求的值而且在无从机选择的从机模式下，在一次传输的时间后没有更多的串口时钟输入时，用户可以设定该位来强制完成当前的传输，然后用户就可以收到一个传输完成的中断事件。 注：当异常中止事件激活时该位会被硬件自动清零。</p>
[8]	NOSLVSEL	从机模式下的无从机选择

		该位用于忽略从机模式下的从机选择信号。当 SPI 控制器被设置为从机设备时，它可以工作于3线接口，包括 SPICLK, SPI_MISO 和 SPI_MOSI。 0 = 控制器是 4-线双向接口。 1 = 控制器是 3-线双向接口（从机模式下）。当该位被置 1 时，在 GO_BUSY 位激活和串行时钟输入后，控制器开始发送/接收数据。 注：在无从机选择信号的模式下，SS_LTRIG, SPI_SSR[4] 必须设为 1。
[7:1]	Reserved	保留
[0]	DIV_ONE	<p>SPI 时钟分频器控制</p> <p>0 = SPI 时钟速度由 SPI_DIVIDER 寄存器的设定决定。</p> <p>1 = 使能 DIV_ONE 特性。SPI 时钟速度等于系统时钟速度。</p> <p>注：</p> <ol style="list-style-type: none"> 当该位为 1 时，REORDER 域和 VARCLK_EN 域必须配置为0。也就是字节重排序功能，字节休眠功能和可调时钟功能都必须禁用。 当该位被置 1 时，TX_BIT_LEN 不能为1。



5.10 定时器控制器 (TMR)

5.10.1 概述

定时器控制器包含 4 组 32-位定时器，TIMER0~TIMER3，提供用户便捷的计数定时功能。定时器模块可支持例如频率测量，计数，间隔时间测量，时钟产生，延迟时间等功能。定时器可在计时溢出时产生中断信号，也可在操作过程中提供计数的当前值。

5.10.2 特征

- 4 组 32-位定时器，带24位向上定时器和一个8位的预分频计数器
- 每个定时器都有独立的时钟源
- 提供 one-shot, periodic, toggle 和 continuous 计数操作模式
- 超时周期 = (输入的定时器时钟周期) * (8-位预分频计数器 + 1) * (24-位 TCMP)
- 最大计数周期 = $(1 / T \text{ MHz}) * (2^8) * (2^{24})$, T 是定时器周期
- 通过 TDR (定时器数据寄存器) 可读取内部 24 位向上计数器的值
- 支持事件计数功能可用于计数外部管脚的事件
- 支持输入捕捉功能可用于捕捉或复位计数值

5.10.3 框图

每个通道带一个 8 位预分频计数器，一个 24 位向上计数器，一个 24 位比较寄存器和一个中断请求信号。参考 图 5-63 定时器控制框图。每个通道有 4 个时钟源选项。图 5-64 为时钟源控制功能。软件可以变成 8-位 预分频计数器来决定 24-位向上计数器的时钟周期。

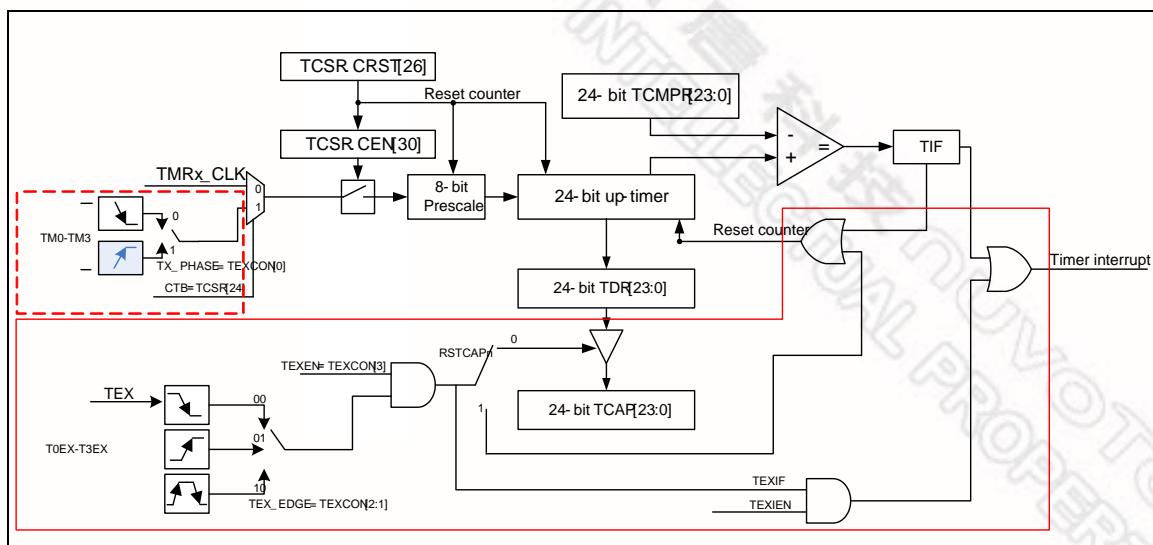


图 5-63 定时器控制器框图

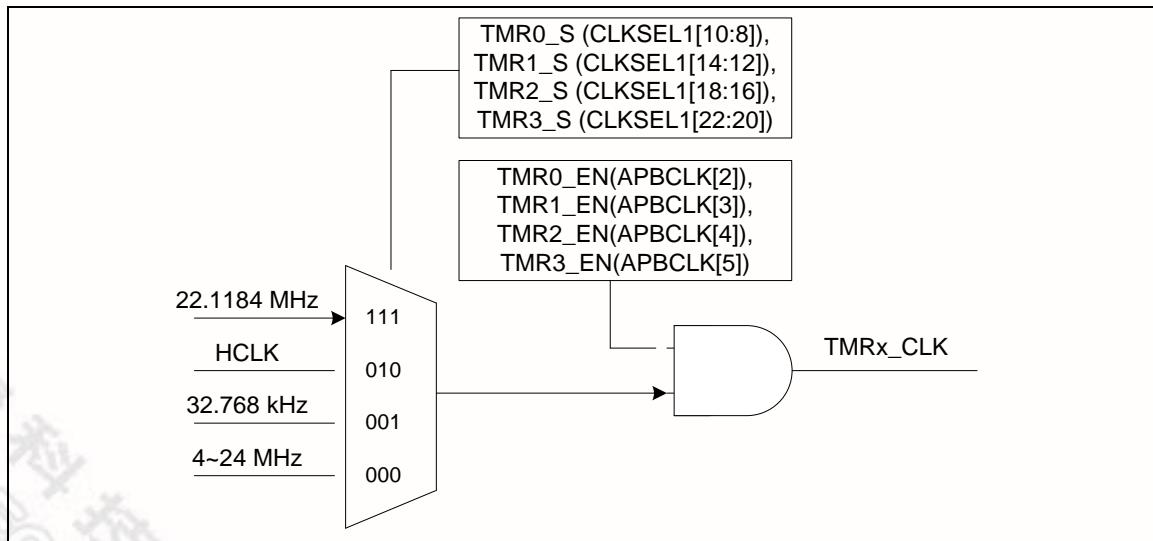


图 5-64 定时器控制器的时钟源

5.10.4 功能描述

定时器控制器提供 one-shot, period, toggle 和 continuous counting 模式操作，还提供事件计数功能来计数来自外部管脚的事件，输入捕捉功能用来捕捉或复位定时器计数值。每种操作功能模式如下所示：

5.10.4.1 One -Shot 模式

如果定时器工作在单周期 (one-shot) 模式且 CEN (TCSR[30] 定时器使能位) 置1，则定时器的计数器开始计数。一旦定时器计数器的值达到定时器比较寄存器 (TCMPR) 的值，且 IE (TCSR[29] 中断使能位) 置1，则定时器中断标志置位，产生中断信号并送到 NVIC 通知 CPU，表明定时器计数发生溢出。如果IE (TCSR[29] 中断使能位) 置0，无中断信号发生。在此工作模式下，一旦定时器计数器的值达到定时器比较寄存器 (TCMPR) 的值时，定时器计数器的值回到计数初始值，而且 CEN 位 (定时器使能位) 被定时器控制器清零。定时器计数操作停止，在设定定时器比较寄存器 (TCMPR) 的值和置CEN 位 (定时器使能位) 为 1 后，该比较的操作仅执行一次。因此，该操作称为One-Shot 模式。

5.10.4.2 Periodic 模式

如果定时器工作在周期模式且 CEN (TCSR[30]定时器使能位) 置1，定时器计数器开始计数。一旦定时器计数器的值达到定时器比较寄存器 (TCMPR) 的值，且 IE (中断使能位) 设置为1，则定时器中断标志置位且产生中断信号，并发送到 NVIC 通知 CPU，表示定时器计数溢出发生。如果 IE (TCSR[29] 中断使能位) 置为 0，无中断信号发生。在该工作模式下，一旦定时器计数器的值达到定时器比较器寄存器 (TCMPR) 的值，定时器计数器的值返回计数初始值且 CEN 保持为1 (持续使能计数)。定时器计数器再次计数。如果软件清除中断标志，一旦定时器计数器的值与定时器比较寄存器 (TCMPR) 的值匹配且 IE (中断使能位) 设置为1'b1，产生中断信号并送到 NVIC 通知 CPU。也就是说，定时器计数与 TCMPR 比较的操作是周期性的。直到 CEN 设置为0，定时器计数操作才会停止，中断信号的产生也是周期性的。因此，这种操作模式称为Periodic 模式。

5.10.4.3 Toggle 模式

如果定时器工作在 toggle 模式且 CEN (TCSR[30] 定时器使能位) 置1，定时器计数器开始计数。一旦定时器计数器的值与定时器比较寄存器 TCMR 的值匹配时，且 IE (TCSR[29] 中断使能位) 设置为 1，则定时器中断标志置位，产生中断信号并送到 NVIC 通知 CPU。表示定时器发生计数溢出。相应 toggle 输出(tout) 信号置1。在这种操作模式，一旦定时器计数器的值与定时器比较寄存器 (TCMPR) 的值匹配，定时器计数器的值返回到计数初始值且 CEN 保持为 1 (持续使能计数)。定时器计数器重新计数。如果中断标志由软件清除，一旦定时器计数器的值与定时器比较寄存器中 (TCMPR) 的值匹配且 IE (中断使能位) 置1，则定时器中断标志置位，发生中断信号，并送到 NVIC 再次通知 CPU。相应 toggle 输出 (tout) 信号置 0。定时器计数操作在 CEN 设置为 0 之后才停止。因此，toggle 输出 (tout) 信号以 50% 的占空比反复改变。所以这种操作模式称为 Toggle 模式。

5.10.4.4 Continuous Counting 模式

如果定时器工作在连续计数模式且 CEN (TCSR[30] 定时器使能位) 置1, 如果 IE (TCSR[29] 中断使能位) 设置为 1, 则相应的中断信号的产生取决于 TDR = TCMR。用户可以立即改变不同的 TCMR 值而不需要禁用定时器计数和重新开始计数。例如, TCMR 一开始为 80 (TCMR 的值必须大于1但小于 2^{24})。首先, 当 TDR 的值等于 80 且 IE 使能, 则 TIF (定时器中断标志) 被置为1且产生中断信号, 并发送到 NVIC 通知 CPU。但是此时 CEN 保持为1 (持续使能计数)而且 TDR 的值也不会回到 0, 它还在继续计数81, 82, 83, … 到 $2^{24}-1$, 0, 1, 2, 3, … 到 $2^{24}-1$ 一次又一次。接下来, 如果用户设置 TCMR 为 200 而且将 TIF 清零, 则当 TDR 达到 200 时, 定时器中断产生且 TIF 被置为1, 中断信号又一次被发送到 NVIC 通知 CPU。最后用户设置 TCMR 为 500 而且将 TIF 清零, 则当 TDR 达到 500 时, 定时器中断产生且 TIF 被置为1, 中断信号再次被发送到 NVIC 通知 CPU。从应用来看, 中断发生取决于 TCMR。在此模式下, 定时器计数是连续的, 因此, 这种操作模式称为 continuous counting 模式。

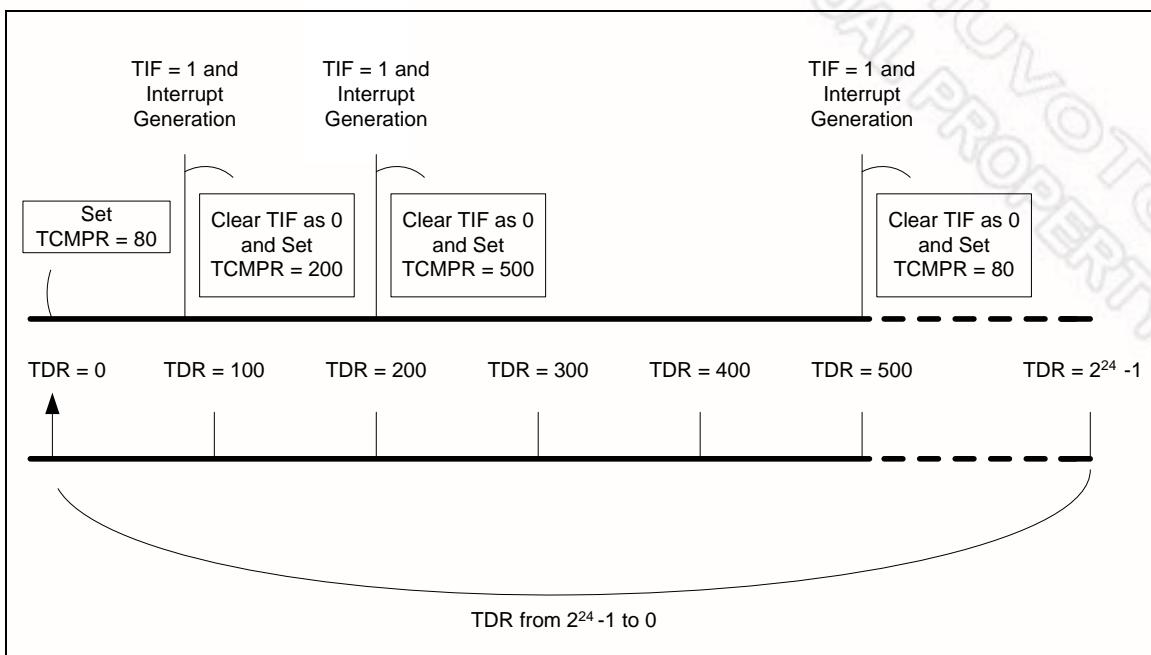


图 5-65 Continuous Counting 模式

5.10.4.5 事件计数功能

定时器控制器提供一个可以对从 TM0~TM3 管脚事件计数的应用, 该模式被称为事件计数模式。在事件计数模式下, 定时器控制器的时钟源 TMRx_CLK, 如图 5-65 所示, 必须设为 HCLK。可以通过设置 TEXCONx[7] 来使能或禁用 TM0~TM3 防抖动功能以及设置 TEXCONx[0] 来设定 TM0~TM3 下降沿或上升沿计数。如果禁用计数防抖动, 事件计数源操作频率必须小于 1/3 HCLK 频率或如果使能计数防抖动, 事件计数源操作频率必须小于 1/8 HCLK 频率, 否则返回的 TDR 值将是不正确的。

5.10.4.6 输入捕捉功能

定时器控制器还提供输入捕捉功能用来捕捉或复位定时器计数值。如果 **TEXEN**（定时器外部管脚使能）设为 1, **RSTCAPn** 设为 0, 当 **TEX**（定时器外部管脚）管脚触发条件发生, 定时器计数值(**TDR**)将被捕捉入 **TCAP** 寄存器。该模块有 4 个 **TEX** 源定义管脚, **T0EX~T3EX** 管脚。如果 **TEXEN** 设为 1, **RSTCAPn** 设为 1, 当 **TEX** 管脚触发条件发生时, **TDR** 将被设为 0。**TEX** 触发边沿由 **TEX_EDGE** 选择。当 **TEX** 触发发生, **TEXIF**（定时器外部中断标志位）被设为 1, 如果使能 **TEXIEN**（定时器外部中断使能位）为 1, 中断信号产生并被发送到 **NVIC** 通知 **CPU**。通过设定 **TEXCONx[6]** 提供 **T0EX~T3EX** 使能或禁用捕捉防抖动功能。如果禁用 **TEX** 防抖动, **TEX** 源操作频率必须小于 1/3 HCLK 频率, 或者如果使能 **TEX** 防抖动功能, 必须小于 1/8 HCLK 频率。

5.10.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
TMR_BA01 = 0x4001_0000				
TMR_BA23 = 0x4011_0000				
TCSR0	TMR_BA01+0x00	R/W	Timer0 控制和状态寄存器	0x0000_0005
TCMPR0	TMR_BA01+0x04	R/W	Timer0 比较寄存器	0x0000_0000
TISR0	TMR_BA01+0x08	R/W	Timer0 中断状态寄存器	0x0000_0000
TDR0	TMR_BA01+0x0C	R	Timer0 数据寄存器	0x0000_0000
TCAP0	TMR_BA01+0x10	R	Timer0 捕捉数据寄存器	0x0000_0000
TEXCON0	TMR_BA01+0x14	R/W	Timer0 外部控制寄存器	0x0000_0000
TEXISR0	TMR_BA01+0x18	R/W	Timer0 外部中断状态寄存器	0x0000_0000
TCSR1	TMR_BA01+0x20	R/W	Timer1 控制和状态寄存器	0x0000_0005
TCMPR1	TMR_BA01+0x24	R/W	Timer1 比较寄存器	0x0000_0000
TISR1	TMR_BA01+0x28	R/W	Timer1 中断状态寄存器	0x0000_0000
TDR1	TMR_BA01+0x2C	R	Timer1 数据寄存器	0x0000_0000
TCAP1	TMR_BA01+0x30	R	Timer1 捕捉数据寄存器	0x0000_0000
TEXCON1	TMR_BA01+0x34	R/W	Timer1 外部控制寄存器	0x0000_0000
TEXISR1	TMR_BA01+0x38	R/W	Timer1 外部中断状态寄存器	0x0000_0000
TCSR2	TMR_BA23+0x00	R/W	Timer2 控制和状态寄存器	0x0000_0005
TCMPR2	TMR_BA23+0x04	R/W	Timer2 比较寄存器	0x0000_0000
TISR2	TMR_BA23+0x08	R/W	Timer2 中断状态寄存器	0x0000_0000
TDR2	TMR_BA23+0x0C	R	Timer2 数据寄存器	0x0000_0000
TCAP2	TMR_BA23+0x10	R	Timer2 捕捉数据寄存器	0x0000_0000
TEXCON2	TMR_BA23+0x14	R/W	Timer2 外部控制寄存器	0x0000_0000
TEXISR2	TMR_BA23+0x18	R/W	Timer2 外部中断状态寄存器	0x0000_0000
TCSR3	TMR_BA23+0x20	R/W	Timer3 控制和状态寄存器	0x0000_0005
TCMPR3	TMR_BA23+0x24	R/W	Timer3 比较寄存器	0x0000_0000
TISR3	TMR_BA23+0x28	R/W	Timer3 中断状态寄存器	0x0000_0000
TDR3	TMR_BA23+0x2C	R	Timer3 数据寄存器	0x0000_0000



TCAP3	TMR_BA23+0x30	R	Timer3 捕捉数据寄存器	0x0000_0000
TEXCON3	TMR_BA23+0x34	R/W	Timer3 外部控制寄存器	0x0000_0000
TEXISR3	TMR_BA23+0x38	R/W	Timer3 外部中断状态寄存器	0x0000_0000



5.10.6 寄存器描述

定时器控制寄存器 (TCSR)

寄存器	偏移量	R/W	描述	复位后的值
TCSR0	TMR_BA01+0x00	R/W	Timer0 控制和状态寄存器	0x0000_0005
TCSR1	TMR_BA01+0x20	R/W	Timer1 控制和状态寄存器	0x0000_0005
TCSR2	TMR_BA23+0x00	R/W	Timer2 控制和状态寄存器	0x0000_0005
TCSR3	TMR_BA23+0x20	R/W	Timer3 控制和状态寄存器	0x0000_0005

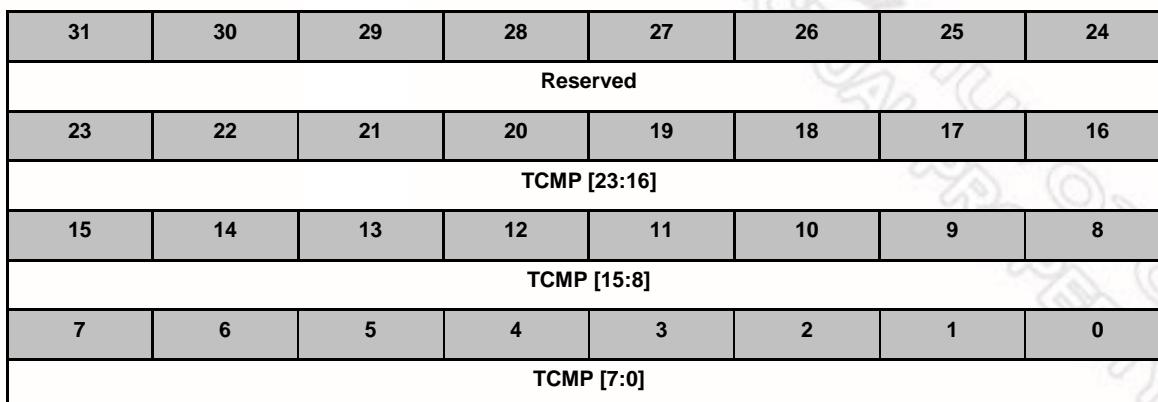
31	30	29	28	27	26	25	24
DBGACK_TMR	CEN	IE	MODE[1:0]		CRST	CACT	CTB
23	22	21	20	19	18	17	16
Reserved							TDR_EN
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PRESCALE[7:0]							

Bits	描述
[31]	DBGACK_TMR 仿真器 (ICE) 调试模式响应禁止 (写保护位) 0 = 仿真器调试模式响应影响定时器计数。 当调试器调试模式响应时，定时器计数器将被固定住。 1 = 仿真器调试模式响应禁用。 无论调试器调试模式响应与否，定时器计数器将持续计数下去。
[30]	CEN 定时器使能位 1 = 开始计数 0 = 停止/暂停计数 注1：在停止状态，设置 CEN 为 1 将使能 24-位向上计数器从上次停止的计数值继续计数。 注2：在 one-shot 模式下 (MODE [28:27]=00)，当相应的定时中断产生时 (IE [29]=1)，该位由硬件自动清零。
[29]	IE 中断使能位 1 = 使能定时器中断 0 = 禁用定时器中断 当定时器中断使能，当计数值与TCMPR寄存器内数值相同时，触发中断。

		定时器工作模式
	模式	定时器工作模式
[28:27]	MODE	00 当定时器定义为单触发模式 (<i>one-shot</i>) 时，定时器溢出仅触发中断一次（如果 IE 使能），进入中断后 CEN 由硬件自动清除为 0。
		01 当定时器定义为周期模式 (<i>period</i>) 时，定时器每次溢出都触发相应中断（如果 IE 使能）。
		10 当定时器工作在 <i>toggle</i> 模式，中断信号周期性产生（如果 IE 使能）。且相应的信号 (tout) 以占空比为 50% 周期改变。
		11 定时器工作在连续计数 (<i>continuous counting</i>) 模式。相应的中断在 TDR = TCMR 时产生（如果 IE 使能）。然而，24位的向上定时器继续计数而不会复位。详细描述请参考章节 5.10.4.4。
[26]	CRST	定时器复位位 设置该位将重置 24-位向上计数器，8-位预分频计数器以及使 CEN 为 0。 0 = 该位写 0 无效。 1 = 重置定时器的 8-位预分频计数器，内部 24-位向上计数器和 CEN 位。
[25]	CACT	定时器激活状态位 (只读) 该位表示当前定时器计数器的状态。 0 = 定时器未激活 1 = 定时器激活
[24]	CTB	计数模式使能位 该位是计数模式使能位。当定时器用作事件计数器时，该位需要设置成 1 且定时器作为事件计数器外部触发引脚。计数器根据 TX_PHASE 定义的相位在外部管脚的上升沿/下降沿检测相位。 1 = 使能计数器模式 0 = 禁用计数器模式
[23:17]	Reserved	保留
[16]	TDR_EN	数据载入使能 当 TDR_EN 被置位后，计数器运行时，TDR (Timer 数据寄存器) 将被 24-位 向上计数器的值不断更新。 1 = Timer 数据寄存器更新使能。 0 = Timer 数据寄存器禁用更新。
[15:8]	Reserved	保留
[7:0]	PREScale	预分频计数器 时钟输入根据 PREScale +1 进行预分频。如果 PREScale 数值为0，不进行预分频。

定时器比较寄存器 (TCMPR)

寄存器	偏移量	R/W	描述	复位后的值
TCMPR0	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]	Reserved	保留
[23:0]	TCMP	<p>定时器比较值</p> <p>TCMP 是24位比较寄存器。当内部 24位向上计数器的值与 TCMP 的值相等时，如果 TCSR.IE[29]=1，就产生定时器中断请求。TCMP 的值为定时器计数周期。</p> <p>超时周期 = (定时器时钟输入周期) * (8-bit PRESCALE + 1) * (24-bit TCMP)</p> <p>注1：不能向 TCMP 里写 0x0 或 0x1，否则内核将运行到未知状态。</p> <p>注2：当定时器工作在 continuous counting 模式时，如果软件写一个新的值到 TCMP，24位向上计数定时器将继续计数。如果定时器工作在其他模式，如果软件写一个新的值到 TCMP，定时器将使用新比较值并退出当前计数，开始重新计数。</p>

定时器中断状态寄存器 (TISR)

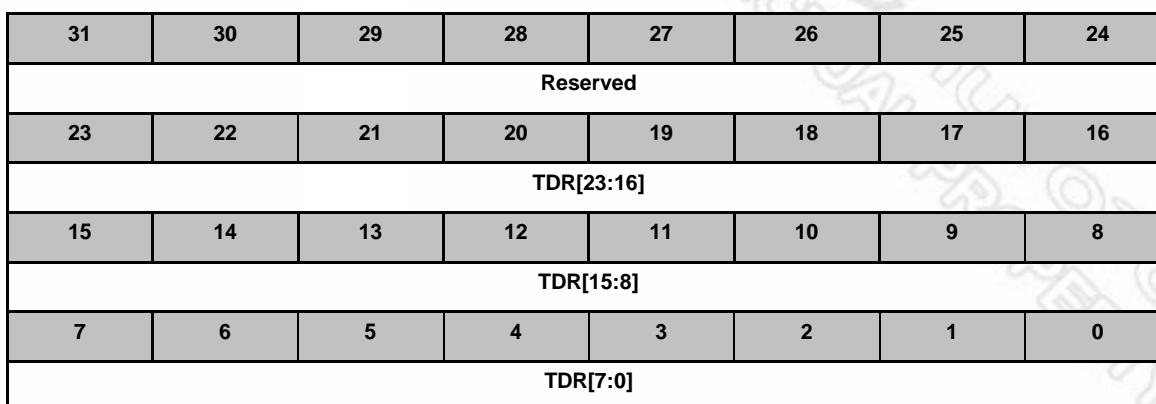
寄存器	偏移量	R/W	描述	复位后的值
TISR0	TMR_BA01+0x08	R/W	Timer0 中断状态寄存器	0x0000_0000
TISR1	TMR_BA01+0x28	R/W	Timer1 中断状态寄存器	0x0000_0000
TISR2	TMR_BA23+0x08	R/W	Timer2 中断状态寄存器	0x0000_0000
TISR3	TMR_BA23+0x28	R/W	Timer3 中断状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	Reserved	保留
[0]	TIF	定时器中断标志位 定时器中断状态标志位。 当内部 24-位向上计数定时器与定时器比较值 (TCMP)，TIF 位由硬件置位。该位写 1 清零。

定时器数据寄存器 (TDR)

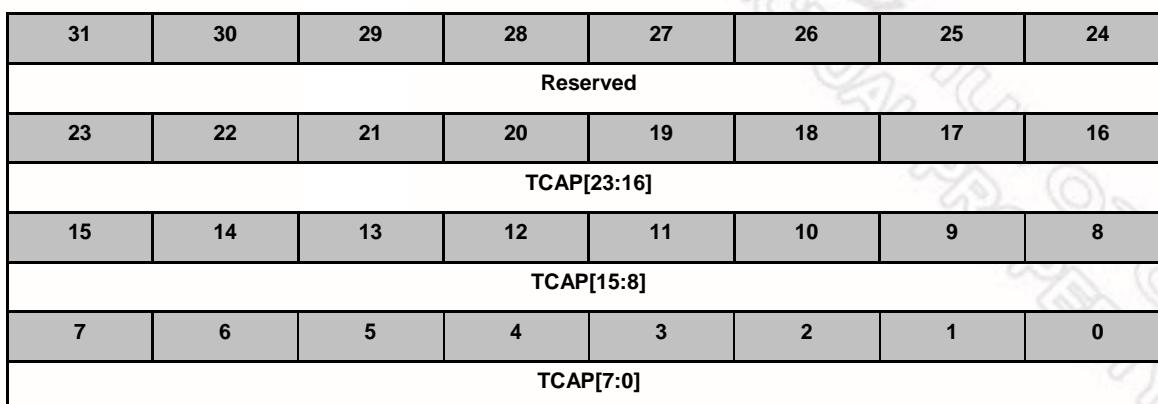
寄存器	偏移量	R/W	描述	复位后的值
TDR0	TMR_BA01+0x0C	R	Timer0 数据寄存器	0x0000_0000
TDR1	TMR_BA01+0x2C	R	Timer1 数据寄存器	0x0000_0000
TDR2	TMR_BA23+0x0C	R	Timer2 数据寄存器	0x0000_0000
TDR3	TMR_BA23+0x2C	R	Timer3 数据寄存器	0x0000_0000



Bits	描述	
[31:24]	Reserved	保留
[23:0]	TDR	<p>定时器数据寄存器</p> <p>1. CTB (TCSR[24]) = 0 : TDR 是 24-位向上计数定时器的值。</p> <p>如果 TCSR[24] 设为 0, 用户可以通过读 TDR 获取当前 24-位向上计数定时器的值。</p> <p>2. CTB (TCSR[24]) = 1 : TDR is 24-位 向上计数事件计数器的值。</p> <p>如果 TCSR[24] 设为1, 用户可以通过读 TDR 获取当前 24-位向上计数事件计数器的值。</p>

定时器捕捉数据寄存器 (TCAP)

寄存器	偏移量	R/W	描述	复位后的值
TCAP0	TMR_BA01+0x10	R	Timer0 捕捉数据寄存器	0x0000_0000
TCAP1	TMR_BA01+0x30	R	Timer1 捕捉数据寄存器	0x0000_0000
TCAP2	TMR_BA23+0x10	R	Timer2 捕捉数据寄存器	0x0000_0000
TCAP3	TMR_BA23+0x30	R	Timer3 捕捉数据寄存器	0x0000_0000



Bits	描述	
[31:24]	Reserved	保留
[23:0]	TCAP	定时器捕捉数据寄存器 当 TEXEN (TEXCON[3]) 被置位， RSTCAPn(TTXCON[4]) 为 0， TEX_EDGE(TEXCON[2:1]) 设置对应的 TEX 管脚上的变化发生时，内部 24-位向上计数定时器的值将被载入到 TCAP。用户可以读该寄存器获得计数器的值。

定时器外部控制寄存器 (TEXCON)

寄存器	偏移量	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	RSTCAPn	TEXEN	TEX_EDGE	TX_PHASE	

Bits	描述	
[31:8]	Reserved	保留
[7]	TCDB	<p>定时器计数器管脚防抖动使能位 1 = 使能防抖动 0 = 禁用防抖动 如果该位使能, TM0~TM3 管脚边沿检测带防抖动电路。</p>
[6]	TEXDB	<p>定时器外部捕捉管脚防抖动使能位 1 = 使能防抖动 0 = 禁用防抖动 如果该位使能, T0EX~T3EX 管脚边沿检测带防抖动电路。</p>
[5]	TEXIEN	<p>定时器外部中断使能位 1 = 使能定时器外部中断 0 = 禁用定时器外部中断 如果定时器外部中断使能, 当 TEX_EDGE(TEXCON[2:1]) 设定的对应的 TEX 管脚上的变化发生时, 定时器产生外部中断信号, 发生到 NVIC 通知 CPU。 例如, 当 TEXIEN = 1, TEXEN = 1, TEX_EDGE = 00, TEX 管脚上1 到 0的变化将导致 TEXIF(TEXISR[0]) 中断标志位被置位, 然后中断信号产生并被发生到 NVIC 通知 CPU。</p>
[4]	RSTCAPn	定时器外部复位计数器/捕捉模式选择

		1 = TEX 变化用来作为定时器计数器复位功能。 0 = TEX 变化用来作为定时器捕捉功能。
[3]	TEXEN	定时器外部管脚使能 该位使能 TEX 管脚上的复位/捕捉功能。 1 = TEX 管脚上的变化将导致定时器捕捉或定时器复位。 0 = TEX 管脚将被忽略。
[2:1]	TEX_EDGE	定时器外部管脚边沿检测 00 = TEX 管脚上 1 到 0 的变化将被检测。 01 = TEX 管脚上 0 到 1 的变化将被检测。 10 = TEX 管脚上 1 到 0 或 0 到 1 的变化将被检测。 11 = 保留。
[0]	TX_PHASE	定时器外部计数相位 该位表示外部计数管脚相位。 1 = 外部计数管脚的上升沿将被统计。 0 = 外部计数管脚的下降沿将被统计。

定时器外部中断状态寄存器 (TISR)

寄存器	偏移量	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]	Reserved	保留
[0]	TEXIF	<p>定时器外部中断标志位 该位表示定时器外部中断状态。</p> <p>当 TEXEN (TEXCON[3]) 为 1, EX_EDGE (TEXCON[2:1]) 设定对应的 TEX 管脚上变化发生时, 该位被硬件置位。该位写 1 清零。</p> <p>例如, 当 TEXEN = 1, TEX_EDGE = 00, 且 TEX 管脚上从 1 变到 0 将导致 TEXIF 被置位。</p>

5.11 看门狗定时器 (WDT)

5.11.1 概述

看门狗定时器的用途是在软件出问题时执行系统复位功能，可以防止系统无限止地挂机，除此之外，看门狗定时器还可将芯片由掉电模式唤醒。看门狗定时器包含一个18位的自动运行的计数器，可编程其定时溢出间隔。表 5-7 为看门狗定时溢出间隔选择，图 5-67 为看门狗中断信号与复位信号的时序。

设置 WTE(WDTCR[7]) 使能看门狗定时器和 WDT 计数器开始计数。当计数器达到选择的定时溢出间隔，看门狗定时器中断标志 WTIF 将被立即置位，并请求 WDT 中断（如果看门狗定时器中断使能位 WTIE 置位），同时紧接着会有一个指定周期 ($1024 * T_{WDT}$) 的延时。用户必须在指定周期内设置 WTR (WDTCR[0])（看门狗定时器复位）为高，重置18位 WDT 计数器，防止芯片复位。WTR 位在 WDT 计数重置后自动由硬件清零。通过设置 WTIS (WDTCR[10:8]) 有8个定时溢出间隔可选择。如果在指定延迟时间终止后，WDT 计数没有被清零，看门狗定时器将置看门狗定时器复位标志 (WTRF) 为高并使芯片复位。这个复位将持续 63 个 WDT 时钟 (T_{RST})，然后芯片重启，并从复位向量 (0x0000_0000) 处执行程序。WTRF 将不被看门狗复位清零。用户可用软件轮询 WTRF 识别复位源。WDT 还提供唤醒功能，当芯片在掉电状态，且看门狗定时器唤醒功能使能位 (WDTR[4]) 置位，如果 WDT 计数器达到了由 WTIS (WDTCR [10:8]) 定义的指定时间间隔，芯片将由掉电状态唤醒。例1，如果 WTIS 被置为 000，则芯片从掉电状态唤醒的指定时间间隔为 $2^4 * T_{WDT}$ 。当软件设定掉电命令时，然后芯片进入掉电状态。在 $2^4 * T_{WDT}$ 时间过去后，芯片从掉电状态唤醒。例2，如果 WTIS (WDTCR [10:8]) 被置为 111，则芯片从掉电状态唤醒的指定时间间隔为 $2^{18} * T_{WDT}$ 。当软件设定掉电命令时，然后芯片进入掉电状态。在 $2^{18} * T_{WDT}$ 时间过去后，芯片从掉电状态唤醒。注意，如果 WTRE (WDTCR [1]) 为 1，一旦芯片被唤醒，就必须通过设定 WTR (WDTCR [0]) 为 1 清除看门狗定时器计数。否则，如果在芯片被唤醒开始到软件清除看门狗定时器计数时间超过 $1024 * T_{WDT}$ ，没有通过设置 WTR (WDTCR [0]) 为 1 清除看门狗定时器计数，芯片将被看门狗定时器复位。

WTIS	Timeout Interval Selection T_{TIS}	Interrupt Period T_{INT}	WTR Timeout startingInterval (WDT_CLK=10 kHz) MIN. T_{WTR} ~ Max. T_{WTR}
000	$2^4 * T_{WDT}$	$1024 * T_{WDT}$	1.6 ms ~ 104 ms
001	$2^6 * T_{WDT}$	$1024 * T_{WDT}$	6.4 ms ~ 108.8 ms
010	$2^8 * T_{WDT}$	$1024 * T_{WDT}$	25.6 ms ~ 128 ms
011	$2^{10} * T_{WDT}$	$1024 * T_{WDT}$	102.4 ms ~ 204.8 ms
100	$2^{12} * T_{WDT}$	$1024 * T_{WDT}$	409.6 ms ~ 512 ms
101	$2^{14} * T_{WDT}$	$1024 * T_{WDT}$	1.6384 s ~ 1.7408 s
110	$2^{16} * T_{WDT}$	$1024 * T_{WDT}$	6.5536 s ~ 6.656 s
111	$2^{18} * T_{WDT}$	$1024 * T_{WDT}$	26.2144 s ~ 26.3168 s

表 5-7 看门狗定时溢出间隔选择

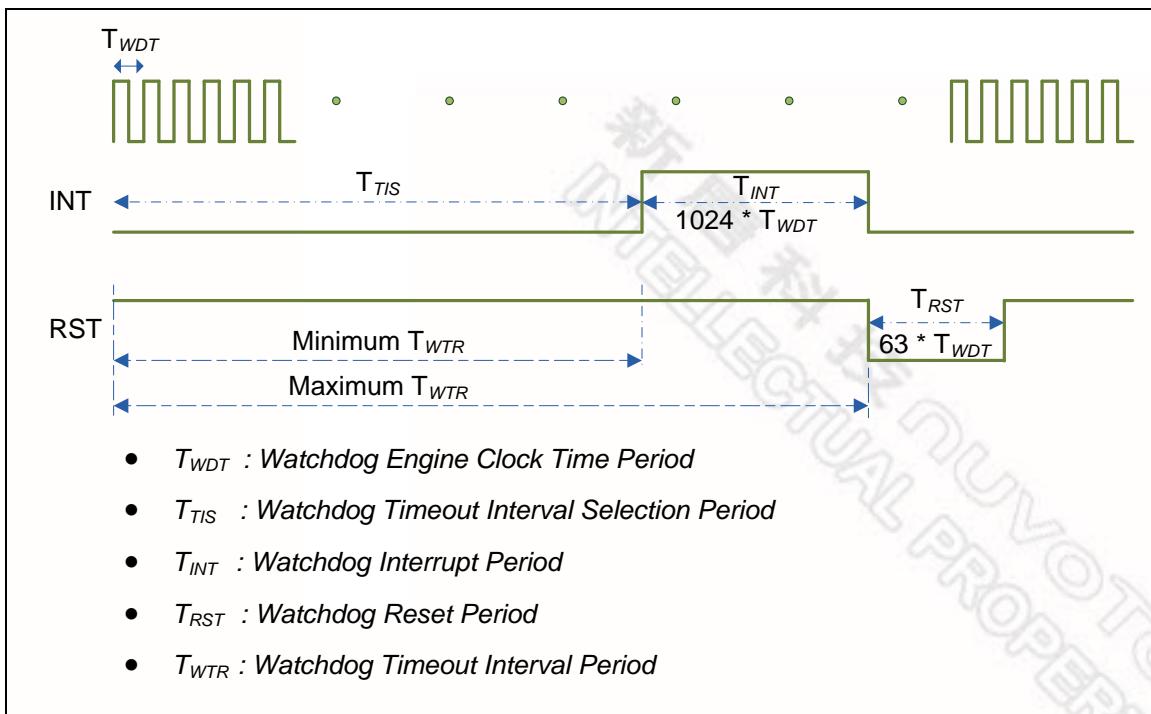


图 5-66 中断和复位信号时序

5.11.2 特征

- 18-位计数器避免芯片跑飞。
- 可选择的定时溢出间隔 ($2^{14} \sim 2^{18}$)，定时溢出间隔为 104 ms ~ 26.3168 s (如果 WDT_CLK = 10 kHz)。
- 复位周期 = $(1 / 10 \text{ kHz}) * 63$, 如果 WDT_CLK = 10 kHz。

5.11.3 框图

看门狗定时器时钟控制和框图如下所示。

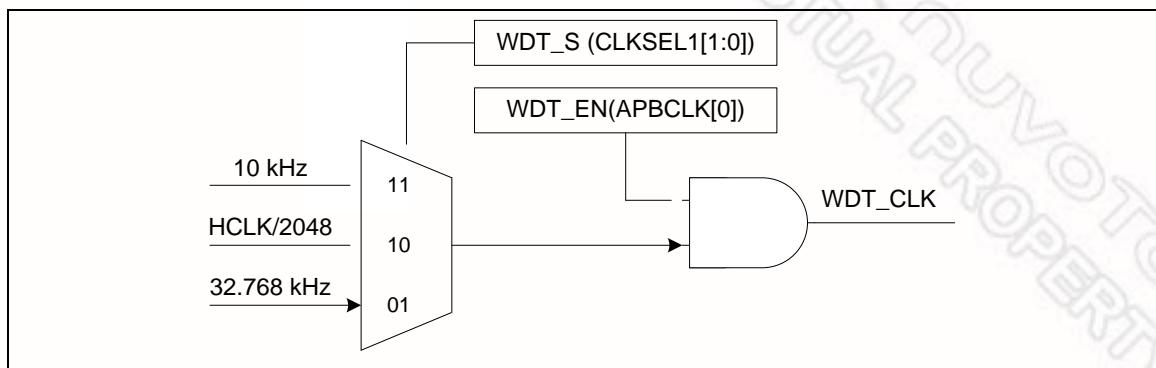


图 5-67 看门狗定时器时钟控制

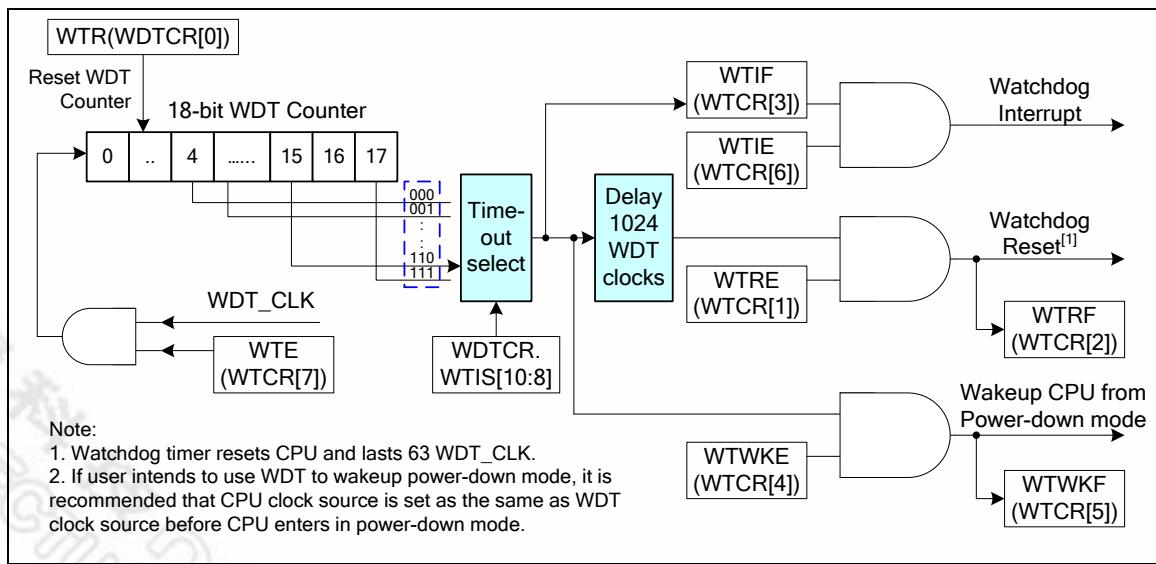


图 5-68 看门狗定时器框图



5.11.4 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
WDT_BA = 0x4000_4000				
WTCR	WDT_BA+0x00	R/W	看门狗定时器控制寄存器	0x0000_0700



5.11.5 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

注：该寄存器所有位受保护。编程该位是需要依次向地址 0x5000_0100 写入“59h”，“16h”，“88h”来解除保护。参考寄存器 REGWRPROT（地址 GCR_BA+0x100）。

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

Bits	描述																																				
[31]	DBGACK_WDT 仿真器 (ICE) 调试模式响应禁止 (写保护位) 0 = 仿真器调试模式响应影响看门狗定时器计数。 当调试器调试模式响应时，看门狗定时器计数器将被固定住。 1 = 仿真器调试模式响应禁用。 无论调试器调试模式响应与否，看门狗定时器计数器将持续计数下去。																																				
[30:11]	Reserved 保留																																				
[10:8]	WTIS 看门狗定时器间隔选择 (写保护位) 这些位选择看门狗定时器的定时溢出间隔。 <table border="1" style="margin-left: 20px;"> <tr> <th>WTIS</th> <th>定时溢出间隔</th> <th>中断周期</th> <th>WTR 定时溢出间隔 (WDT_CLK=10 kHz)</th> </tr> <tr> <td>000</td> <td>$2^4 * T_{WDT}$</td> <td>$(2^4 + 1024) * T_{WDT}$</td> <td>1.6 ms ~ 104 ms</td> </tr> <tr> <td>001</td> <td>$2^6 * T_{WDT}$</td> <td>$(2^6 + 1024) * T_{WDT}$</td> <td>6.4 ms ~ 108.8 ms</td> </tr> <tr> <td>010</td> <td>$2^8 * T_{WDT}$</td> <td>$(2^8 + 1024) * T_{WDT}$</td> <td>25.6 ms ~ 128 ms</td> </tr> <tr> <td>011</td> <td>$2^{10} * T_{WDT}$</td> <td>$(2^{10} + 1024) * T_{WDT}$</td> <td>102.4 ms ~ 204.8 ms</td> </tr> <tr> <td>100</td> <td>$2^{12} * T_{WDT}$</td> <td>$(2^{12} + 1024) * T_{WDT}$</td> <td>409.6 ms ~ 512 ms</td> </tr> <tr> <td>101</td> <td>$2^{14} * T_{WDT}$</td> <td>$(2^{14} + 1024) * T_{WDT}$</td> <td>1.6384 s ~ 1.7408 s</td> </tr> <tr> <td>110</td> <td>$2^{16} * T_{WDT}$</td> <td>$(2^{16} + 1024) * T_{WDT}$</td> <td>6.5536 s ~ 6.656 s</td> </tr> <tr> <td>111</td> <td>$2^{18} * T_{WDT}$</td> <td>$(2^{18} + 1024) * T_{WDT}$</td> <td>26.2144 s ~ 26.3168 s</td> </tr> </table>	WTIS	定时溢出间隔	中断周期	WTR 定时溢出间隔 (WDT_CLK=10 kHz)	000	$2^4 * T_{WDT}$	$(2^4 + 1024) * T_{WDT}$	1.6 ms ~ 104 ms	001	$2^6 * T_{WDT}$	$(2^6 + 1024) * T_{WDT}$	6.4 ms ~ 108.8 ms	010	$2^8 * T_{WDT}$	$(2^8 + 1024) * T_{WDT}$	25.6 ms ~ 128 ms	011	$2^{10} * T_{WDT}$	$(2^{10} + 1024) * T_{WDT}$	102.4 ms ~ 204.8 ms	100	$2^{12} * T_{WDT}$	$(2^{12} + 1024) * T_{WDT}$	409.6 ms ~ 512 ms	101	$2^{14} * T_{WDT}$	$(2^{14} + 1024) * T_{WDT}$	1.6384 s ~ 1.7408 s	110	$2^{16} * T_{WDT}$	$(2^{16} + 1024) * T_{WDT}$	6.5536 s ~ 6.656 s	111	$2^{18} * T_{WDT}$	$(2^{18} + 1024) * T_{WDT}$	26.2144 s ~ 26.3168 s
WTIS	定时溢出间隔	中断周期	WTR 定时溢出间隔 (WDT_CLK=10 kHz)																																		
000	$2^4 * T_{WDT}$	$(2^4 + 1024) * T_{WDT}$	1.6 ms ~ 104 ms																																		
001	$2^6 * T_{WDT}$	$(2^6 + 1024) * T_{WDT}$	6.4 ms ~ 108.8 ms																																		
010	$2^8 * T_{WDT}$	$(2^8 + 1024) * T_{WDT}$	25.6 ms ~ 128 ms																																		
011	$2^{10} * T_{WDT}$	$(2^{10} + 1024) * T_{WDT}$	102.4 ms ~ 204.8 ms																																		
100	$2^{12} * T_{WDT}$	$(2^{12} + 1024) * T_{WDT}$	409.6 ms ~ 512 ms																																		
101	$2^{14} * T_{WDT}$	$(2^{14} + 1024) * T_{WDT}$	1.6384 s ~ 1.7408 s																																		
110	$2^{16} * T_{WDT}$	$(2^{16} + 1024) * T_{WDT}$	6.5536 s ~ 6.656 s																																		
111	$2^{18} * T_{WDT}$	$(2^{18} + 1024) * T_{WDT}$	26.2144 s ~ 26.3168 s																																		

[7]	WTE	看门狗定时器使能（写保护位） 0 = 禁用看门狗定时器（该动作将重置内部计数器） 1 = 使能看门狗定时器
[6]	WTIE	看门狗中断使能（写保护位） 0 = 禁用看门狗中断 1 = 使能看门狗中断
[5]	WTWKF	看门狗唤醒标志位 如果看门狗定时器引起芯片从掉电模式下唤醒，该位将被置高。必须由软件向该位写1清零。 0 = 看门狗定时器不能引起芯片唤醒。 1 = 芯片由空闲或掉电模式被看门狗定时溢出唤醒。
[4]	WTWKE	看门狗定时器唤醒功能使能位（写保护位） 0 = 禁用看门狗唤醒芯片功能 1 = 使能看门狗唤醒功能，看门狗定时溢出能够将芯片从掉电模式唤醒。 注：只有 WDT 的时钟源为 RC10K，芯片才能被 WDT 唤醒。
[3]	WTIF	看门狗定时器中断标志 如果看门狗定时器中断使能，该位置位表示看门狗定时器中断发生。 0 = 看门狗定时器中断没有发生。 1 = 看门狗定时器中断发生。 注：该位写1清除。
[2]	WTRF	看门狗定时器复位标志位 当看门狗定时器溢出引发复位，硬件将置位该位，通过读取该位可以确认复位是否由看门狗引起。该位通过写1手动清除，如果 WTRE 禁用，看门狗定时器溢出对该位无影响。 0 = 看门狗定时器服务没发生。 1 = 看门狗定时器复位发生。 注：该位写1清除。
[1]	WTRE	看门狗定时器复位使能（写保护位） 置该位将使能看门狗定时器复位功能。 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。
[0]	WTR	清看门狗定时器（写保护位） 置该位将清除看门狗定时器。 0 = 写0到该位无效果。 1 = 重置看门狗定时器。 注：该位将被硬件自动清除。

5.12 UART 接口控制器 (UART)

NuMicro™ NUC130/NUC140 提供最多 3 个通用异步收发器 (UART) 通道。UART0 支持高速 UART，UART1~2 支持普通速度 UART，此外，只有 UART0 和 UART1 支持流控制功能。

5.12.1 概述

通用异步收发器 (UART) 在从外设收到数据的时候执行串行到并行的转换，从 CPU 发送数据的时候执行并行到串行的转换。该串口控制器同时支持 IrDA SIR 功能，LIN 主机/从机模式功能和 RS-485 模式功能。每个 UART 通道支持 7 种类型的中断，包括发送FIFO 空中断 (INT_THRE)，接收阈值到达中断 (INT_RDA)，线状态中断（奇偶校验错误，格式错误或者break 中断）(INT_RLS) , 接收缓存超时中断 (INT_TOUT)，MODEM/唤醒状态中断 (INT_MODEM)，缓冲错误中断 (INT_BUF_ERR) 和 LIN 接收间断域检测中断 (INT_LIN_RX_BREAK)。Interrupts of UART0 和 UART2 共享中断号12 (向量号为 28)；中断号 13 (向量号为 29) 仅支持 UART1 中断。参考 NVIC 章节对系统中断的描述。

UART0 内嵌一个 64-位 发送 FIFO (TX_FIFO) 和一个 64-位接收 FIFO (RX_FIFO) 来降低 CPU 的中断出现的数量，UART1~2 内嵌一个 16-位 发送 FIFO (TX_FIFO) 和一个 16-位接收 FIFO (RX_FIFO)。在操作过程中 CPU 可以随时读 UART 的状态。报告的状态信息包括已经被 UART 执行的传输操作的类型和条件，也包括当接收数据可能发生的 4 种错误条件 (parity error, framing error, break interrupt 和 buffer error)。UART 包括一个可编程的波特率发生器，它可以将输入晶振除以一个除数来得到收发器需要的串行时钟。波特率公式为波特率 = $UART_CLK / M * [BRD + 2]$ ，其中 M 和 BRD 在波特率分频寄存器 (UA_BAUD) 中定义。表 5-8 列举了不同变量条件下的等式，表 5-9 为 UART 波特率设置表。

Mode	DIV_X_EN	DIV_X_ONE	Divider X	BRD	波特率公式
0	0	0	Don't care	A	$UART_CLK / [16 * (A+2)]$
1	1	0	B	A	$UART_CLK / [(B+1) * (A+2)]$, B must >= 8
2	1	1	Don't care	A	$UART_CLK / (A+2)$, A must >=7

表 5-8 UART 波特率公式

System clock = internal 22.1184 MHz high speed oscillator						
Baud rate	Mode0		Mode1		Mode2	
	Parameter	Register	Parameter	Register	Parameter	Register
921600	x	x	A=0,B=11	0x2B00_0000	A=22	0x3000_0016
460800	A=1	0x0000_0001	A=1,B=15 A=2,B=11	0x2F00_0001 0x2B00_0002	A=46	0x3000_002E
230400	A=4	0x0000_0004	A=4,B=15 A=6,B=11	0x2F00_0004 0x2B00_0006	A=94	0x3000_005E
115200	A=10	0x0000_000A	A=10,B=15 A=14,B=11	0x2F00_000A 0x2B00_000E	A=190	0x3000_00BE

57600	A=22	0x0000_0016	A=22,B=15 A=30,B=11	0x2F00_0016 0x2B00_001E	A=382	0x3000_017E
38400	A=34	0x0000_0022	A=62,B=8 A=46,B=11 A=34,B=15	0x2800_003E 0x2B00_002E 0x2F00_0022	A=574	0x3000_023E
19200	A=70	0x0000_0046	A=126,B=8 A=94,B=11 A=70,B=15	0x2800_007E 0x2B00_005E 0x2F00_0046	A=1150	0x3000_047E
9600	A=142	0x0000_008E	A=254,B=8 A=190,B=11 A=142,B=15	0x2800_00FE 0x2B00_00BE 0x2F00_008E	A=2302	0x3000_08FE
4800	A=286	0x0000_011E	A=510,B=8 A=382,B=11 A=286,B=15	0x2800_01FE 0x2B00_017E 0x2F00_011E	A=4606	0x3000_11FE

表 5-9 UART 波特率设置表

UART0 和 UART1 控制器用 2 种低电平信号支持自动流控制功能, /CTS (clear-to-send) 和 /RTS (request-to-send), 用来控制 UART 和外部设备 (如: Modem) 之间的数据流传输。当使能自动流控制时, UART 将不允许接收数据直到 UART 向外部设备发送 /RTS。当 RX FIFO 内地字节数等于 RTS_TRI_LEVEL (UA_FCR [19:16]) 的值, /RTS 信号停止。当 UART 控制器从外部设备侦测到 /CTS 信号时, 向外发送数据。如果 /CTS 未被探测到, UART 将不向外发送数据。

UART 控制器提供串行 IrDA (SIR, 串行红外) 功能 (用户需置位 IrDA_EN (UA_FUN_SEL [1]) 使能 IrDA 功能)。SIR 定义短距离红外异步串行传输模式, 该模式有 1 个开始位, 8 个 数据位和 1 个停止位。最大数据速率 为 115.2 Kbps (半双工)。IrDA SIR 模块包括 IrDA SIR 协议编码/解码器。仅具有 IrDA SIR 半双工协议, 所以不能同时传输和接收数据。IrDA SIR 物理层规定在传输和接收之间至少 10ms 输出延时。该特性由软件执行。

UART 控制器的另一个可选功能是 LIN (Local Interconnect Network) 功能。通过设置 UA_FUN_SEL 寄存器的 LIN_EN 位进行选择。在 LIN 模式下, 依照 LIN 标准需要 1 个开始位, 8 个数据位和 1 个停止位。

NuMicro™ NUC100 系列, UART 控制器的另一个可选功能是 RS-485 9-位模式功能, 由 RTS 脚控制方向或由软件编程 GPIO (PB.2 for RTS0 and PB.6 for RTS1) 执行该功能。RS-485 模式由设置寄存器 UA_FUN_SEL 来选择。RS-485 驱动控制通过使用异步串行端口的 RTS 控制信号使能 RS-485 驱动。在 RS-485 模式, RX 和 TX 的许多特性与 UART 相同。



5.12.2 特征

- 全双工，异步通信
- 独立接收/发送 64/16/16 字节 (UART0/UART1/UART2) FIFO，用于数据装载。
- 支持硬件自动流控制/流控制功能 (CTS, RTS) 和可编程的 RTS 流控制触发电平 (UART0 和 UART1 支持)
- 可编程的接收缓冲触发极限值
- 每个通道独立的可编程的波特率发生器
- 支持 CTS 唤醒功能 (UART0 and UART1 支持)
- 支持 7-位接收缓存定时溢出检测功能
- UART0/UART1 可以采用 DMA 控制器
- 在上一次的停止位与下一次的开始位之间通过设置寄存器 UA_TOR [DLY] 可编程发送数据延迟时间
- 支持 break error, frame error, parity error 和 receive/transmit 缓存溢出检测功能。
- 完全可编程的串口特性
 - 可编程为 5-, 6-, 7-, 8-位的数据位
 - 可编程的奇偶校验位, even, odd, no parity 或 stick parity bit 产生和侦测
 - 可编程为 1, 1.5, 或 2 位的停止位
- 支持 IrDA SIR 功能模式
 - 普通模式下支持 3-/16-位时间
- 支持 LIN 功能模式
 - 支持 LIN 主机/从机模式
 - 发送器支持可编程的 break 产生功能
 - 接收器支持 break 侦测功能
- 支持 RS-485 功能模式
 - 支持 RS-485 9-位模式
 - 由 RTS 管脚提供支持硬件或软件直接使能控制

5.12.3 框图

UART 时钟控制和框图见 图 5-70 和 图 5-71。

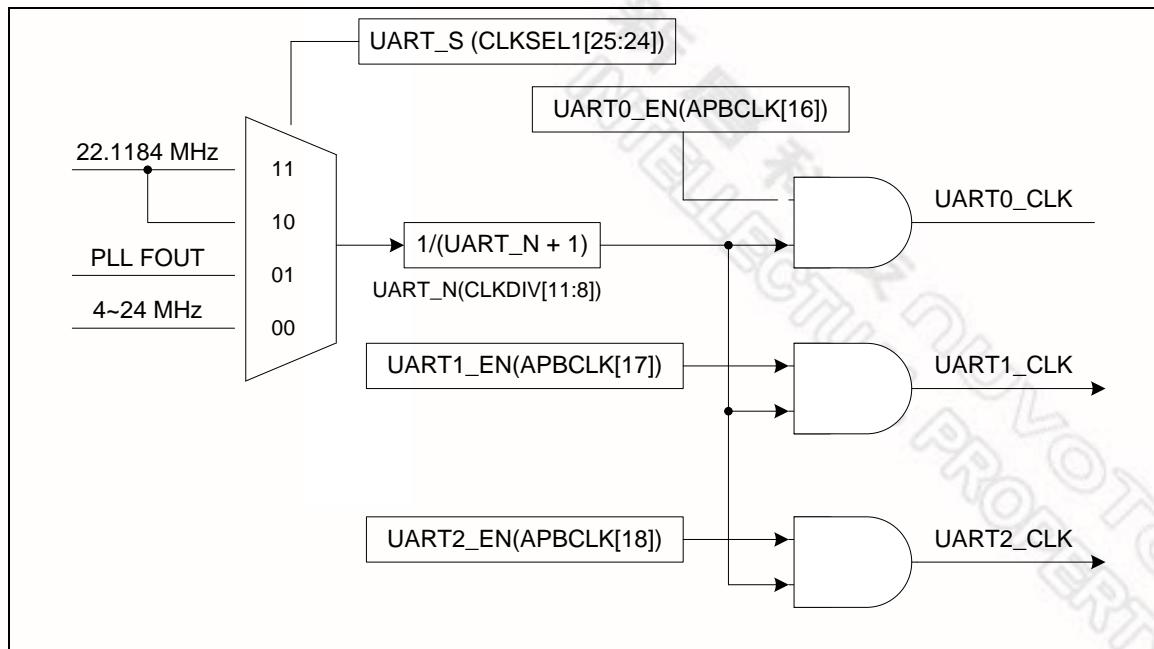


图 5-69 UART 时钟控制图

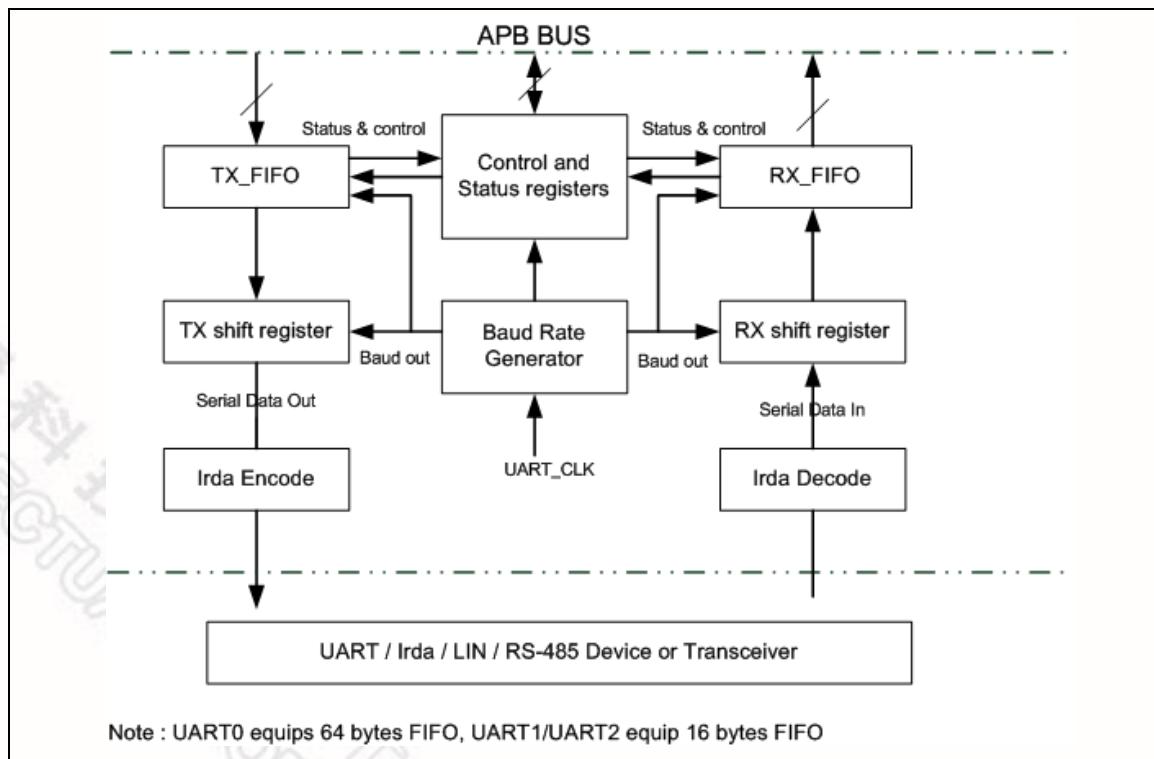


图 5-70 UART 框图



TX_FIFO

发送器用一个 64/16 字节的 FIFO 缓存来降低 CPU 的中断数量。

RX_FIFO

接收器用一个 64/16 字节的 FIFO (每个字节加 3 个比特的纠错位) 缓存来降低 CPU 的中断数量

TX 移位寄存器

移位发送数据到数据串行输出控制模块

RX 移位寄存器

控制将接收到的数据移入串行移位模块

Modem 控制寄存器

这个寄存器控制 到 MODEM 的接口或数据集 (一个模拟 MODEM 外设)

波特率发生器

将外部时钟除以一个除数来产生期望的波特率时钟。参考波特率公式。

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

此寄存器设定，包括 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) 用于发送和接收。时间超时控制寄存器 (UA_TOR) 应用于识别时间超时中断。该寄存器也包括中断使能寄存器 (UA_IER) 和中断状态寄存器 (UA_ISR) 来使能或禁用中断响应并且识别发生的中断。有 七种类型的中断，发送器 FIFO 空中断 (INT_THRE)，接收器达到阈值中断 (INT_RDA)，线状态中断 (parity error 或 framing error 或 break interrupt) (INT_RLS)，定时溢出中断 (INT_TOUT)，MODEM/唤醒状态中断 (INT_MODEM)，缓存错误中断 (INT_BUF_ERR) 和 LIN 接收器 break 域检测中断 (INT_LIN_RX_BREAK)。

下图为自动流控制框图。

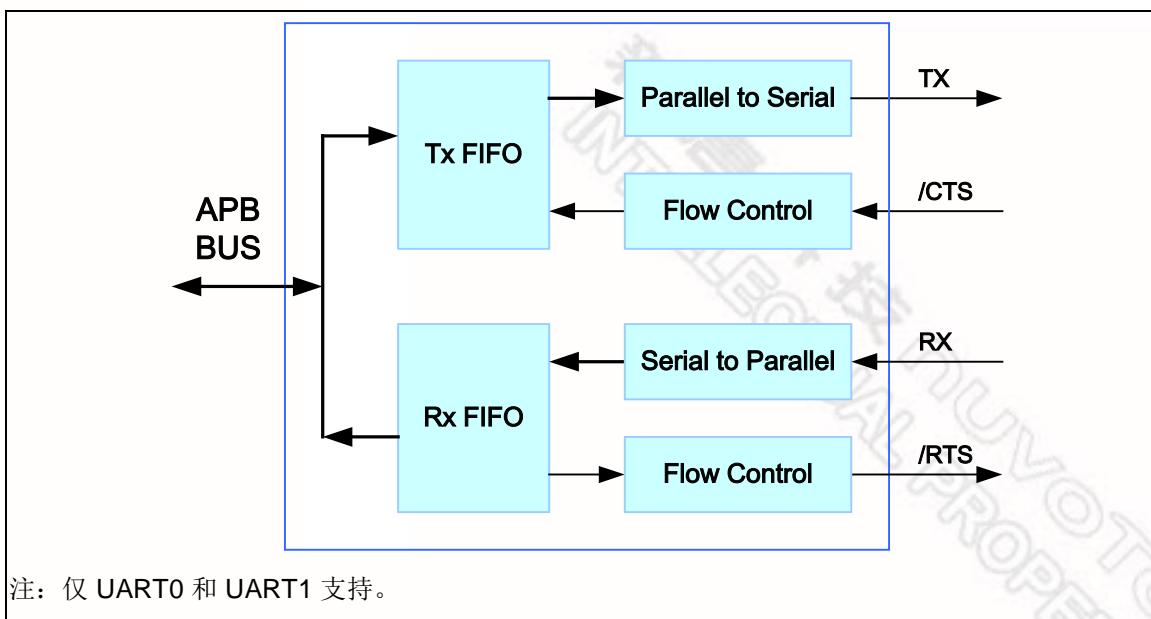


图 5-71 自动流控制框图

5.12.4 IrDA 模式

UART 支持 IrDA SIR (Serial Infrared) 传输编码和接收解码，通过设定 **UA_FUN_SEL** 寄存器的 **IrDA_EN** 位来选择 IrDA 模式。

IrDA 模式下，**UA_BAUD [DIV_X_EN]** 寄存器必须禁用。

Baud Rate = Clock / (16 * BRD)，其中 BRD 是波特率分频 **UA_BAUD** 寄存器。

下图为 IrDA 控制框图。

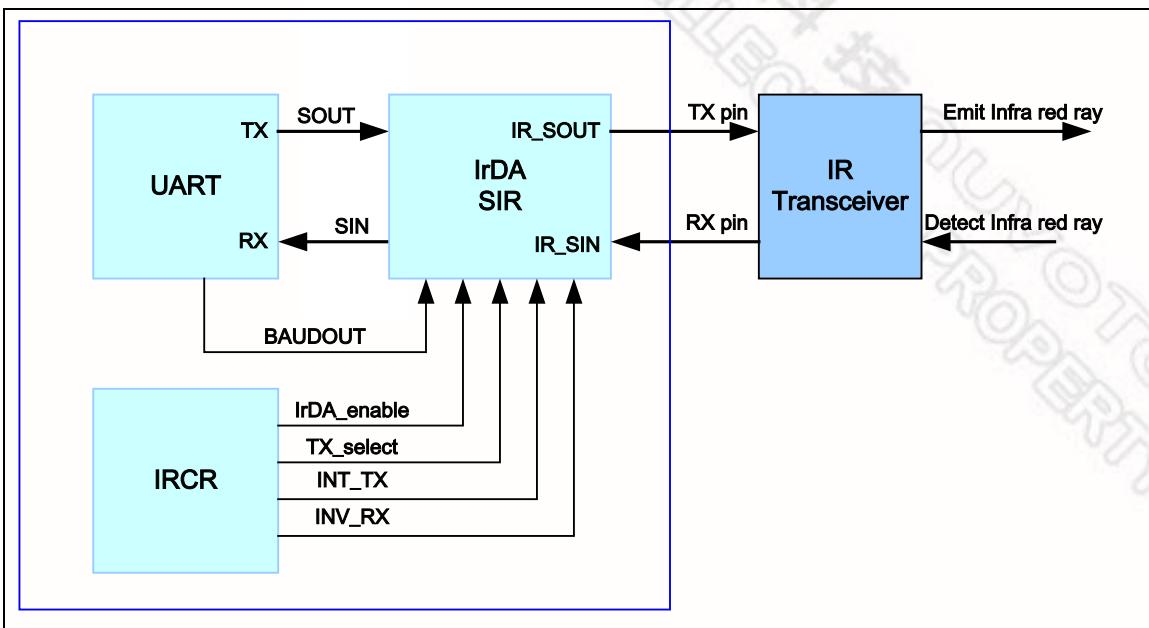


图 5-72 IrDA 框图

5.12.4.1 IrDA SIR 传送编码

IrDA SIR 传送编码调制 Non-Return-to Zero (NRZ) 传送位从 UART 输出。IrDA SIR 物理层指定使用归零反向调制编码 (Return-to-Zero, Inverted (RZI))，用一个红外光脉冲代表逻辑 0，被调整的脉冲输出到外部输出驱动器和红外线发光二极管。

在正常模式下，传输脉冲的宽度为 3/16 波特率周期。

5.12.4.2 IrDA SIR 接收解码

IrDA SIR 接收解码器对来自输入探测器的归零位比特流进行解调，并将接收到的 NRZ 串行比特流输出到 UART。在空闲状态里，解码器通常输入为高。（因此，IRCR 的 bit 6 默认设为1）。

当解码器输入为低时，表明接收到一个起始位。

5.12.4.3 IrDA SIR 操作

IrDA SIR 编码/解码提供 UART 数据流和半双工串行 SIR 之间的转换。IrDA 编码/解码波形图如下：

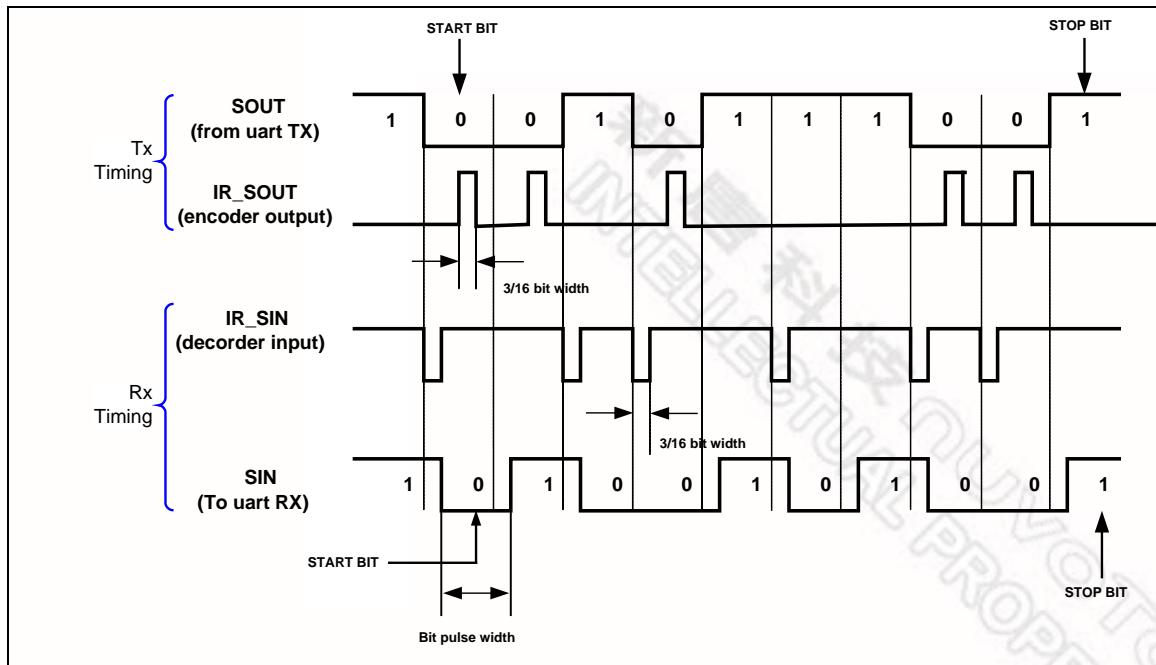


图 5-73 IrDA TX/RX 时序图

5.12.5 LIN (Local Interconnection Network) 模式

UART 支持 LIN 功能，通过设定 **UA_FUN_SEL** 寄存器的 **LIN_EN** 位选择 LIN 模式。在 LIN 模式下，每个字节域依照 LIN 的标准，初始由一个为 0 开始位（显性），后跟 8 个数据位（LSB 优先），最后是值为 1 的停止位（隐性）。下图为 LIN 功能模式的结构图：

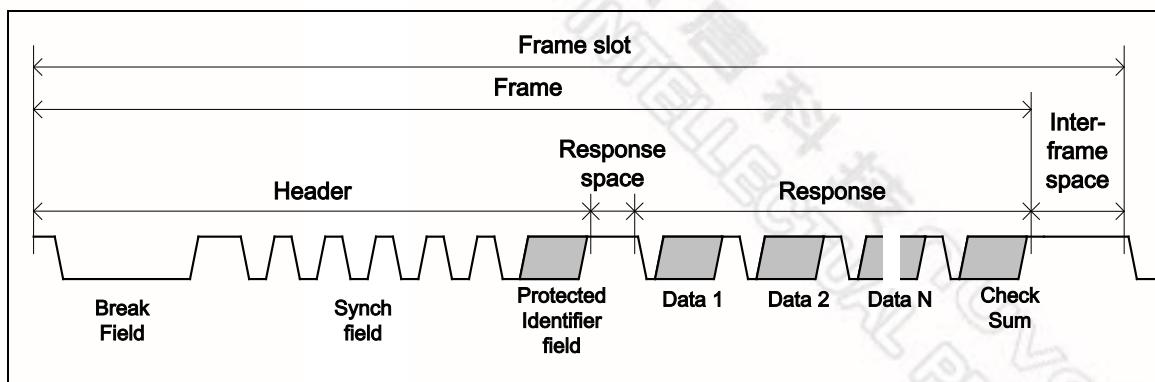


图 5-74 LIN 帧结构

LIN 总线发送传输 (TX) 编程流程:

1. 设置寄存器 **UA_FUN_SEL** 的 **LIN_EN** 位使能 LIN 总线模式。
2. 写 **UA_LIN_BKFL** 到 **UA_LIN_BCNT** 选择 break 域长度（break 域长度是 **UA_LIN_BKFL + 2**）。
3. 设置寄存器 **UA_LIN_BCNT** 的 **LIN_TX_EN** 位开始传输（当 break 域操作完成，**LIN_TX_EN** 将被自动清除）。
4. 写 0x55 到 **UA_THR** 请求同步域传输。
5. 写保护标识符值到 **UA_THR**，请求标识符域传输。
6. 当最后字节 **THR** 的停止位被发送到总线后，硬件将置 **UA_FSR** 寄存器中的 **TE_FLAG** 为 1。
7. 写 **N** 字节数据和检验和 (Checksum) 到 **UA_THR**，然后重复步骤 5 和步骤 6 传输数据。

LIN 总线接收传输 (RX) 编程流程:

1. 设置寄存器 **UA_FUN_SEL** 的 **LIN_EN** 位使能 LIN 总线模式。
2. 设置寄存器 **UA_LIN_BCNT** 的 **LIN_RX_EN** 位使能 LIN RX 模式。
3. 等待 **UA_ISR** 中的标志位 **LIN_RX_BREAK_IF** 用来检查 RX 有无 Break 域接收。
4. 等待 **UA_ISR** 中的标志位 **RDA_IF** 和读寄存器 **UR_RBR**。

5.12.6 RS-485 功能模式

UART 支持 **RS-485 9-位模式功能**。设置寄存器 UA_FUN_SEL 选择RS-485 模式。RS-485 通过 RTS 驱动控制异步串口的控制信号，使能 RS-485 驱动器。RS-485 模式，RX 与 TX 的许多特性与 UART 一样。

RS-485 发送模式，控制器可以配置成 RS-485 可寻址的从机模式，RS-485 主机发送可通过设置奇偶检验位 (9th bit) 为 1标识地址特性。对于数据特性，奇偶检验位设置为 0。设置寄存器 UA_LCR 控制第9位（PBE , EPE 和 SPE置位时，第9位发送 0; PBE 和 SPE 置位，EPE清零时，第 9 位发送1）。该控制器支持三种操作模式：RS-485 普通多点操作模式 (NMM)，RS-485 自动地址识别模式 (AAD) 和 RS-485 自动方向控制模式 (AUD)，可通过 UA_RS-485_CSR 寄存器的设置选择其中一种工作模式，通过设置 UA_TOR [DLY] 寄存器可以设置上一个停止位与下一个开始位之间的延迟时间。

RS-485 普通多点操作模式 (NMM)

RS-485 普通多点操作模式，首先，软件决定在检测到地址字节之前的数据是否存储到 RX-FIFO。如果软件想忽略在检测到地址之前的任何数据，流程是设置UART_FCR[RS485_RX_DIS]，然后使能 UA_RS-485[RS485_NMM]，接收器将会忽略数据，直到检测到地址字节 (bit9 =1) 并且地址字节数据存储到 RX-FIFO。如果软件想接收在检测到地址字节之前的任何数据，流程是禁用UART_FCR[RS485_RX_DIS]，然后使能 UA_RS-485[RS485_NMM]，接收器将接收任何数据。如果检测到地址字节 (bit9 =1)，会产生一个中断到 CPU，软件可以通过设置 UA_RS-485_CSR [RX_DIS]，决定是否使能或禁用接收器接收数据。如果使能接收器，就会接收所有字节数据并存储到 RX-FIFO。如果禁用接收器，会忽略所有接收到的字节数据，直到检测到下一个地址字节。如果设置寄存器 UA_RS-485_CSR [RX_DIS] 禁用接收器，当检测到下一个地址字节，控制器将清除 UA_RS-485_CSR [RX_DIS] 位且地址字节数据将存储到 RX-FIFO。

RS-485 自动地址识别模式 (AAD)

RS-485 自动地址识别模式，接收器在检测到地址字节 (bit9=1) 并且地址字节数据与 UA_RS-485[ADDR_MATCH] 的值相匹配之前将忽略所有数据。地址字节数据将存储在 RX-FIFO。所有接收字节数据将被接收并存储于 RX-FIFO 直到地址字节或数据字节与 UA_RS-485[ADDR_MATCH] 的值不匹配。

RS-485 自动方向模式 (AUD)

RS-485 控制器的另一个功能是 **RS-485 自动方向控制**。RS-485 通过 RTS 驱动控制异步串口的控制信号，使能RS-485 驱动器。RTS 连接到RS-485 驱动器，设置RTS线为高（逻辑1）使能RS-485 驱动器。设置 RTS 为低（逻辑0），使驱动器进入 tri-state 状态。用户通过设置寄存器 UA_MCR 中的 LEV_RTS 位改变 RTS 驱动电平。

编程流程示例：

1. 设置 UA_FUN_SEL 中的 FUN_SEL 选择 RS-485 功能。
2. 设置寄存器 UA_FCR 中的 RX_DIS 位使能或禁用 RS-485 接收器。
3. 设置 RS-485_NMM 或 RS-485_AAD 模式。
4. 如果选择 RS-485_AAD， ADDR_MATCH 设置成自动地址匹配值。
5. 设置 RS-485_AUD 来决定是否为自动方向控制。

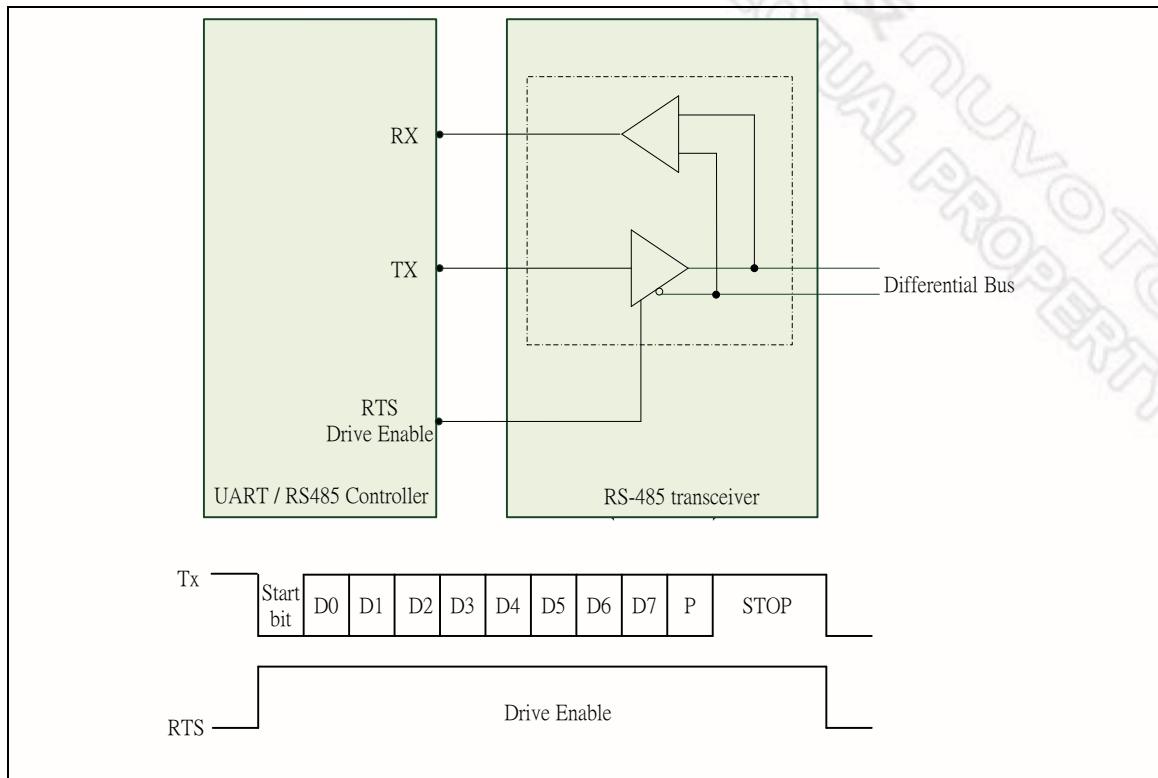


图 5-75 RS-485 帧结构



5.12.7 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
UART Base Address :				
Channel0 : UART0_BA (High Speed) = 0x4005_0000				
Channel1 : UART1_BA (Normal Speed)= 0x4015_0000				
Channel2 : UART2_BA (Normal Speed)= 0x4015_4000				
UA_RBR	UART0_BA+0x00	R	UART0 接收缓存寄存器	Undefined
	UART1_BA+0x00	R	UART1 接收缓存寄存器	Undefined
	UART2_BA+0x00	R	UART2 接收缓存寄存器	Undefined
UA_THR	UART0_BA+0x00	W	UART0 发送保持寄存器	Undefined
	UART1_BA+0x00	W	UART1 发送保持寄存器	Undefined
	UART2_BA+0x00	W	UART2 发送保持寄存器	Undefined
UA_IER	UART0_BA+0x04	R/W	UART0 中断使能寄存器	0x0000_0000
	UART1_BA+0x04	R/W	UART1 中断使能寄存器	0x0000_0000
	UART2_BA+0x04	R/W	UART2 中断使能寄存器	0x0000_0000
UA_FCR	UART0_BA+0x08	R/W	UART0 FIFO 控制寄存器	0x0000_0000
	UART1_BA+0x08	R/W	UART1 FIFO 控制寄存器	0x0000_0000
	UART2_BA+0x08	R/W	UART2 FIFO 控制寄存器	0x0000_0000
UA_LCR	UART0_BA+0x0C	R/W	UART0 线控制寄存器	0x0000_0000
	UART1_BA+0x0C	R/W	UART1 线控制寄存器	0x0000_0000
	UART2_BA+0x0C	R/W	UART2 线控制寄存器	0x0000_0000
UA_MCR	UART0_BA+0x10	R/W	UART0 Modem 控制寄存器	0x0000_0000
	UART1_BA+0x10	R/W	UART1 Modem 控制寄存器	0x0000_0000
	UART2_BA+0x10	R/W	保留	0x0000_0000
UA_MSR	UART0_BA+0x14	R/W	UART0 Modem 状态寄存器	0x0000_0000
	UART1_BA+0x14	R/W	UART1 Modem 状态寄存器	0x0000_0000
	UART2_BA+0x14	R/W	保留	0x0000_0000
UA_FSR	UART0_BA+0x18	R/W	UART0 FIFO 状态寄存器	0x1040_4000
	UART1_BA+0x18	R/W	UART1 FIFO 状态寄存器	0x1040_4000
	UART2_BA+0x18	R/W	UART2 FIFO 状态寄存器	0x1040_4000

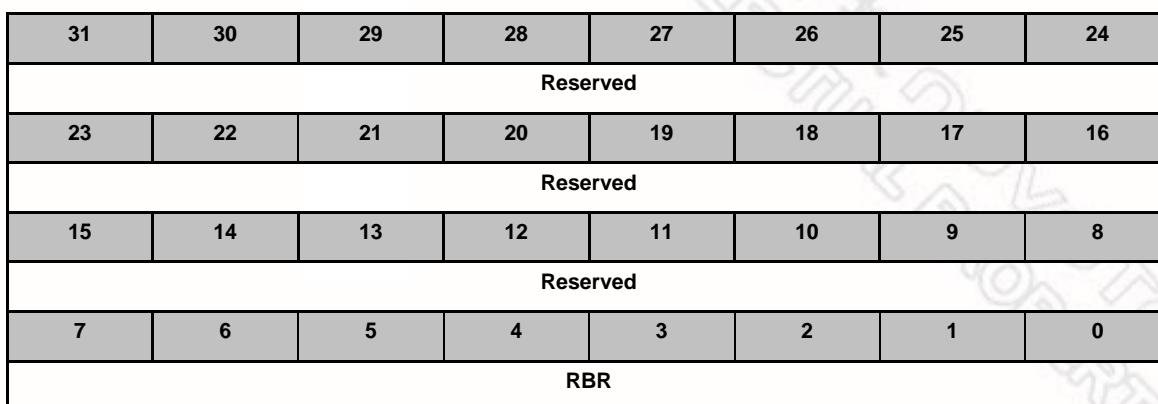
UA_ISR	UART0_BA+0x1C	R/W	UART0 中断状态寄存器	0x0000_0002
	UART1_BA+0x1C	R/W	UART1 中断状态寄存器	0x0000_0002
	UART2_BA+0x1C	R/W	UART2 中断状态寄存器	0x0000_0002
UA_TOR	UART0_BA+0x20	R/W	UART0 超时寄存器	0x0000_0000
	UART1_BA+0x20	R/W	UART1 超时寄存器	0x0000_0000
	UART2_BA+0x20	R/W	UART2 超时寄存器	0x0000_0000
UA_BAUD	UART0_BA+0x24	R/W	UART0 波特率分频寄存器	0x0F00_0000
	UART1_BA+0x24	R/W	UART1 波特率分频寄存器	0x0F00_0000
	UART2_BA+0x24	R/W	UART2 波特率分频寄存器	0x0F00_0000
UA_IRCR	UART0_BA+0x28	R/W	UART0 IrDA 控制寄存器	0x0000_0040
	UART1_BA+0x28	R/W	UART1 IrDA 控制寄存器	0x0000_0040
	UART2_BA+0x28	R/W	UART2 IrDA 控制寄存器	0x0000_0040
UA_ALT_CSR	UART0_BA+0x2C	R/W	UART0 复用控制/状态寄存器	0x0000_0000
	UART1_BA+0x2C	R/W	UART1 复用控制/状态寄存器	0x0000_0000
	UART2_BA+0x2C	R/W	UART2 复用控制/状态寄存器	0x0000_0000
UA_FUN_SEL	UART0_BA+0x30	R/W	UART0 功能选择寄存器	0x0000_0000
	UART1_BA+0x30	R/W	UART1 功能选择寄存器	0x0000_0000
	UART2_BA+0x30	R/W	UART2 功能选择寄存器	0x0000_0000



5.12.8 寄存器描述

接收缓存寄存器 (UA_RBR)

寄存器	偏移量	R/W	描述	复位后的值
UA_RBR	UART0_BA+0x00	R	UART0 接收缓存寄存器	Undefined
	UART1_BA+0x00	R	UART1 接收缓存寄存器	Undefined
	UART2_BA+0x00	R	UART2 接收缓存寄存器	Undefined



Bits	描述	
[31:8]	Reserved	保留
[7:0]	RBR	接收缓存寄存器（只读） 读此寄存器，UART 将返回一组从 RX 管脚接收到的 8 位数据(LSB first)

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	UART0_BA+0x00	W	UART0 发送保持寄存器	Undefined
	UART1_BA+0x00	W	UART1 发送保持寄存器	Undefined
	UART2_BA+0x00	W	UART2 发送保持寄存器	Undefined

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

Bits	描述	
[31:8]	Reserved	保留
[7:0]	THR	发送保持寄存器 通过写该寄存器, UART 将通过 TX 管脚 (LSB first) 发送 8-位数据。

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	UART0_BA+0x04	R/W	UART0 中断使能寄存器	0x0000_0000
	UART1_BA+0x04	R/W	UART1 中断使能寄存器	0x0000_0000
	UART2_BA+0x04	R/W	UART2 中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
DMA_RX_EN	DMA_TX_EN	AUTO_CTS_EN	AUTO_RTS_EN	TIME_OUT_EN	Reserved		LIN_RX_BRK_IEN
7	6	5	4	3	2	1	0
Reserved	WAKE_EN	BUF_ERR_IE_N	RTO_IEN	MODEM_IEN	RLS_IEN	THRE_IEN	RDA_IEN

Bits	描述	
[31:16]	Reserved	保留
[15]	DMA_RX_EN	<p>RX DMA 使能 (UART2 通道不可用) 该位使能或禁用 RX DMA 服务。 1 = 使能 RX DMA 0 = 禁用 RX DMA</p>
[14]	DMA_TX_EN	<p>TX DMA 使能 (UART2 通道不可用) 该位使能或禁用 TX DMA 服务。 1 = 使能 TX DMA 0 = 禁用 TX DMA</p>
[13]	AUTO_CTS_EN	<p>CTS 自动流控制使能 (UART2 通道不可用) 1 = 使能 CTS 自动流控制 0 = 禁用 CTS 自动流控制 当 CTS 自动流控制使能时, 当侦测到 CTS 输入信号, UART 将向外部驱动器发送数据 (UART 直到 CTS 被侦测才发送数据)</p>
[12]	AUTO_RTS_EN	<p>RTS 自动流控制使能 (UART2 通道不可用) 1 = 使能 RTS 自动流控制 0 = 禁用 RTS 自动流控制 当 RTS 自动流使能, 当RX FIFO 的数值和 UA_FCR[RTS_TRILEV]相等时, UART 将</p>

		禁止 RTS 信号
[11]	TIME_OUT_EN	Time Out 计数器使能 1 = 使能 Time-out 计数器 0 = 禁用 Time-out 计数器
[10:9]	Reserved	保留
[8]	LIN_RX_BRK_IEN	LIN RX Break 域检测中断使能 1 = 使能 Lin 总线 RX break 域中断 0 = 屏蔽 Lin 总线 RX break 域中断 注：该位用于 LIN 功能模式。
[7]	Reserved	保留
[6]	WAKE_EN	UART 唤醒功能使能 (UART2 通道不可用) 0 = 禁用 UART 唤醒功能 1 = 使能 UART 唤醒功能，当系统在掉电模式下，外部 CTS 的改变将芯片从掉电模式下唤醒
[5]	BUF_ERR_IEN	Buffer Error 中断使能 1 = 使能 INT_BUF_ERR 0 = 屏蔽 INT_BUF_ERR
[4]	RTO_IEN	RX Time Out 中断使能 1 = 使能 INT_TOUT 0 = 屏蔽 INT_TOUT
[3]	MODEM_IEN	Modem 状态中断使能 (UART2 通道不可用) 1 = 使能 INT_MODEM 0 = 屏蔽 INT_MODEM
[2]	RLS_IEN	线上接收中断状态使能 1 = 使能 INT_RLS 0 = 屏蔽 INT_RLS
[1]	THRE_IEN	发送保持寄存器空中断使能 1 = 使能 INT_THRE 0 = 禁用 INT_THRE
[0]	RDA_IEN	可接收数据中断使能 1 = 使能 INT_RDA 0 = 禁用 INT_RDA

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	UART0_BA+0x08	R/W	UART0 FIFO 控制寄存器	0x0000_0000
	UART1_BA+0x08	R/W	UART1 FIFO 控制寄存器	0x0000_0000
	UART2_BA+0x08	R/W	UART2 FIFO 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				RTS_TRI_LEV			
15	14	13	12	11	10	9	8
Reserved							RX_DIS
7	6	5	4	3	2	1	0
RFITL				Reserved	TFR	RFR	Reserved

Bits	描述																			
[31:20]	Reserved	保留																		
[19:16]	RTS_TRILEV	<p>RTS 触发自动流控制使用 (UART2 通道不可用)</p> <table border="1" style="margin-left: 20px;"> <tr> <th>RTS_TRILEV</th> <th>Trigger Level (Bytes)</th> </tr> <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>0100</td> <td>30/14 (高速/常速)</td> </tr> <tr> <td>0101</td> <td>46/14 (高速/常速)</td> </tr> <tr> <td>0110</td> <td>62/14 (高速/常速)</td> </tr> <tr> <td>others</td> <td>62/14 (高速/常速)</td> </tr> </table> <p>注：该寄存器用于自动 RTS 流控制。</p>	RTS_TRILEV	Trigger Level (Bytes)	0000	01	0001	04	0010	08	0011	14	0100	30/14 (高速/常速)	0101	46/14 (高速/常速)	0110	62/14 (高速/常速)	others	62/14 (高速/常速)
RTS_TRILEV	Trigger Level (Bytes)																			
0000	01																			
0001	04																			
0010	08																			
0011	14																			
0100	30/14 (高速/常速)																			
0101	46/14 (高速/常速)																			
0110	62/14 (高速/常速)																			
others	62/14 (高速/常速)																			
[15:9]	Reserved	保留																		
[8]	RX_DIS	<p>接收器禁用寄存器</p> <p>是否禁用接收器 (置 1 禁用接收器)</p> <p>1 = 禁用接收器 0 = 使能接收器</p>																		

		注：该位用于 RS-485 普通多点模式。需要在 UA_ALT_CSR [RS-485_NMM] 之前设置。																		
[7:4]	RFITL	<p>RX FIFO 中断 (INT_RDA) 触发级别</p> <p>FIFO 接收字节数等于 RFITL 后，RDA_IF 将被置位（如果UA_IER [RDA_IEN] 使能，将产生中断）。</p> <table border="1"> <thead> <tr> <th>RFITL</th><th>INTR_RDA Trigger Level (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> <tr><td>0100</td><td>30/14 (高速/常速)</td></tr> <tr><td>0101</td><td>46/14 (高速/常速)</td></tr> <tr><td>0110</td><td>62/14 (高速/常速)</td></tr> <tr><td>others</td><td>62/14 (高速/常速)</td></tr> </tbody> </table>	RFITL	INTR_RDA Trigger Level (Bytes)	0000	01	0001	04	0010	08	0011	14	0100	30/14 (高速/常速)	0101	46/14 (高速/常速)	0110	62/14 (高速/常速)	others	62/14 (高速/常速)
RFITL	INTR_RDA Trigger Level (Bytes)																			
0000	01																			
0001	04																			
0010	08																			
0011	14																			
0100	30/14 (高速/常速)																			
0101	46/14 (高速/常速)																			
0110	62/14 (高速/常速)																			
others	62/14 (高速/常速)																			
[3]																				
Reserved																				
保留																				
<p>TX 软件复位</p> <p>当 TX_RST 置位，发送 FIFO 内的所有数据和 TX 内部状态机将被清除。</p> <p>1 = 该位写 1 将复位 TX 内部状态机和指针。 0 = 该位写 0 无效。</p> <p>注：该位至少 3 个 UART 时钟周期后会自动清零。</p>																				
<p>RX 软件复位</p> <p>当 RX_RST 置位，接收 FIFO 内的所有数据和 RX 内部状态机将被清除。</p> <p>1 = 该位写 1 将复位 RX 内部状态机和指针。 0 = 该位写 0 无效。</p> <p>注：该位至少 3 个 UART 时钟周期后会自动清零。</p>																				
[0]																				
Reserved																				
保留																				

Line 控制寄存器 (UA_LCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_LCR	UART0_BA+0x0C	R/W	UART0 Line 控制寄存器	0x0000_0000
	UART1_BA+0x0C	R/W	UART1 Line 控制寄存器	0x0000_0000
	UART2_BA+0x0C	R/W	UART2 Line 控制寄存器	0x0000_0000

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

Bits	描述	
[31:7]	Reserved	保留
[6]	BCB	Break 控制位 当该位被置逻辑 1，串行数据输出 (TX) 将强制到 Spacing 状态 (logic 0)。该位仅作用于 TX，对传输逻辑不起作用。
[5]	SPE	Stick 奇偶校验使能 1 = 如果 bit 3 和 4 为逻辑 1，奇偶校验位发送和检验值为逻辑 0。如果 bit 3 是 1，bit 4 是 0，则奇偶校验位发送和检验值为 1。 0 = 禁用 Stick 奇偶校验
[4]	EPE	Even 奇偶校验使能 1 = 逻辑 1 的偶数数目在每个字中被发送和检验 0 = 逻辑 1 的奇数数目在每个字节中被发送和检验 该位只在 bit 3 (parity bit enable) 置位时有效。
[3]	PBE	奇偶校验位使能 1 = 每一个发送字符中都产生奇偶校验位，对每一个传进来的数据进行奇偶校验位检测 0 = 无奇偶校验位
[2]	NSB	“停止位” 数目 1= 当发送数据，选择 5-位 字长度时，产生 1.5 “STOP bit” 0= 当发生数据时，产生 1 “STOP bit” 当选择 6-, 7- 和 8-位字长度时，产生 2 “STOP bit”

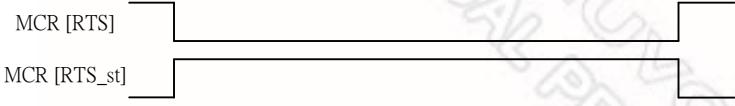
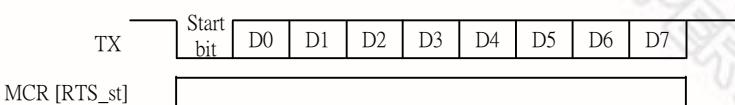
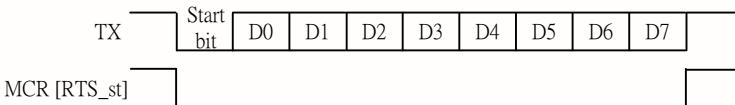
字长度选择			
[1:0]	WLS	WLS[1:0]	Character length
		00	5-bit
		01	6-bit
		10	7-bit
		11	8-bit

**MODEM 控制寄存器 (UA_MCR) (UART2 通道不可用)**

寄存器	偏移量	R/W	描述	复位后的值
UA_MCR	UART0_BA+0x10	R/W	UART0 Modem 控制寄存器	0x0000_0000
	UART1_BA+0x10	R/W	UART1 Modem 控制寄存器	0x0000_0000
	UART2_BA+0x10	R/W	保留	0x0000_0000

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

Bits	描述	
[31:14]	Reserved	保留
[13]	RTS_ST	RTS 管脚状态 (只读) (UART2 通道不可用) 该位表示 RTS 的输出管脚状态。
[12:10]	Reserved	保留

		RTS 触发电平 (UART2 通道不可用) 该位可改变 RTS 触发电平。 1= 高电平触发 0= 低电平触发 <u>UART Mode : MCR[Lev_RTS] = 1</u>  <u>UART Mode : MCR[Lev_RTS] = 0</u>  <u>RS-485 Mode : MCR[Lev_RTS] = 1</u>  <u>RS-485 Mode : MCR[Lev_RTS] = 0</u> 
[9]	LEV_RTS	保留
[8:2]	Reserved	保留
[1]	RTS	RTS (Request-To-Send) 信号 (UART2 通道不可用) 0 = 驱动 RTS 管脚为逻辑 1 (如果 LEV_RTS 设为低电平触发)。 1 = 驱动 RTS 管脚为逻辑 0 (如果 LEV_RTS 设为低电平触发)。 0 = 驱动 RTS 管脚为逻辑 0 (如果 LEV_RTS 设为高电平触发)。 1 = 驱动 RTS 管脚为逻辑 1 (如果 LEV_RTS 设为高电平触发)。
[0]	Reserved	保留

Modem 状态寄存器 (UA_MSR) (UART2 通道不可用)

寄存器	偏移量	R/W	描述	复位后的值
UA_MSR	UART0_BA+0x14	R/W	UART0 Modem 状态寄存器	0x0000_0000
	UART1_BA+0x14	R/W	UART1 Modem 状态寄存器	0x0000_0000
	UART2_BA+0x14	R/W	保留	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	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			CTS_ST	Reserved			DCTSF

Bits	描述	
[31:9]	Reserved	保留
[8]	LEV_CTS	<p>CTS 触发电平 (UART2 通道不可用) 该位可改变 CTS 触发电平。 1= 高电平触发 0= 低电平触发</p>
[7:5]	Reserved	保留
[4]	CTS_ST	<p>CTS Pin Status (只读) (UART2 通道不可用) 该位表示 CTS 的管脚状态。</p>
[3:1]	Reserved	保留
[0]	DCTSF	<p>侦测 CTS 状态改变标志位 (只读) (UART2 通道不可用) This bit is set 每当 CTS 的输入状态改变时，该位置位，如果此时 UA_IER [MODEM_IEN] 为1，将向 CPU 产生重大请求。 该位软件写 1 清零。</p>

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	UART0_BA+0x18	R/W	UART0 FIFO 状态寄存器	0x1040_4000
	UART1_BA+0x18	R/W	UART1 FIFO 状态寄存器	0x1040_4000
	UART2_BA+0x18	R/W	UART2 FIFO 状态寄存器	0x1040_4000

31	30	29	28	27	26	25	24
Reserved			TE_FLAG	Reserved			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
Reserved	BIF	FEF	PEF	RS485_ADD_DETF	Reserved		RX_OVER_IF

Bits	描述
[31:29]	Reserved 保留
[28]	TE_FLAG 发送空标志位 (Read Only) 当 TX FIFO (UA_THR) 为空, 而且最后一个字节的 STOP 位已发送, 该位由硬件置位。 当 TX FIFO 不为空或最后一个字节传输未完成, 该位自动清除。
[27:25]	Reserved 保留
[24]	TX_OVER_IF TX 溢出错误中断标志位 (Read Only) 如果 TX FIFO (UA_THR) 已满, 再向 UA_THR 写入, 将会导致该位置位为逻辑 1。 注: 该位只读, 但可以写 1 清除。
[23]	TX_FULL 发送 FIFO 满 (Read Only) 该位表示 TX FIFO 是否已满 当 TX_POINTER 等于 64/16/16(UART0/UART1/UART0), 该位置位, 否则该位由硬件清除。
[22]	TX_EMPTY 发送 FIFO 空 (Read Only) 该位表示 TX FIFO 是否为空。 当 TX FIFO 的最后一个字节已经被传输到发送移位寄存器中, 硬件将置该位为高。当写数据到 THR (TX FIFO 非空), 该位将被清除。
[21:16]	TX_POINTER TX FIFO 指针 (Read Only) 该域表示 TX FIFO 缓存指针。当 CPU 写 1 字节到 UA_THR, TX_POINTER 增加1。

		当 TX FIFO 的 1 个字节传输到发送移位寄存器, TX_POINTER 减 1。
[15]	RX_FULL	<p>接收 FIFO 满 (Read Only)</p> <p>该位表示 RX FIFO 是否已满。</p> <p>当 RX_POINTER 等于 64/16/16(UART0/UART1/UART2), 该位置位, 否则该位由硬件清除。</p>
[14]	RX_EMPTY	<p>接收 FIFO 空 (Read Only)</p> <p>该位表示 RX FIFO 是否为空。</p> <p>当CPU 读完 RX FIFO 的最后一个字节, 硬件置该位为高。当 UART 收到任何新的数据, 该位被清除。</p>
[13:8]	RX_POINTER	<p>RX FIFO 指针 (Read Only)</p> <p>该域表示 RX FIFO 缓存指针。当 UART 从外部设备接收到 1 个字节, RX_POINTER 增加 1。如果 CPU 读取 RX FIFO 的 1 个字节, RX_POINTER 减 1。</p>
[7]	Reserved	保留
[6]	BIF	<p>Break 中断标志位 (Read Only)</p> <p>每当接收到数据输入 (RX) 维持在 “spacing state” (logic 0) 的时间长于一个全字的传输时间 (即“start bit” + data bits + parity + stop bits 的总时间), 该位置 1。当 CPU 向该位写 1, 该位重置。</p> <p>注: 该位只读, 但可以写 1 清除。</p>
[5]	FEF	<p>Framing Error 标志位 (Read Only)</p> <p>每当接收到的字符没有合法的 “stop bit” (即stop bit 跟着最后的数据位后或奇偶校验位检测为0) 该位置 1。当 CPU 向该位写 1, 该位重置。</p> <p>注: 该位只读, 但可以写 1 清除。</p>
[4]	PEF	<p>Parity Error 标志位 (Read Only)</p> <p>每当接收到的字符没有合法的奇偶校验位, 该位置 1。当 CPU 向该位写 1, 该位重置。</p> <p>注: 该位只读, 但可以写 1 清除。</p>
[3]	RS485_ADD_DETF	<p>RS-485 地址字节检测标志位 (Read Only)</p> <p>当在 RS-485 模式, 接收器检测任一接收地址字节字符 (bit9 = '1'), 该位置 1 且置 UA_ALT_CSR [RS-485_ADD_EN]。当 CPU 向该位写 1, 该位重置。</p> <p>注: 该位用于 RS-485 功能模式。</p> <p>注: 该位只读, 但可以写 1 清除。</p>
[2:1]	Reserved	保留
[0]	RX_OVER_IF	<p>RX Overflow Error IF (Read Only)</p> <p>当 RX FIFO 溢出, 该位置位。</p> <p>如果接收到的数据数量大于 RX_FIFO (UA_RBR) 的范围 (64/16 字节 UART0/UART1), 该位置位。</p> <p>注: 该位只读, 但可以写 1 清除。</p>

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	UART0_BA+0x1C	R/W	UART0 中断状态寄存器				0x0000_0002
	UART1_BA+0x1C	R/W	UART1 中断状态寄存器				0x0000_0002
	UART2_BA+0x1C	R/W	UART2 中断状态寄存器				0x0000_0002

31	30	29	28	27	26	25	24
HW_LIN_RX_BREAK_INT	Reserved	HW_BUF_ER_R_INT	HW_TOUT_INT	HW_MODEM_INT	HW_RLS_INT	Reserved	
23	22	21	20	19	18	17	16
HW_LIN_RX_BREAK_IF	Reserved	HW_BUF_ER_R_IF	HW_TOUT_IF	HW_MODEM_IF	HW_RLS_IF	Reserved	
15	14	13	12	11	10	9	8
LIN_RX_BREAK_INT	Reserved	BUF_ERR_IN_T	TOUT_INT	MODEM_INT	RLS_INT	THRE_INT	RDA_INT
7	6	5	4	3	2	1	0
LIN_RX_BREAK_IF	Reserved	BUF_ERR_IF	TOUT_IF	MODEM_IF	RLS_IF	THRE_IF	RDA_IF

Bits	描述	
[31]	HW_LIN_RX_BREAK_INT	DMA 模式下, LIN 总线 RX Break 域检测中断标志 (Read Only) 如果 LIN_RX_BRK_IEN 和 HW_LIN_RX_BREAK_IF 都被置为1, 该位置 1。 1 = DMA 模式下, LIN RX Break 中断产生。 0 = DMA 模式下, 无 LIN RX Break 中断产生。
[30]	Reserved	保留
[29]	HW_BUF_ERR_INT	DMA 模式下, Buffer Error 中断标志 (Read Only) 如果 BUF_ERR_IEN 和 HW_BUF_ERR_IF 都被置为1, 该位置 1。 1 = DMA 模式下, buffer error 中断产生。 0 = DMA 模式下, 无 buffer error 中断产生。
[28]	HW_TOUT_INT	DMA 模式, Time Out 中断标志 (Read Only) 如果 TOUT_IEN 和 HW_TOUT_IF 都被置为1, 该位置 1。 1 = DMA 模式下, Tout 中断产生。 0 = DMA 模式下, 无 Tout 中断产生。
[27]	HW_MODEM_INT	DMA 模式, MODEM 状态中断标志 (Read Only) (UART2 通道不可用) 如果 MODEM_IEN 和 HW_MODEM_IF 都被置为1, 该位置 1。 1 = DMA 模式下, Modem 中断产生。

		0 = DMA 模式下, 无 Modem 中断产生。
[26]	HW_RLS_INT	DMA 模式, 接收 Line 状态中断标志 (Read Only) 如果 RLS_IEN 和 HW_RLS_IF 都被置为1, 该位置 1。 1 = DMA 模式下, RLS 中断产生。 0 = DMA 模式下, 无 RLS 中断产生。
[25:24]	Reserved	保留
[23]	HW_LIN_RX_BREAK_IF	DMA 模式, LIN 总线 RX Break 域检测中断标志 (Read Only) 当 RX 接收到 LIN break 域该位置1。如果 UA_IER [LIN_RX_BRK_IEN] 使能, LIN RX break 中断将产生。 注: 该位只读, 但可写 1 到 UA_ISR [7] 清除该位。
[22]	Reserved	保留
[21]	HW_BUF_ERR_IF	DMA 模式, Buffer Error 中断标志 (Read Only) 当 TX 或 RX FIFO 溢出 (TX_OVER_IF 或 RX_OVER_IF 被置位), 该位置位。当 BUF_ERR_IF 被置位, 传输也许不正确。如果 UA_IER [BUF_ERR_IEN] 使能, buffer error 中断将产生。 注: 当 TX_OVER_IF 和 RX_OVER_IF 都被清除时, 该位被清除。
[20]	HW_TOUT_IF	DMA 模式, Time Out 中断标志 (Read Only) 当 RX FIFO 非空而且 RX FIFO 无活动发生, 超时计数器等于 TOIC 时, 该位置位。如果 UA_IER [TOUT_IEN] 使能, Tout 中断将产生。 注: 该位只读, 用户可以读 UA_RBR (RX 在活动时) 清除该位。
[19]	HW_MODEM_IF	DMA 模式, MODEM 中断标志 (Read Only) (UART2 通道不可用) 当 CTS 管脚出现状态改变 (DCTS=1), 该位置位。如果 UA_IER [MODEM_IEN] 使能, Modem 中断将产生 注: 该位只读, 当 DCTS 位被写 1 清除时, 该位重置为 0。
[18]	HW_RLS_IF	DMA 模式, 接收 Line 状态标志 (Read Only) 当 RX 接收数据有 parity error, framing error 或 break error (至少 BIF, FEF 和 PEF 3 位中有 1 位被置) 该位置位。如果 UA_IER [RLS_IEN] 使能, RLS 中断产生。 注: 当在 RS-485 功能模式, 该域包括“接收检测任一接收到的地址字节字符 (bit9 = '1') 位”。 注: 该位只读, 当 BIF, FEF 和 PEF 三位都被清除后, 该位重置为 0。
[17:16]	Reserved	保留
[15]	LIN_RX_BREAK_INT	LIN 总线 RX Break 域检测中断标志 (Read Only) 如果 LIN_RX_BRK_IEN 和 LIN_RX_BREAK_IF 都被置1, 该位置 1。 1 = LIN RX Break 中断产生 0 = 无 LIN RX Break 中断产生
[14]	Reserved	保留
[13]	BUF_ERR_INT	Buffer Error 中断标志 (Read Only) 如果 BUF_ERR_IEN 和 BUF_ERR_IF 都被置1, 该位置 1。

		1 = buffer error 中断产生 0 = 无 buffer error 中断产生
[12]	TOUT_INT	Time Out 中断标志 (Read Only) 如果 TOUT_IEN 和 TOUT_IF 都被置1, 该位置 1。 1 = The Tout 中断产生 0 = 无 Tout 中断产生
[11]	MODEM_INT	MODEM 状态中断标志 (Read Only) (UART2 通道不可用) 如果 MODEM_IEN 和 MODEM_IF 都被置1, 该位置 1。 1 = Modem 中断产生 0 = No Modem 中断产生
[10]	RLS_INT	接收 Line 状态中断标志 (Read Only). 如果 RLS_IEN 和 RLS_IF 都被置1, 该位置 1。 1 = The RLS 中断产生 0 = 无 RLS 中断产生
[9]	THRE_INT	发送保持寄存器空中断标志 (Read Only). 如果 THRE_IEN 和 THRE_IF 都被置1, 该位置 1。 1 = THRE中断产生 0 = 无 THRE中断产生
[8]	RDA_INT	接收可用数据中断标志 (Read Only). 如果 RDA_IEN 和 RDA_IF 都被置1, 该位置 1。 1 = RDA中断产生 0 = 无 RDA中断产生
[7]	LIN_RX_BREAK_IF	LIN 总线 RX Break 域检测标志 (Read Only) 当 RX 接收到 LIN Break 域, 该位置 1。如果 UA_IER [LIN_RX_BRK_IEN] 使能, LIN RX Break 中断将发生。 注: 该位只读, 但可写 1 清除。
[6]	Reserved	保留
[5]	BUF_ERR_IF	Buffer Error 中断标志 (Read Only) 当 TX 或 RX FIFO 溢出 (TX_OVER_IF 或 RX_OVER_IF 置位) 该位置位。当 BUF_ERR_IF 置位, 传输可能不正确。如果 UA_IER [BUF_ERR_IEN] 使能, buffer error 中断产生。 注: 当 TX_OVER_IF 和 RX_OVER_IF 被清除时, 该位清除。
[4]	TOUT_IF	Time Out 中断标志 (Read Only) 当 RX FIFO 非空而且 RX FIFO 无活动发生, 超时计数器等于 TOIC 时, 该位置位。如果 UA_IER [TOUT_IEN] 使能, Tout 中断将产生。 注: 该位只读, 用户可以读 UA_RBR (RX 处于活动状态) 清除该位。
[3]	MODEM_IF	MODEM 中断标志 (Read Only) (UART2 通道不可用) 当 CTS 管脚有状态改变 (DCTS=1), 该位置位。如果 UA_IER [MODEM_IEN] 使能,

		Modem 中断将产生。 注：该位只读，当向 DCTSF 写 1 清除 DCTSF 后，该位清除。
[2]	RLS_IF	<p>接收 Line 中断标志 (Read Only).</p> <p>当 RX 接收数据有 parity error, framing error 或 break error (至少 BIF, FEF 和 PEF 3 位中有 1 位被置) 该位置位。如果 UA_IER [RLS_IEN] 使能，RLS 中断将产生。</p> <p>注：当在 RS-485 功能模式，该域包括“接收检测任一接收到的地址字节字符 (bit9 = '1') 位”。</p> <p>注：该位只读，当 BIF, FEF 和 PEF 三位都被清除后，该位重置为 0。</p>
[1]	THRE_IF	<p>发送保持寄存器空中断标志 (Read Only).</p> <p>当 TX FIFO 的最后数据传输到发送移位寄存器时，该位置位。如果 UA_IER [THRE_IEN] 使能，THRE 中断将产生。</p> <p>注：该位只读，当写数据到 THR (TX FIFO 非空) 时，该位将被清除。</p>
[0]	RDA_IF	<p>接收可用数据中断标志 (Read Only).</p> <p>当 RX FIFO 的数据数量等于 RFITL 时，RDA_IF 将被置位。如果 UA_IER [RDA_IEN] 使能，RDA 中断将产生。</p> <p>注：该位只读，当 RX FIFO 的未读数据低于阈值水平时，该位将被清除。</p>

UART 中断源	中断使能位	中断指示中断控制	中断标志位	标志位清除
LIN RX Break Field Detected interrupt	LIN_RX_BRK_IEN	HW_LIN_RX_BREAK_INT	HW_LIN_RX_BREAK_IF	Write '1' to LIN_RX_BREAK_IF
Buffer Error Interrupt INT_BUF_ERR	BUF_ERR_IEN	HW_BUF_ERR_INT	HW_BUF_ERR_IF = (TX_OVER_IF or RX_OVER_IF)	Write '1' to TX_OVER_IF/RX_OVER_IF
RX Timeout Interrupt INT_TOUT	RTO_IEN	HW_TOUT_INT	HW_TOUT_IF	Read UA_RBR
Modem Status Interrupt INT_MODEM	MODEM_IEN	HW_MODEM_INT	HW_MODEM_IF = (DCTS)	Write '1' to DCTS
Receive Line Status Interrupt INT_RLS	RLS_IEN	HW_RLS_INT	HW_RLS_IF = (BIF or FEF or PEF or RS-485_ADD_DET)	Write '1' to BIF/FEF/PEF/RS-485_ADD_DET
Transmit Holding Register Empty Interrupt INT_THRE	THRE_IEN	HW_THRE_INT	HW_THRE_IF	Write UA_THR
Receive Data Available Interrupt INT_RDA	RDA_IEN	HW_RDA_INT	HW_RDA_IF	Read UA_RBR

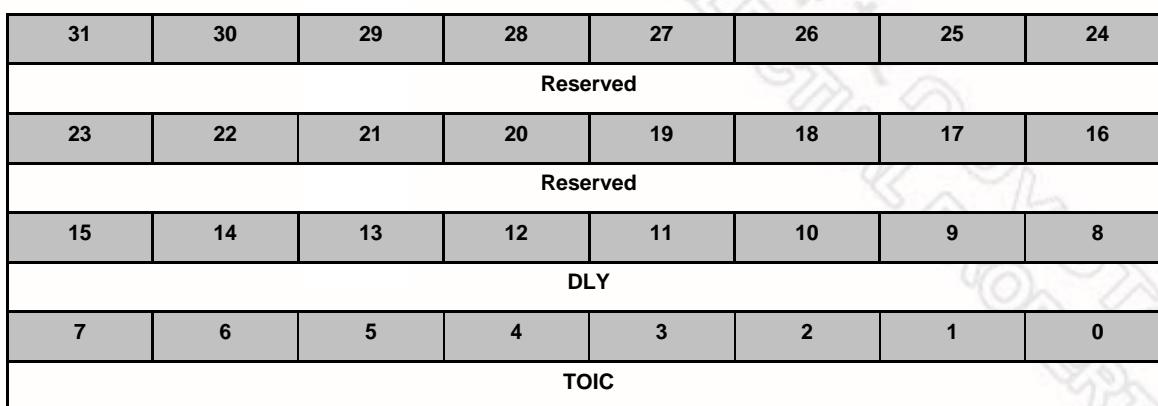
表 5-10 DMA 模式下 UART 的中断源和标志位表

UART 中断源	中断使能位	中断指示中断控制	中断标志位	标志位清除
LIN RX Break Field Detected interrupt	LIN_RX_BRK_IEN	LIN_RX_BREAK_INT	LIN_RX_BREAK_IF	Write '1' to LIN_RX_BREAK_IF
Buffer Error Interrupt INT_BUF_ERR	BUF_ERR_IEN	BUF_ERR_INT	BUF_ERR_IF = (TX_OVER_IF or RX_OVER_IF)	Write '1' to TX_OVER_IF/RX_OVER_IF
RX Timeout Interrupt INT_TOUT	RTO_IEN	TOUT_INT	TOUT_IF	Read UA_RBR
Modem Status Interrupt INT_MODEM	MODEM_IEN	MODEM_INT	MODEM_IF = (DCTS)	Write '1' to DCTS
Receive Line Status Interrupt INT_RLS	RLS_IEN	RLS_INT	RLS_IF = (BIF or FEF or PEF)	Write '1' to BIF/FEF/PEF
Transmit Holding Register Empty Interrupt INT_THRE	THRE_IEN	THRE_INT	THRE_IF	Write UA_THR
Receive Data Available Interrupt INT_RDA	RDA_IEN	RDA_INT	RDA_IF	Read UA_RBR

表 5-11 软件模式下 UART 的中断源和标志位表

超时寄存器 (UA_TOR)

寄存器	偏移量	R/W	描述	复位后的值
UA_TOR	UART0_BA+0x20	R/W	UART0 超时寄存器	0x0000_0000
	UART1_BA+0x20	R/W	UART1 超时寄存器	0x0000_0000
	UART2_BA+0x20	R/W	UART2 超时寄存器	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留
[15:8]	DLY	<p>TX 延迟时间值 该域用于编程上一次停止位和下一次开始位之间的传输延迟时间。</p>
[7:0]	TOIC	<p>Time Out 中断比较器 当 RX FIFO 接收到一个新的数据字，超时计数器重置并开始计数 (the counting clock = baud rate)。一旦超时计数器的内容 (TOUT_CNT) 等于超时中断比较器 (TOIC)，如果此时 UA_IER [RTO_IEN] 为使能，则接收超时中断 (INT_TOUT) 产生。新来的数据字或 RX FIFO 为空清除 INT_TOUT。为了避免接收超时中断一接收一个字符就立即产生，TOIC 的值必须设置在 40 到 255 之间。例如，如果 TOIC 为 40，当 UART 传输设置为 1 stop 位 和 无奇偶校验位时，超时中断将在4个字符没有收到之后产生。</p>

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	UART0_BA+0x24	R/W	UART0 波特率分频寄存器	0x0F00_0000
	UART1_BA+0x24	R/W	UART1 波特率分频寄存器	0x0F00_0000
	UART2_BA+0x24	R/W	UART2 波特率分频寄存器	0x0F00_0000

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

Bits	描述	
[31:30]	Reserved	保留
[29]	DIV_X_EN	<p>分频 X 使能 BRD = 波特率分频， 波特率的公式为： 波特率 = Clock / [M * (BRD + 2)]; 其中 M 的默认值为 16。 1 = 使能分频 X (等式中 M = X+1, 但 DIVIDER_X[27:24] 必须 ≥ 8) 0 = 禁用分频 X (等式中 M = 16) 参考下文中的表格可获得更多信息。 注： 当在 IrDA 模式， 该位禁用。</p>
[28]	DIV_X_ONE	<p>Divider X equal 1 1 = Divider M = 1 (等式中的 M = 1, 但 BRD [15:0] 必须 ≥ 3)。 0 = Divider M = X (等式中的 M = X+1,, 但 DIVIDER_X[27:24] 必须 $t \geq 8$) 参考下文中的 表 5-12 获取更多信息。</p>
[27:24]	DIVIDER_X	Divider X 波特率分频 M = X+1.
[23:16]	Reserved	保留
[15:0]	BRD	波特率分频 该域表示波特率分频

模式	DIV_X_EN	DIV_X_ONE	DIVIDER X	BRD	波特率公式
0	Disable	0	Don't care	A	UART_CLK / [16 * (A+2)]
1	Enable	0	B	A	UART_CLK / [(B+1) * (A+2)] , B must >= 8
2	Enable	1	Don't care	A	UART_CLK / (A+2), A must >=7

表 5-12 波特率方程式表

IrDA 控制寄存器 (IRCR)

寄存器	偏移量	R/W	描述	复位后的值
UA_IRCR	UART0_BA+0x28	R/W	UART0 IrDA 控制寄存器	0x0000_0040
	UART1_BA+0x28	R/W	UART1 IrDA 控制寄存器	0x0000_0040
	UART2_BA+0x28	R/W	UART2 IrDA 控制寄存器	0x0000_0040

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

Bits	描述	
[31:7]	Reserved	保留
[6]	INV_RX	INV_RX 1= RX 输入信号反转 0= 无反转
[5]	INV_TX	INV_TX 1= TX 输出信号反转 0= 无反转
[4:2]	Reserved	保留
[1]	TX_SELECT	TX_SELECT 1= 使能 IrDA 发送 0= 使能 IrDA 接收
[0]	Reserved	保留

注：当在 IrDA 模式，UA_BAUD [DIV_X_EN] 寄存器必须禁用。（波特方程必须为 Clock / 16 * (BRD)）

UART 复用控制/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	UART0_BA+0x2C	R/W	UART0 复用控制/状态寄存器	0x0000_0000
	UART1_BA+0x2C	R/W	UART1 复用控制/状态寄存器	0x0000_0000
	UART2_BA+0x2C	R/W	UART2 复用控制/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
Reserved							
15	14	13	12	11	10	9	8
RS485_ADD_EN	Reserved				RS485_AUD	RS485_AAD	RS485_NMM
7	6	5	4	3	2	1	0
LIN_TX_EN	LIN_RX_EN	Reserved		UA_LIN_BKFL			

Bits	描述	
[31:24]	ADDR_MATCH	地址匹配值寄存器 该位包括 RS-485 地址匹配值。 注：该位用于 RS-485 自动地址检测模式。
[23:16]	Reserved	保留
[15]	RS485_ADD_EN	RS-485 地址检测使能 该位用于使能 RS-485 地址检测使能模式。 1 = 使能地址检测模式 0 = 禁用地址检测模式 注：该位用于 RS-485 任意操作模式。
[14:11]	Reserved	保留
[10]	RS485_AUD	RS-485 自动方向模式 (AUD) 1 = 使能 RS-485 自动方向操作模式 (AUO) 0 = 禁用 RS-485 自动方向操作模式 (AUO) 注：在RS-485_AAD 或 RS-485_NMM 操作模式有效。
[9]	RS485_AAD	RS-485 自动地址检测操作模式 (AAD) 1 = 使能 RS-485 自动地址方向操作模式 (AAD) 0 = 禁用 RS-485 自动地址方向操作模式 (AAD)

		注：在 RS-485_NMM 操作模式下无效。
[8]	RS485_NMM	<p>RS-485 普通多点操作模式 (NMM)</p> <p>1 = 使能 RS-485 普通多点操作模式 (NMM) 0 = 禁用 RS-485 普通多点操作模式 (NMM)</p> <p>注：在 RS-485_AAD 操作模式下无效。</p>
[7]	LIN_TX_EN	<p>LIN TX Break Mode Enable</p> <p>1 = 使能 LIN TX Break 模式 0 = 禁用 LIN TX Break 模式</p> <p>注：当 TX break 域传输操作完成后，该位将被自动清除。</p>
[6]	LIN_RX_EN	<p>LIN RX 使能</p> <p>1 = 使能 LIN RX 模式。 0 = 禁用 LIN RX 模式</p>
[5:4]	Reserved	保留
[3:0]	UA_LIN_BKFL	<p>UART LIN Break 域长度</p> <p>该域表示一个 4 位 LIN TX break 域计数。</p> <p>注：break 域长度为 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	UART0_BA+0x30	R/W	UART0 功能选择寄存器	0x0000_0000
	UART1_BA+0x30	R/W	UART1 功能选择寄存器	0x0000_0000
	UART2_BA+0x30	R/W	UART2 功能选择寄存器	0x0000_0000

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

Bits	描述	
[31:2]	Reserved	保留
[1:0]	FUN_SEL	功能选择使能 00 = UART 功能 01 = 使能 LIN 功能 10 = 使能 IrDA 功能 11 = 使能 RS-485 功能



5.13 控制器局域网 (CAN)

5.13.1 概述

C_CAN 由 CAN 内核, 报文 RAM, 报文处理器, 控制寄存器和模块接口(见图 5-76)构成。CAN 内核通信符合 CAN 协议规范 2.0A 和 2.0B。位速率可达到 1MBit/s。为了和物理层相连, 需要另外的收发器硬件。

在 CAN 网络中通信, 各个报文对象是可配置的。报文对象和用于接收报文过滤的标识符掩码存储在报文 RAM 中。所有关于报文处理的功能在报文处理器中执行。这些功能包括接收过滤、CAN 内核与报文 RAM 之间的报文传输和传送请求以及模块中断的产生。

C_CAN 的寄存器组可以通过模块接口被软件直接访问, 这些寄存器用来控制/配置 CAN 内核和报文处理器, 以及访问报文 RAM。

5.13.2 特征

- 支持 CAN 协议规范 2.0 A 和 2.0 B。
- 位速率最大达 1 MBit/s。
- 32 个报文对象。
- 每个报文对象都有自己的标识符掩码。
- 可编程的 FIFO 模式 (链接报文对象)
- 可屏蔽中断。
- 对于时间触发的 CAN 应用可禁用自动重传模式。
- 自检测操作时的可编程环回模式。
- 16-bit 模块接口到 AMBA APB 总线。
- 支持唤醒功能。

5.13.3 框图

C_CAN 和 AMBA 的 APB 总线间的接口。图 5-76 为 C_CAN 的框图。

- **CAN 内核**

CAN 协议控制器和 Rx/Tx 移位寄存器用来进行报文的串/并转换。

- **报文 RAM**

保存报文对象和标识符掩码

- **寄存器**

所有寄存器用于控制和配置 C_CAN。

- **报文处理器**

状态机控制 CAN 内核的 Rx/Tx 移位寄存器和报文 RAM 间的数据传输，以及控制和配置寄存器中设定的中断的产生。

- **模块接口**

C_CAN 和来自ARM的 AMBA APB 16-位总线相接的接口。

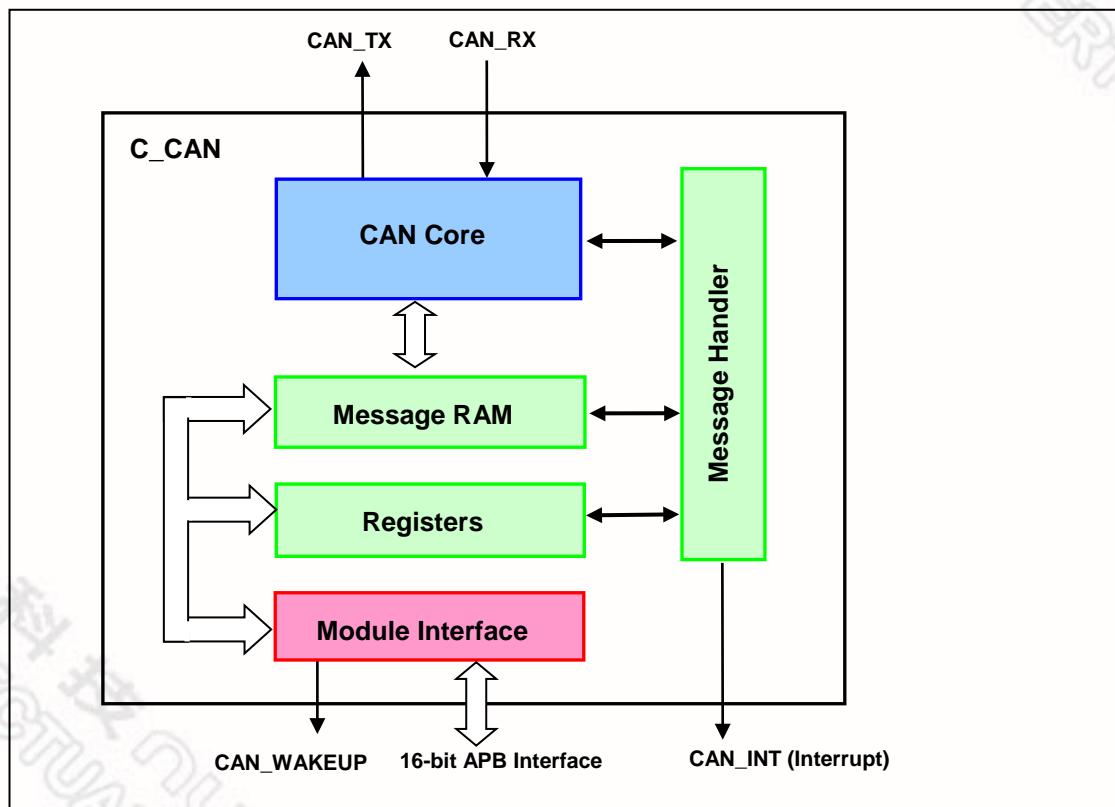


图 5-76 CAN 外设框图

5.13.4 功能描述

5.13.4.4 软件初始化

软件初始化通过设置 CAN 控制寄存器的 **Init** 位，或者由软件或硬件进行复位，或者进入总线关闭状态开始。

当 **Init** 位被置位后，所有的 CAN 总线上的报文传输都被停止，**CAN_TX** 输出管脚上的状态是隐性的（高电平）。错误管理逻辑 (EML) 计数器不变。设置 **Init** 位不会改变任何配置寄存器。

为了初始化 CAN 控制器，软件必须设定位定时寄存器和每个报文对象。如果一个报文对象不是必须的，相应的 **MsgVal** 位必须清除，否则，整个报文对象需要被初始化。

当 CAN 控制寄存器的 **Init** 和 **CCE** 位都被置位时，配置位定时的位定时寄存器和波特率预分频扩展寄存器的访问才被使能。

复位 **Init** 位（只能通过软件）完成软件初始化。然后在参加总线活动和开始报文传输之前需要等待 11 个连续隐性位（总线空闲）的位序列的出现，位流处理器 (BSP)（见章节 5.13.6.10：配置位定时）使自身与总线上的数据传输同步。

报文对象的初始化独立于 **Init**，可以在忙碌时完成。但是在 BSP 开始传输报文之前，报文对象必须配置好特定的标识符。

在正常工作期间去改变一个报文对象的配置时，软件必须复位相应的 **MsgVal** 位然后开始。当配置完成，**MsgVal** 被再一次置位。

5.13.4.5 CAN 报文传输

一旦 C_CAN 被初始化以及 **Init** 位被重置为 0，C_CAN 内核就使自身与 CAN 总线同步，并开始报文传输。

如果接收到的报文通过了报文处理器的接收过滤，将会被存储在相应的报文对象中。保存在报文对象中的整个报文包括所有的仲裁位，DLC 和 8 个字节数据。如果使用了标识符掩码，则被认为“无所谓”的仲裁位可能会在报文对象中被覆盖。

软件可以通过接口寄存器在任何时间去读或写每条报文。报文处理器保证了数据的一致性，以防发生同时存取。

发送的报文由应用软件进行更新。如果一个报文存在永久性的报文对象（仲裁位和控制位在配置时被设定），那么仅有数据字节被更新，**TxRqst** 位和 **NewDat** 位被置位即开始传送。如果几个传输的报文被指派给同一个报文对象（当报文对象的数量不够用时），在发送这些报文前必须对整个报文对象进行配置。

任何数量的报文对象传送可以在同一时间请求。报文对象传送的顺序依据它们的内部优先级。报文可以随时更新或设置为无效，甚至在发送请求还在等待阶段也可进行。一条报文在其未发送前被更新时，则旧的数据将被丢弃。

依据报文对象的配置，通过对一个匹配了标识符的远程帧的接收，可以自动产生报文发送请求。

5.13.4.6 禁用自动重传

依照 CAN 规范（见 ISO11898, 6.3.3 恢复管理），C_CAN 为那些在传送过程中丢失仲裁或被错误干扰的帧，提供了自动重传的机制。在传送完全成功之前，帧传送服务是不能被用户证实的。也就意味着，默认情况下，自动重传是使能的。自动重传也可以被禁用以用来使能 C_CAN 工作在时间触发 CAN (TTCAN, 见 ISO11898-4) 环境。

通过设置 CAN 控制器中的禁用自动重传 (**DAR**) 位为 1 来禁用自动重传模式。在这种操作模式下，用户必须考虑报文缓存控制寄存器中 **TxRqst** 和 **NewDat** 位的不同情况：

- 当传送开始时，各个报文缓存的 **TxRqst** 位被清除，而 **NewDat** 位一直置位。
- 当传送完全成功时，**NewDat** 被清除。
- 当传送失败（丢失仲裁或错误），**NewDat** 位保持置位。

为了重新开始传送，软件必须再将 **TxRqst** 位置位。

5.13.5 测试模式

设定 CAN 控制寄存器的 **Test** 位进入测试模式。在测试模式中，测试寄存器的 **Tx1**, **Tx0**, **LBack**, **Silent** 和 **Basic** 位可写。**Rx** 位监视 **CAN_RX** 管脚的状态，因此该位只读。当 **Test** 位被清除时，所有的测试寄存器功能被禁用。

5.13.5.1 静默模式 (*Silent Mode*)

通过设置测试寄存器中的 **Silent** 位为1，CAN 内核可被设为 Silent 模式。在 Silent 模式中，**C_CAN** 能够接收有效的数据帧和有效的远程帧，但是它在 CAN 总线上仅发送隐性位。而且它不能启动发送。如果 CAN 内核被请求去发送一个显性位（ACK 位，错误帧），该位在内部自动改道以便 CAN 内核能监视该显性位，而 CAN 总线可能仍保持在隐性状态。Silent 模式可以用来分析 CAN 总线的运输状况，当传送显性位时不影响 CAN 总线。图 5-77 为 Silent 模式下，**CAN_TX** 和 **CAN_RX** 与 CAN 内核的连接信号。

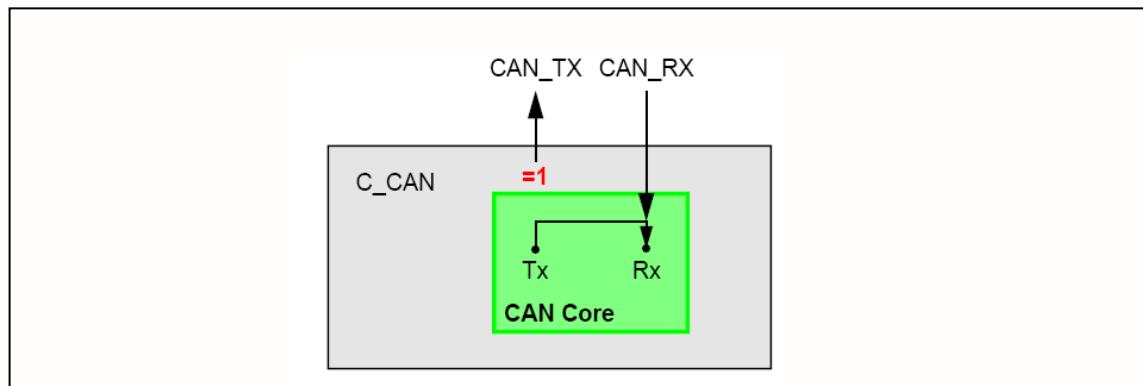
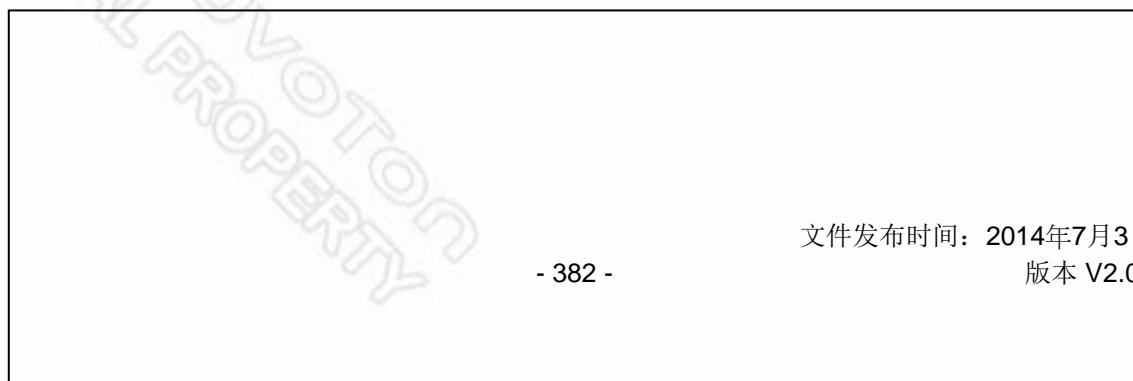


图 5-77 Silent 模式下的 CAN 内核

5.13.5.2 环回 (*Loop Back*) 模式

通过设定测试寄存器的 **LBack** 位为1，CAN 能够被设为 Loop Back 模式。在 Loop Back 模式下，CAN 内核把自己发送的报文作为接收到的报文对待，并将它们保存在接收缓存中（如果它们通过了接收过滤）。图 5-79 为 Loop Back 模式下，**CAN_TX** 和 **CAN_RX** 与 CAN 内核的连接信号。



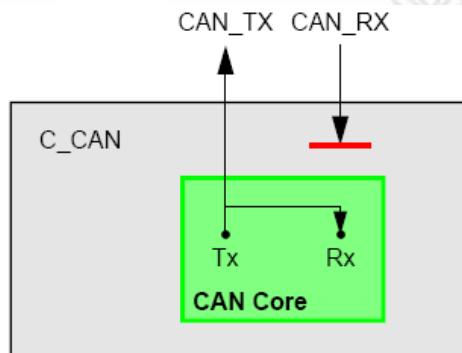


图 5-78 Loop Back 模式下的 CAN 内核

该模式提供自检功能。为了不受外部的干扰，在 Loop Back 模式下，CAN 内核忽略应答错误（从数据/远程帧应答场内采样的隐性位）。在该模式下，CAN 内核从自己的 Tx 输出执行一个内部反馈到它的 Rx 输入。**CAN_RX** 输入管脚的真实值被 CAN 内核忽略了。发送的报文能够在 **CAN_TX** 管脚上监视。

5.13.5.3 环回模式和静默模式的整合 (Loop Back Combined with Silent Mode)

通过设定 **LBack** 和 **Silent** 位同时为1，还可能将 Loop Back 模式和 Silent 模式结合一起使用。该模式可用于“Hot Selftest”，意味着 C_CAN 能在不影响连接到**CAN_TX** 和 **CAN_RX** 管脚的 CAN 系统的情况下被测试。在该模式，**CAN_RX** 管脚被从 CAN 内核断开，**CAN_TX** 管脚保持隐性。图 5-79 为环回模式和静默模式整合下，**CAN_TX** 和 **CAN_RX** 与 CAN 内核的连接信号。

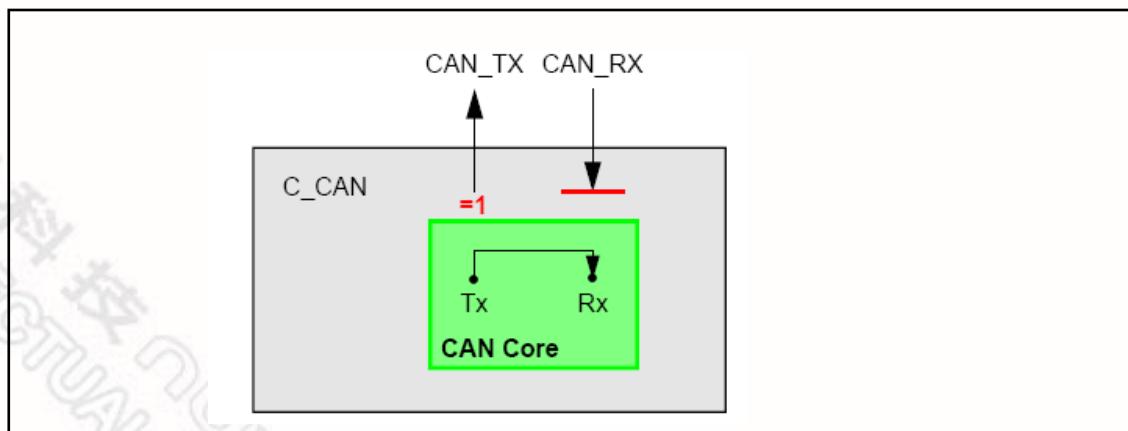


图 5-79 环回模式和静默模式整合下的 CAN 内核

5.13.5.4 基本 (Basic) 模式

通过设定测试寄存器的 **Basic** 位为 1 可以设定 CAN 内核工作在 Basic 模式。该模式下，C_CAN

文件发布时间：2014年7月3日

工作时没有报文 RAM。

IF1 寄存器用作发送缓存。设定 **IF1** 命令请求寄存器中的 **Busy** 位为 1，**IF1** 寄存器中内容请求发送。当 **Busy** 位置位时，**IF1** 寄存器锁定。**Busy** 位指示传送正在挂起。

一旦 CAN 总线空闲，**IF1** 寄存器会被载入到 CAN 内核的移位寄存器，然后传送开始。当传输完成时，**Busy** 位复位，并释放锁定的 **IF1** 寄存器。

当 **IF1** 寄存器锁定时，挂起的发送可以被随时中止，方法为复位 **IF1** 寄存器的 **Busy** 位。如果软件已经复位了 **Busy** 位，那么防止仲裁丢失或错误出现的重传是被禁用的。

IF2 寄存器用作接收缓存。收到报文后，移位寄存器的内容存储到 **IF2** 寄存器，且不经过接收过滤。

此外，在报文传输的过程中，移位寄存器的真实内容能够被监视。每次通过写 **IF2** 命令请求寄存器中的 **Busy** 位为 1 时，读报文对象都会被初始化，移位寄存器的内容被保存在 **IF2** 寄存器。

在 Basic 模式下，所有报文对象的相关的控制和状态位以及 **IFn** 掩码寄存器的控制位赋值都被关闭。不会计算命令请求寄存器的报文编号。**IF2** 报文控制寄存器的 **NewDat** 和 **MsgLst** 位保留其功能。**DLC3-0** 指示接收到的 **DLC**，其他控制位读取值皆为 ‘0’。

5.13.5.5 软件控制 CAN_TX 管脚

CAN 发送管脚 **CAN_TX** 上有四种输出功能可用。除了它的缺省功能（串行数据输出），CAN 发送管脚还能驱动 CAN 采样点信号用来监视 CAN 内核的位定时，还能驱动显性或隐性的常量值。后两种功能，结合可读的 CAN 接收管脚 **CAN_RX**，能够用于检测 CAN 总线的物理层。

CAN_TX 管脚的输出模式通过设定 CAN 测试寄存器的 **Tx1** 和 **Tx0** 位进行选择。

3 种 **CAN_TX** 管脚测试功能和所有的 CAN 协议功能有干扰，当 CAN 报文传输或者选择任一测试模式(Loop Back Mode, Silent Mode, 或 Basic Mode) 时，**CAN_TX** 必须使用它的默认功能。

5.13.6 CAN 通信

5.13.6.1 管理报文对象

报文 RAM 中的报文对象配置（除 CAN 控制寄存器的 **MsgVal**, **NewDat**, **IntPnd**, 和 **TxRqst** 位外）将不会被芯片复位影响。在应用软件清除 **Init** 位之前，所有的报文对象必须被应用软件初始化或者是“无效”(**MsgVal** = ‘0’) 的，且位定时必须被配置好。

当设置其中之一的接口寄存器的屏蔽、仲裁、控制和数据域为期望值后，报文对象的配置即完成了。通过写相应的**IFn** 命令请求寄存器，可将 **IFn** 报文缓存寄存器载入到报文 RAM 中被寻址的报文对象。

当 CAN 控制寄存器的 **Init** 位被清除，CAN 内核的 CAN 协议控制状态机和报文处理器状态机控制 **C_CAN** 的内部数据流。那些通过接收过滤而收到的报文被保存在报文 RAM 中。挂起的报文发送请求被载入到 CAN 内核的移位寄存器，并通过 CAN 总线进行发送。

应用软件通过 **IFn** 接口寄存器读取收到的报文以及更新将被发送的报文。依照配置，应用软件会被特定的 CAN 报文和 CAN 错误事件打断。

5.13.6.2 报文处理器状态机

报文处理器控制着 CAN 内核的 Rx/Tx 移位寄存器、报文 RAM 和 **IFn** 寄存器之间的数据传输。

报文处理器 FSM 控制如下功能：

- 从 IFn 寄存器到报文 RAM 的数据传输
- 从报文 RAM 到 IFn 寄存器的数据传输
- 从移位寄存器到报文 RAM 的数据传输
- 从报文 RAM 到移位寄存器的数据传输
- 从移位寄存器到验证过滤单元的数据传输
- 扫描报文 RAM 寻找匹配报文对象
- TxRqst 标志位处理
- 中断处理

5.13.6.2.1 报文 RAM 的数据传输

当应用软件初始化 IFn 寄存器和报文 RAM 间的数据传输时，报文处理器设置各自命令请求寄存器 (CAN_IFn_CRR) 的 Busy 位为 ‘1’。在传输完成后，Busy 位再次被清除（见图 5-80）。

各个命令掩码寄存器指定是一个完整的报文对象还是报文对象的部分将被传输。因为报文 RAM 的结构，无法写一个报文对象的单独的位/字节。总是必须写入完整的报文对象到报文 RAM。因此，从 IFn 寄存器到报文 RAM 的数据传输需要一个读-修改-写周期。首先，报文对象的那些不变的部分从报文 RAM 读出，然后报文缓存寄存器的完整内容必须写入报文对象。

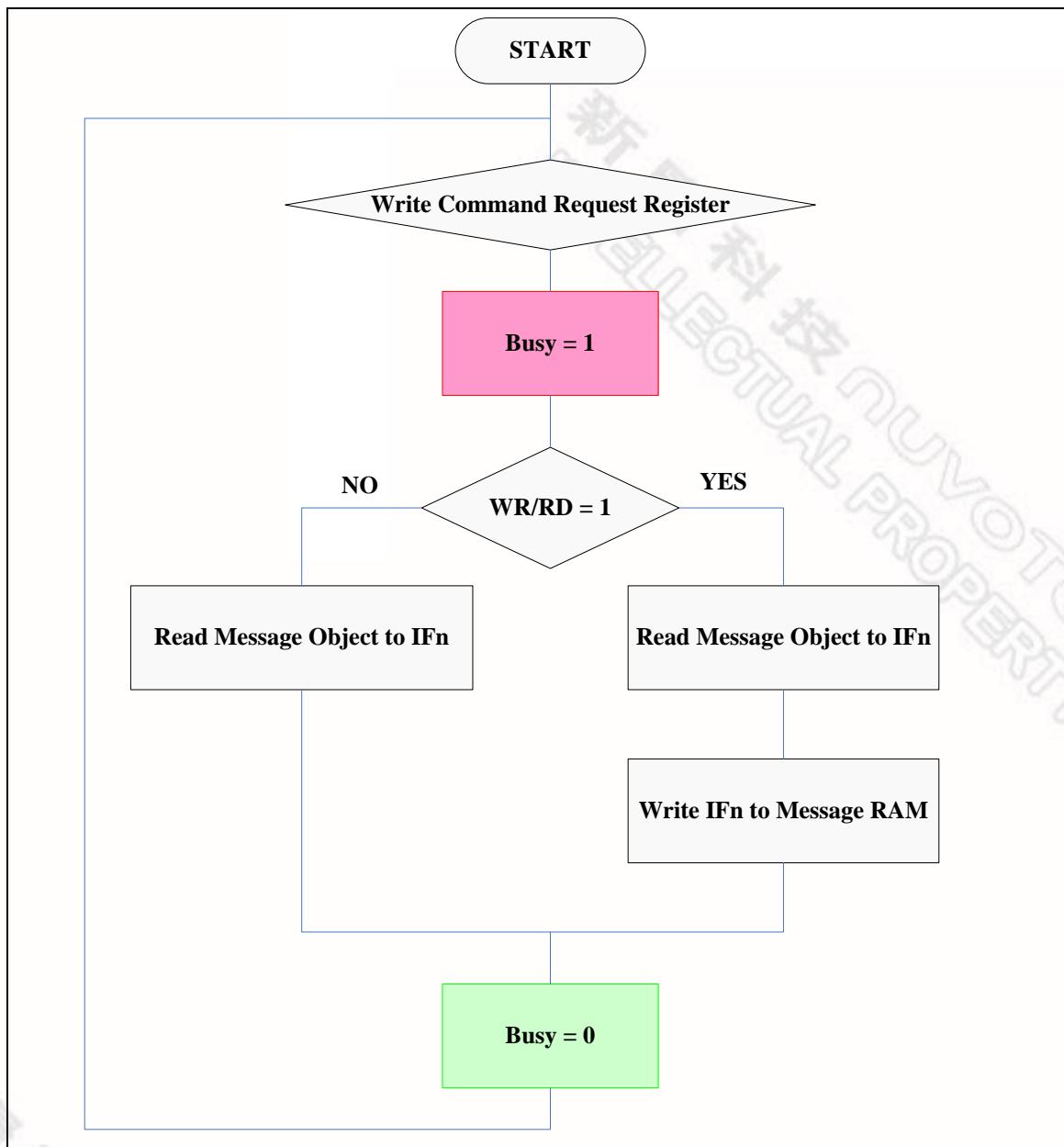


图 5-80 IFn 寄存器和报文间的数据传输

在一个报文对象部分写入后，没被命令掩码寄存器选中的报文缓存寄存器将设置选中的报文对象的真正内容。

在一个报文对象部分读取后，没被命令掩码寄存器选中的报文缓存寄存器将保持不变。

5.13.6.2.2 报文传送

如果 CAN 内核的移位寄存器单元准备载入，而且在 IFn 寄存器和报文 RAM 之间没有数据传输，则将会检查报文有效寄存器的 **MsgVal** 位和传送请求寄存器的 **TxRqst** 位。具有最高优先级的挂起发送请求的有效报文对象被报文处理器载入到移位寄存器，发送开始。报文对象的 **NewDat** 位被复位。

成功传送后或者如果自传送开始后没有新的数据写入报文对象 (**NewDat** = '0')，报文控制寄存器 (CAN_IFn_MCR) 的 **TxRqst** 位将被复位。如果报文控制寄存器 (CAN_IFn_MCR) 的 **TxEIE** 位被置位，中断标识符寄存器的 **IntPnd** 位在传送成功后将被置位。如果 **C_CAN** 丢失仲裁或者在传送过程中有错误发生，报文将会在 CAN 总线空闲的时候立即重传。同时，如果有更高优先级的报文传送被请求，则报文将会按照报文的优先级顺序进行传送。

5.13.6.2.3 收到报文的接收过滤

当一个输入的报文的仲裁和控制域 (Identifier + IDE + RTR + DLC) 完全被移位到 CAN 内核的 Rx/Tx 移位寄存器时，报文处理器 FSM 开始扫描报文 RAM 寻找匹配的有效报文对象。

扫描报文 RAM 寻找匹配的报文对象，要将 CAN 内核移位寄存器的仲裁位载入到接收过滤单元。报文对象 1 的仲裁和掩码域（包括 **MsgVal**, **UMask**, **NewDat**, 和 **EoB**）然后被载入到接收过滤单元和移位寄存器的仲裁位进行比较。该步骤对接下来的每一个报文对象重复进行，直到匹配的报文对象出现或者到达报文 RAM 的末端。

如果匹配发生，扫描停止，报文处理器 FSM 的处理取决于接收帧的类型（数据帧或远程帧）。

数据帧的接收

报文处理器 FSM 将 CAN 内核移位寄存器中的报文保存到报文 RAM 中各自的报文对象。不仅数据字节，而是所有仲裁位和数据长度码都被保存到相应的报文对象。这是用于数据字节和标识符保持联系，即使使用了仲裁掩码寄存器。

NewDat 位被置位用于指示新数据（还没有被软件看到的）已经收到了。当报文对象被读取后，应用软件必须复位 **NewDat** 位。如果在接收的时候，**NewDat** 位已经被置位，**MsgLst** 用来指示之前的数据（假定还没有被软件看到）已丢失。如果 **RxEIE** 位被置位，**IntPnd** 位也置位将使得中断寄存器指向该报文对象。

当请求的数据帧刚刚被收到时，报文对象的 **TxRqst** 位被复位用来阻止远程帧的传送。

远程帧的接收

当收到一个远程帧时，匹配的报文对象的三种不同配置必须考虑：

1) Dir = '1' (direction = transmit), **RmtEn** = '1', **UMask** = '1' 或 '0'

当收到匹配的远程帧时，报文对象的 **TxRqst** 位被置位。报文对象的其他部分保持不变。

2) Dir = '1' (direction = transmit), **RmtEn** = '0', **UMask** = '0'

当收到匹配的远程帧时，报文对象的 **TxRqst** 位保持不变。远程帧被忽略。

3) Dir = '1' (direction = transmit), **RmtEn** = '0', **UMask** = '1'

当收到匹配的远程帧时，报文对象的 **TxRqst** 位被置位。移位寄存器中的仲裁和控制域 (Identifier + IDE + RTR + DLC) 被保存到报文 RAM 中的报文对象，报文对象的 **NewDat** 位被置位。报文对象的数据域保持不变。处理远程帧和处理收到的数据帧方式类似。

5.13.6.2.4 接收/发送优先级

报文对象接收/发送的优先级与报文号相关。报文对象 1 拥有最高的优先级，报文对象 32 拥有最低

的优先级。如果有一个以上的发送请求挂起时，它们将被根据相应报文对象的优先级来进行服务。

5.13.6.3 配置传送对象

表 5-13 展示如何初始化一个发送对象。

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

表 5-13 一个发送对象的初始化

注：appl. = 应用软件。

仲裁寄存器的值（**ID28-0** 和 **Xtd** 位）由应用软件提供。它们定义标识符和发出的报文类型。如果 11-位 标识符（标准帧）被使用，它将被编程到 ID28 - ID18。ID17 - ID0 可以被忽略。

如果 **TxIE** 位被置位，在报文对象成功发送之后，**IntPnd** 位将被置位。

如果 **RmtEn** 位被置位，匹配收到的远程帧将导致 **TxRqst** 位被置位，而且该远程帧会自动被一个数据帧应答。

数据寄存器的值（**DLC3-0**, **Data0-7**）由应用软件提供。在数据有效之前，**TxRqst** 和 **RmtEn** 可能不会被置位。

掩码寄存器（**Msk28-0**, **UMask**, **MXtd**, 和 **MDir** 位）可以被用于（**UMask='1'**）允许有相似标识符的远程帧组设置 **TxRqst** 位。Dir 位不能被屏蔽。

5.13.6.4 更新传送对象

软件可以通过 **IFn** 接口寄存器随时更新发送对象的数据字节，在更新前 **MsgVal** 和 **TxRqst** 不必复位。

即使只有数据字节的一部分要更新，**IFn** 数据 A 寄存器或 **IFn** 数据 B 寄存器的所有 4 字节在这些寄存器的内容传输到报文对象前必须有效。应用软件必须写所有 4 个字节到 **IFn** 数据寄存器或者在软件写新的数据字节前报文对象被传输到 **IFn** 数据寄存器。

当仅 8 个(eight) 数据字节更新时，首先 0x0087 被写入到命令掩码寄存器，然后报文对象的号码被写入到命令请求寄存器，同时更新数据字节及设置 **TxRqst**。

当数据更新时，为了防止在进行中的传输的最后阶段 **TxRqst** 复位，**NewDat** 必须和 **TxRqst** 一起被置位。

当 **NewDat** 和 **TxRqst** 一起被置位时，**NewDat** 将会在新的传送开始时立即被复位。

5.13.6.5 配置接收对象

表 5-14 为接收对象如何初始化。

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

表 5-14 接收对象的初始化

仲裁寄存器的值（**ID28-0** 和 **Xtd** 位）由应用软件提供。它们定义标识符和接受的报文的类型。如果 11-位 标识符（“标准帧”）被使用，它会被编程到 ID28 - ID18。ID17 - ID0 可以被忽略。当带有 11-位 标识符的数据帧被收到时，ID17 - ID0 将被置‘0’。

如果 **RxIE** 位置位，当收到的数据帧被接受而且保存到报文对象时，**IntPnd** 位将被置位。

数据长度码 (DLC3-0) 由应用软件提供。当报文处理器保存一个数据帧到报文对象时，它将会保存收到的数据长度码和 8 数据字节。如果数据长度码少于 8，则报文对象余下的字节将被未指定的值覆盖。

掩码寄存器 (**Msk28-0**, **UMask**, **MXtd**, 和 **MDir** 位) 可以被用于 (**UMask='1'**) 允许有相似标识符的数据帧组被接受。在典型的应用中，**Dir** 位不能被屏蔽。

5.13.6.6 处理接收报文

应用软件可以通过 **IFn** 接口寄存器随时去读一条收到报文。报文处理器的状态机保证数据的一致性。

通常的，软件先写 0x007F 到命令掩码寄存器，然后是报文对象号写入命令请求寄存器。这个结合体将会把整个收到的报文从报文 RAM 传输到报文缓存寄存器。此外，**NewDat** 和 **IntPnd** 位将在报文 RAM 中被清除（不是报文缓存）。

如果报文对象使用掩码进行接收过滤，则仲裁位表示哪条匹配的报文已经收到了。

NewDat 的实际值（当前值）表示自上次该报文对象被读取后，是否收到新的报文。**MsgLst** 的实际值表示自上次该报文对象被读取后，收到的报文是否多于 1 条。**MsgLst** 不会被自动复位。

利用远程帧，软件可以请求另一个 CAN 节点为收到的对象提供新的数据。设定收到对象的 **TxRqst** 位将导致带有收到对象的标识符的远程帧的传送。该远程帧触发另一个 CAN 节点开始发送与之匹配的数据帧。如果在发送远程帧之前收到匹配的数据帧，**TxRqst** 位将被自动复位。

5.13.6.7 配置 FIFO 缓存

除 **EoB** 位之外，属于 FIFO 缓存的接收对象的配置和单独接收对象的配置是一样的。参照章节 5.13.6.5：配置接收对象。

为连接 2 个或更多报文对象到一个 FIFO 缓存，这些报文对象的标识符和掩码（如使用）必须编程设为要匹配的值。因为报文对象内含的优先级，所以带有最低号码的报文对象将会是 FIFO 缓存的第一个报文对象。FIFO 缓存中（除了最后的报文对象外的）所有报文对象的 **EoB** 位都必须被编程设为 0。FIFO 缓存中的最后一个报文对象的 **EoB** 位设为 1，配置它作为模块的结束。

5.13.6.8 接收带 FIFO 缓存的报文

收到的带有匹配 FIFO 缓存标识符的报文会被保存在该 FIFO 缓存的报文对象中，从最低报文号的报文对象开始。

当一条报文保存到 FIFO 缓存的一个报文对象中时，该报文对象的 **NewDat** 位被置位。当 **EoB** 为 0 时通过置位 **NewDat**，报文对象被锁定用于将来的报文处理器的写访问，直到应用软件将 **NewDat** 位写回 0。

报文被保存在 FIFO 缓存中直到达到了该 FIFO 缓存的最后一个报文对象。如果之前没有一个报文对象通过写 **NewDat** 位为 0 释放，则该 FIFO 缓存将来的所有的报文将被写入该 FIFO 缓存的最后一个报文对象中，因此之前的报文会被覆盖。

5.13.6.8.1 从 FIFO 缓存读

当应用软件通过写报文对象的编号到 **IFn** 命令请求寄存器来传输该报文对象的内容到 **IFn** 报文缓存寄存器时，相应的命令掩码寄存器须按如下方式编程，**NewDat** 和 **IntPnd** 位复位为零 (**TxRqst/NewDat** = '1' 和 **CrlntPnd** = '1')。报文控制寄存器的各位总是反映了这些位在被复位之前其状态。

为保证 FIFO 缓存的正确功能，应用软件须从 FIFO 缓存中最低报文号的报文对象读起。

图 5-81 表示应用软件如何处理一组和 FIFO 缓存相连的报文对象。

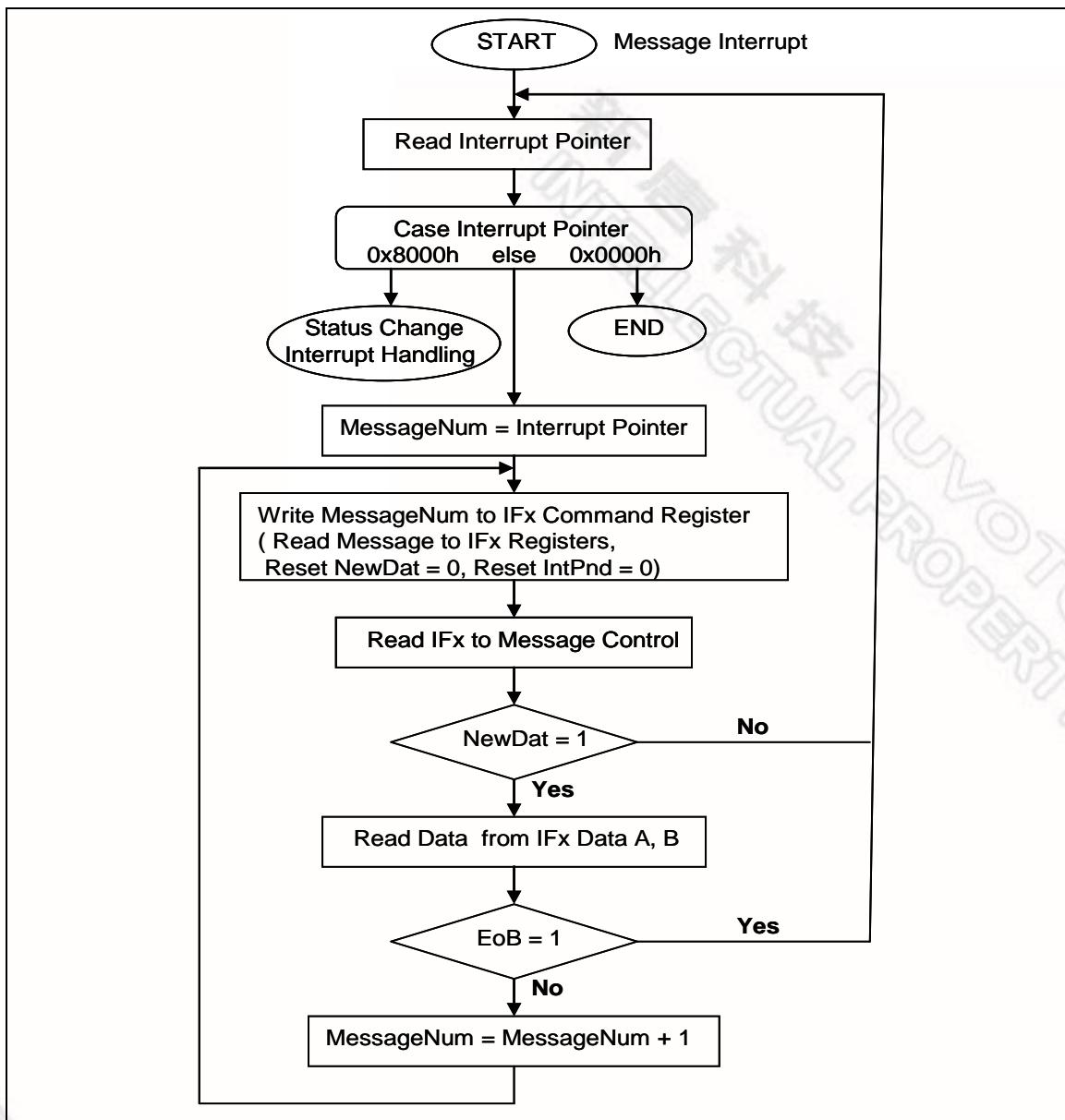


图 5-81 应用软件处理 FIFO 缓存

5.13.6.9 中断处理

如果一些中断挂起，则 CAN 的中断寄存器将指向具有最高优先级的挂起中断，而无需理会它们的时间顺序。一个中断会保持挂起状态直到应用软件清除它。

状态中断拥有最高优先级。在报文中断中，报文对象的中断优先级随着报文号的增加而降低。

报文中断通过清除报文对象的 **IntPnd** 位进行清除。状态中断通过读状态寄存器清除。

中断寄存器的中断标识符 **IntId** 指示引发中断的原因。当没有中断挂起时，寄存器将保持为 0。如果中断寄存器的值不为 0，则有中断挂起。如果 **IE** 被置位，则 **CAN_INT** 中断信号被激活。中断保持激活直到中断寄存器归零（相应的中断被复位）或直到 **IE** 被复位。

值 0x8000 指示有中断正被挂起，原因是 CAN 内核更新了（不是必须改变的）状态寄存器（错误中断或状态中断）。该中断具有最高优先级。应用软件可以更新（复位）状态位 RxOk, TxOk 和 LEC，但是软件对状态寄存器的写访问不会产生或复位中断。

其他的值表示中断源是其中一个报文对象。IntId 指向挂起的具有最高中断优先级的报文中断。

当中断寄存器的值不等于零时（CAN 控制寄存器的 IE 位），应用软件控制状态寄存器的改变是否可能导致一个中断（CAN 控制寄存器的 EIE 和 SIE 位）和中断线是否会激活。甚至当 IE 复位时，中断寄存器仍将会更新。

应用软件追踪报文中断源有 2 种可能方式。一种是它能够查看中断寄存器的 IntId 位，第二种是它可以轮询中断挂起寄存器。

中断处理服务程序读作为中断源的报文可以同时读报文和复位报文对象的 IntPnd 位（命令掩码寄存器的 ClrIntPnd 位）。当 IntPnd 被清除，中断寄存器将指向下一个带有挂起中断的报文对象。

5.13.6.10 配置位定时

即使 CAN 位定时配置的次要错误不会导致立即失败，但是 CAN 网络的表现会在很大程度上降低。

很多情况下，CAN 位同步将修正 CAN 位定时的错误配置达到仅仅是偶尔会产生一个错误帧的水平。但是，在进行仲裁的情况下，当两个或更多 CAN 节点同时试着发送一帧，那么一个错位的采样点可能导致其中一个传送会被动的出现错误。

分析此类独立的错误需要对 CAN 节点内 CAN 位同步以及 CAN 总线上 CAN 节点间相互作用有详细的了解。

5.13.6.10.1 位时间和位速率

CAN 支持的位速率范围是低于 1 kBit/s 最高到 1000 kBit/s。CAN 网络中的每个成员都有自己的时钟发生器，通常是石英振荡器。位时间的定时参数（比如位速率的倒数）可以单独配置给每个 CAN 节点，这样尽管 CAN 节点的振荡器周期 (f_{osc}) 可能不同，仍可创建一个公共的位速率。

这些振荡器的频率不是绝对的稳定，温度或电压的变化或者因为组件恶化会导致一些小的变化。只要这种变化保持在一个特定的振荡器容差范围 (df) 内，CAN 节点能够通过重新同步不同的位速率到位流进行补偿。

依据 CAN 规范，位时间分为四段（见 图 5-82）：同步段，传播时间段，相位缓存段1 和相位缓存段2。每段由一个特定的，可编程数的时间片组成（见 表 5-15）。时间片的长度 (t_q)，是位时间的基本时间单元，由 CAN 控制器的 APB 时钟 f_{APB} 和位定时寄存器 (CAN_BTR) 的 BRP 位定义： $t_q = BRP / f_{APB}$ 。

同步段 Sync_Seg，是位时间的一部分，是 CAN 总线电平跳变边沿发生的地方。Sync_Seg 之外发生跳变边沿和 Sync_Seg 之间的距离称为边沿的相位误差。传播时间段 Prop_Seg 用于补偿 CAN 网络内的物理延时时间。相位缓存段 Phase_Seg1 和 Phase_Seg2 围绕着采样点。

（重）同步跳转宽度 (SJW) 定义了重同步可能将采样点在相位缓存段定义的范围内移动的距离，此用于边沿相位误差的补偿。

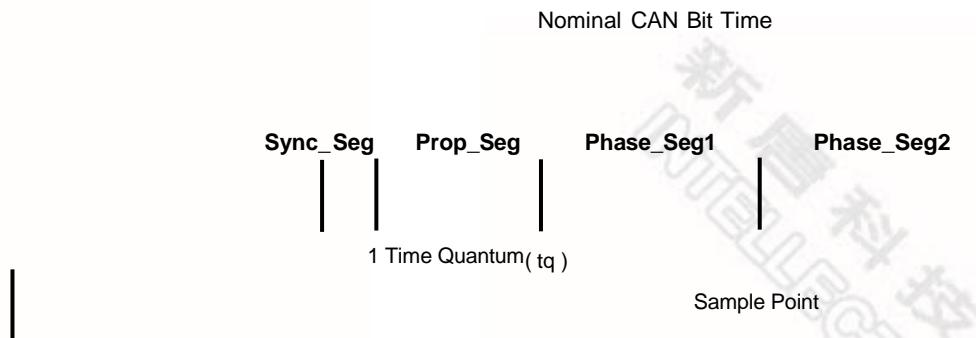


图 5-82 位时间

参数	范围	说明
BRP	[1 .. 32]	定义时间片的长度 t_q
Sync_Seg	$1 t_q$	固定长度，同步总线输入到 APB 时钟
Prop_Seg	[1.. 8] t_q	补偿物理延时时间
Phase_Seg1	[1..8] t_q	可由同步暂时拉长
Phase_Seg2	[1.. 8] t_q	可由同步暂时缩短
SJW	[1 .. 4] t_q	不会比任一个相位缓存段长

该表格描述了 CAN 协议要求的最小可编程范围

表 5-15 CAN 位时间参数

一个给定的位速率可以碰到不同的位时间配置，但是对于 CAN 网络的功能正常，物理延时时间和振荡器容错范围必须考虑。

5.13.6.10.2 传播时间段

该部分位时间用于补偿网络的物理延时时间。这些延时时间由总线上的信号传播时间和 CAN 节点的内部延时时间组成。

CAN 总线上的任何 CAN 节点到位流的同步将会和位流的传送不协调，这是由两个节点之间的传播时间造成的。CAN 协议的非破坏性的逐位仲裁和显性响应位，该位由 CAN 报文的接收器提供，该位要求正在发送位流的 CAN 节点必须也能够接收其他同步到该位流的 CAN 节点发送的显性位。图 5-83 中的示例表示两个 CAN 节点间的相位移动和传播时间。

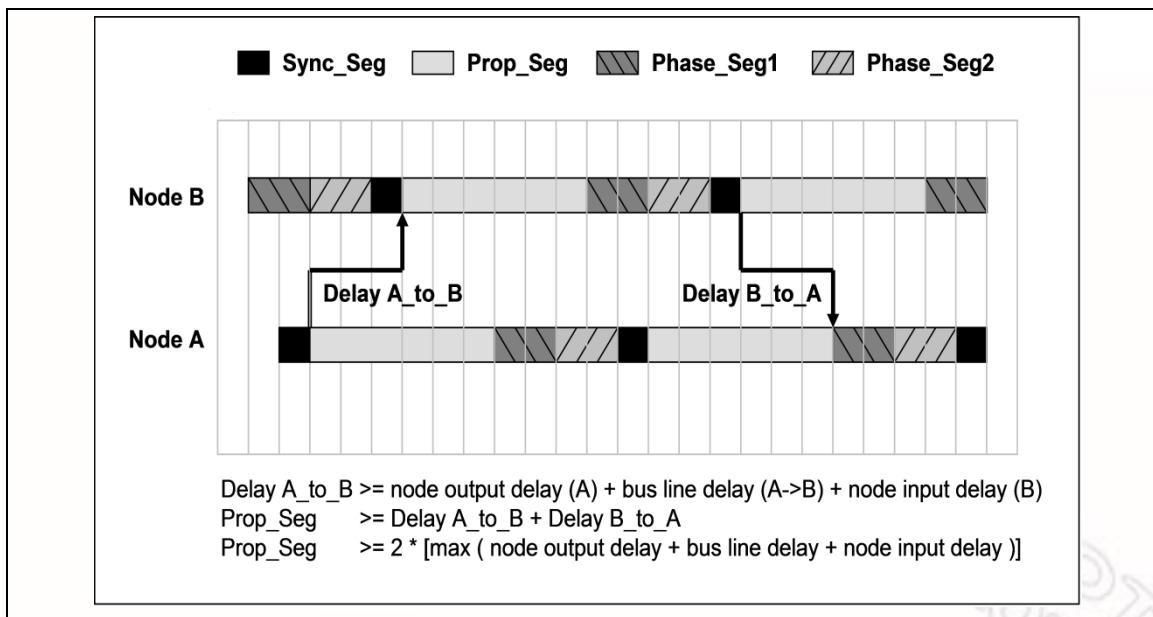


图 5-83 传播时间段

该示例中，节点 A 和 B 都是发送器，为 CAN 总线执行仲裁。节点 A 发送它的起始帧位早于节点 B 不到 1 个位时间，因此节点 B 同步自身到接收边沿（从隐性到显性的边沿）。因为 B 节点在边沿时发送之后已经收到了这个边沿时，B 的位时间段依据 A 转移。B 节点发送了一个高优先级的标识符，所以在 A 发送一个隐性位，它传送一个显性位时，它将在指定的标识符获得仲裁。节点 B 发送的显性位将在节点 A 到达后延时 (B_to_A) 到达。

由于振荡器的容差，节点 A 的采样点实际位置可以在节点 A 相位缓存段标称的任何位置。所以节点 B 的位传送必须在 Phase_Seg1 开始之前到达节点 A。该条件定义 Prop_Seg 的长度。

如果节点 B 传输的（隐形到显性）边沿在 Phase_Seg1 开始后到达节点 A，那么会出现节点 A 采样了一个隐性位代替了显性位的情况，这会导致位错误和错误标志位引起的当前帧的破坏。

这种错误只在当两个节点都对 CAN 总线进行仲裁，这两个节点的振荡器分别处于容差范围相对两端，而且被长总线分开的情况下才会发生。这是一个位定时配置中的次要错误 (Prop_Seg to short) 导致偶发总线错误的例子。

一些 CAN 硬件提供了一个可选择的3次采样模式，但是 C_CAN 没有。在该模式下，CAN 总线的输入信号通过一个低通滤波器，使用 3 个样本和多数逻辑来决定有效位的值。这会导致额外的 1 个输入延时 t_q ，需要更长的 Prop_Seg。

5.13.6.10.3 相位缓存段和同步

相位缓存段 (Phase_Seg1 和 Phase_Seg2) 和同步跳转宽度 (SJW) 用于振荡器误差补偿。通过同步，相位缓存段可以变长或缩短。

同步发送在从隐性到显性的边沿，目的是用来控制边沿和采样点之间的距离。

边沿检测通过采样每个时间片的总线电平当前值进行，并比较该值和前一个采样点的总线电平。同步仅当在前一个采样点一个隐性位被采样，而且当前时间片的总线电平是显性时，同步才可以实

行。

如果边沿发生在 Sync_Seg 里面，那么它是同步，否则边沿和 Sync_Seg 结束处之间的距离是边沿相位误差。如果边沿在 Sync_Seg 之前发生，相位误差是负的，其他情况则是正的。

有两个类型的同步存在：硬同步和重同步。

在帧开始时，总线会进行一次硬同步。只有当重同步发生时，才在帧内进行。

- **硬同步**

硬同步之后，位时间将在 Sync_Seg 的结束处重新开始，而无需关注边沿相位误差。因此硬同步强迫导致硬同步的边沿处于重新开始的位时间同步段内。

- **位重同步**

重同步导致位时间的缩短和延长，以致采样点的位置因为边沿而被移动。

当造成重同步的边沿相位误差是正的，Phase_Seg1 延长。如果相位误差的值少于 SJW，Phase_Seg1 由相位误差的值延长，否则由 SJW 延长。

当造成重同步的边沿相位误差是负的，Phase_Seg2 缩短。如果相位误差的值少于 SJW，Phase_Seg2 由相位误差的值缩短，否则由 SJW 缩短。

当边沿相位误差的值小于或者等于 SJW 的设定值，硬同步和重同步作用相同。如果相位误差大于 SJW，则重同步无法完全补偿相位误差，仍有误差（相位误差 - SJW）存在。

在两个采样点间只能完成一个同步。同步维持一个在边沿和采样点间最小的距离，给总线电平稳定和过滤出小于 (Prop_Seg + Phase_Seg1) 的峰值的时间。

除了噪声脉冲，多数同步是由仲裁导致的。所有节点会“硬”同步于“领先”收发器（先开始发送的收发器）传送的边沿，但是由于传播延时时间，这些节点不能达到理想的同步。“领先”收发器不必获得仲裁，因此各接收器必须同步自身到不同的后来的“领先”收发器，这不同于同步到之前的“领先”发生器。同样的情况也发生在响应域，收发器和一些接收器将必须要与在传送显性响应位时“夺得领先”的接收器取得同步。

当收发器和接收器振荡器时钟周期的差异点在同步时间内（最多10 位）累加时，在仲裁最后的同步将由振荡器误差造成。这些累加的差异不能比 SJW 长，限定了振荡器误差的范围。

图 5-84 为相位缓存段是如何用于相位误差补偿的示例。每两个连续的位定时有 3 副图。上面一副为“late”边沿的同步，下面一副为“early”边沿的同步，中间一副为无同步的参照。

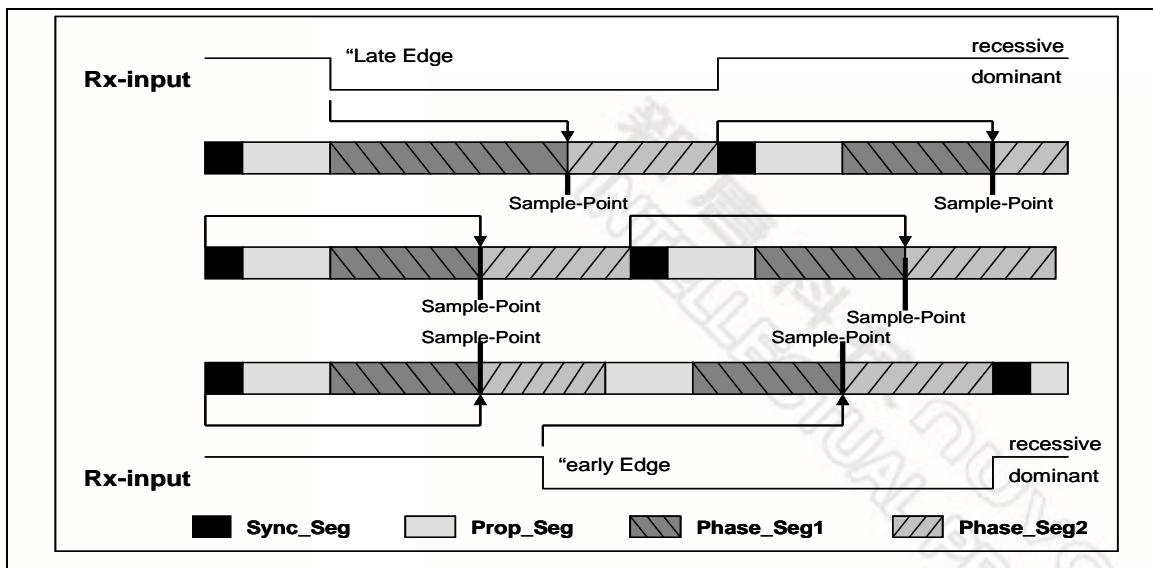


图 5-84 “late” 和 “early” 边沿的同步

在第一个示例中为发生在 Prop_Seg 尾端从隐性到显性的边沿。该边沿为“late”，因为它发生在 Sync_Seg 之后。作为对“late”边沿的反应，Phase_Seg1 被延长，因此从边沿到采样点的距离和在没有边沿发生情况下 Sync_Seg 到采样点本该有的距离一样。“late”边沿的相位误差小于 SJW，所以能被完全补偿，在该位（一个标称的位时间长度）结束处从显性到隐性的边沿，发生在 Sync_Seg 内。

在第二个示例中为发生在 Phase_Seg2 中的从隐性到显性的边沿。该边沿为“early”，因为它发生在 Sync_Seg 之前。作为对“early”边沿的反应，Phase_Seg2 被缩短，Sync_Seg 被忽略，因此从边沿到采样点的距离和在没有边沿发生情况下 Sync_Seg 到采样点本该有的距离一样。和上例相同，该“early”边沿的相位误差小于 SJW，所以能被完全补偿。

相位缓存段被延长或缩短只是暂时的，在下一个位时间，该段将恢复成它们标称设置的值。

在这些例子中，位时间是从 CAN 硬件的状态机的角度定义的，也就是位时间开始和结束的采样点。当同步一个“early”边沿时，该状态机忽略 Sync_Seg，因为它接下来不能将 Phase_Seg2 内发生边沿跳变处的时间片重新定义为 Sync_Seg。

图 5-85 的示例为如何通过同步过滤短显性噪声脉冲。这些例子中尖峰脉冲都在 Prop_Seg 尾端开始，长度都为 (Prop_Seg + Phase_Seg1)。

在第一个例子中，同步跳转宽度大于或等于从隐性到显性的尖峰脉冲边沿的相位误差。因此在尖峰脉冲末尾之后，采样点移动。一个隐性总线电平被采样。

在第二个例子中，SJW 小于相位误差，所以采样点还不足以被移动；显性尖峰脉冲将被作为当前总线电平被采样。

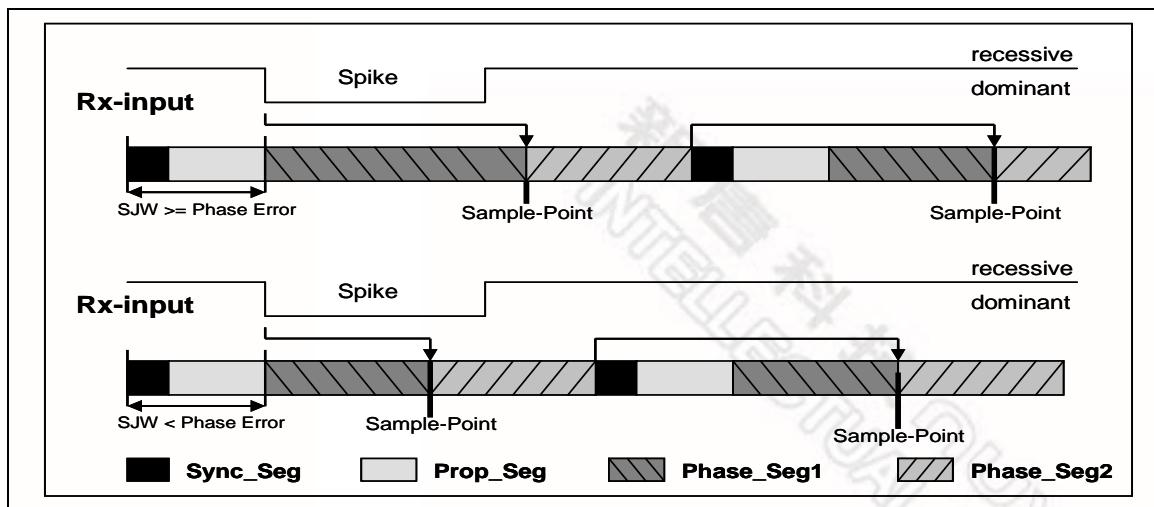


图 5-85 过滤短显性尖峰脉冲

5.13.6.10.4 振荡器容差范围

当 CAN 协议从版本 1.1 发展到版本 1.2，振荡器容差范围增加。（版本 1.0 未在芯片应用中实现过）从显性到隐性的边沿同步选项被丢弃，仅有从隐性到显性的边沿被考虑用于同步。协议更新到版本2.0 (A 和 B) 该部分无更新。

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

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

该值取决于 Phase_Seg1, Phase_Seg2, SJW 和 位时间。最大容错值 df 由两个条件决定（两个必须都满足）：

$$I : df \leq \frac{\text{Min}(\text{Phase_Seg1}, \text{Phase_Seg2})}{2 \times (13 \times \text{bit_time} - \text{Phase_Seg2})}$$

$$II : df \leq \frac{\text{SJW}}{20 \times \text{bit_time}}$$

注：这些条件基于 APB 时钟 = f_{osc} .

必须考虑 SJW 可能不大于相位缓存段较小的部分，传播时间段限制了可被用于相位缓存段位时间部分。

当 Prop_Seg = 1 和 Phase_Seg1 = Phase_Seg2 = SJW = 4 时，允许最大可能的振荡器容差为 1.58%。本组合中传输时间段仅占位时间 10 % 不适用于短位时间，可被用于在总线 40m 长度上位速率达到 125 kBit/s (位时间为 8 μ s) 的情况。



5.13.6.10.5 配置 CAN 协议控制器

在大多数 CAN 工具，也包括 C_CAN，位定时配置设定在 2 个寄存器字节。Prop_Seg 和 and Phase_Seg1 (as TSEG1) 的总和和 Phase_Seg2 (as TSEG2) 组合在一个字节中。SJW 和 BRP 组合在另一个字节中。

在这些位定时寄存器中，四个构成部分 TSEG1, TSEG2, SJW 和 BRP 必须编程设置为某个数值，该值比它们的实际功能值小1。因此，用 [0..n-1] 编程设定值来代替范围[1..n]的值。此方法中，例如 SJW (功能值范围为 [1..4]))仅用两位就可以表示了。

因此位时间的长度 (编程设定值) 为 $[TSEG1 + TSEG2 + 3] t_q$ 或 (功能值) 为 $[Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q$ 。

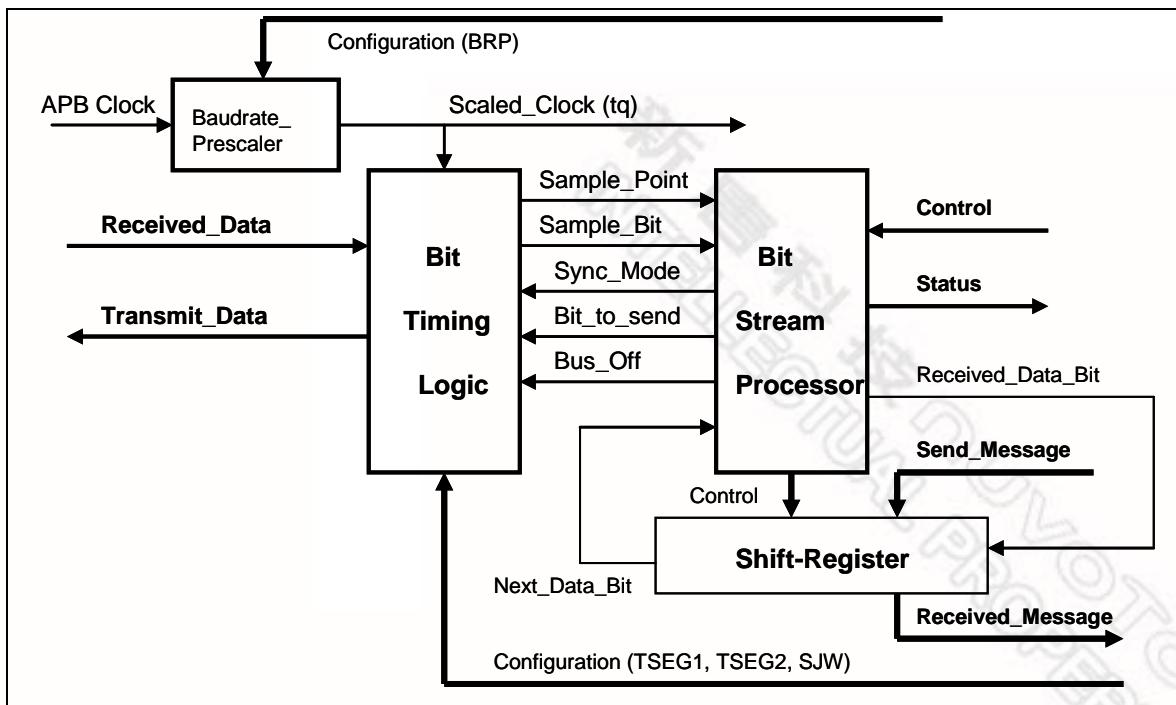


图 5-86 CAN 内核的协议控制器结构

位定时寄存器中的数据是 CAN 协议控制器的配置输入。波特率分频器（用 BRP 配置）定义了时间片的长度，位时间的基本时间单元；位定时逻辑（由 TSEG1, TSEG2 和 SJW 设置）定义了位时间中的时间片数目。

位时间的过程，采样点位置的计算和临时的同步由位定时逻辑 BTL (Bit Timing Logic) 状态机控制，该状态机每时间片就被计算一次。CAN 协议控制器的其他部分，位流处理器 BSP (Bit Stream Processor) 状态机每个位时间在采样点被计算一次。

移位寄存器以串行方式发送报文，以并行方式接收报文。该寄存器的载入和移位由 BSP 控制。

BSP 把报文变成帧，反之亦然。它产生和丢弃包装固定格式位，插入和提取填充位，计算和确认 CRC 码，执行错误管理，以及决定使用的同步类型。该寄存器在采样点被评估并处理采样总线输入位。计算在采样点之后发送的下一位（如 数据位，CRC 位，填充位，错误标记或空闲）所需的时间被称为信息处理时间 (IPT)。

IPT 和应用有关，但是不会长于 $2 t_q$ ；对 C_CAN 来说 IPT 为 $0 t_q$ 。它的长度是 Phase_Seg2 编程设定长度的下限。在同步的情况下，Phase_Seg2 可能被缩短到一个小于 IPT 的值，该值不会影响总线定时。

5.13.6.10.6 计算位定时参数

通常，位定时配置的计算以一个期望的位速率或位时间开始。位时间（1/位速率）的结果必须是 APB 时钟周期的整数倍。

位时间可能包括 4 到 25 个时间片，时间片的长度 t_q 由波特率分频器定义 $t_q = (\text{Baud Rate Prescaler})/f_{\text{apb_clk}}$ 。几个组合通过下面的步骤可达到期望的位时间。

首先，位时间中需要定义的部分是 Prop_Seg。它的长度取决于用 APB 时钟测量的延时时间。最大的总线长度和最大的节点延时一样必须定义用于可扩展的 CAN 总线系统。作为 Prop_Seg 的结果的时间被转化成时间片（向上取最接近的 t_q 的整数倍的值）。

Sync_Seg 是 1 t_q （固定）长度，留下 (位时间 – Prop_Seg – 1) t_q 给两个相位缓存段。如果剩下的 t_q 个数是偶数，相位缓存段有相同长度，Phase_Seg2 = Phase_Seg1，如果是奇数情况，则 Phase_Seg2 = Phase_Seg1 + 1。

Phase_Seg2 最小的标称值也必须要考虑一下。Phase_Seg2 不能比 CAN 控制器的 IPT 短，而 IPT 的范围是 [0..2] t_q ，取决于控制器的实际硬件。

同步跳转宽度的长度被设为它的最大值，即 4 和 Phase_Seg1 中的较小值。

作为结果的配置所需要的振荡器容差范围由章节 5.13.6.10.4：振荡器容差范围 中的方程式计算。

如果多于 1 个的配置是可能的，那么运行最高振荡器容差范围的配置必须是可选的。

带有多个不同系统时钟的 CAN 节点要求不同的配置来达到相同的位速率。CAN 网络中的传播时间计算是基于最长延时时间的节点，该计算对整个网络只做一次。

CAN 系统振荡器容差范围被有着最低容差范围的节点限制了。

根据计算结果，可能会要求减少总线长度或位速率，或者提高振荡器频率的稳定性，目的是寻找满足协议的 CAN 位定时配置。写入位定时寄存器的配置结果为：(Phase_Seg2-1) & (Phase_Seg1+Prop_Seg-1) & (SynchronisationJumpWidth-1)&(Prescaler-1)

高波特率的位定时示例：

该例中，APB_CLK 的时钟为 10 MHz，BRP 为 0，位速率为 1 MBit/s。

T_q	100	ns	$= t_{APB_CLK}$
delay of bus driver	50	ns	
delay of receiver circuit	30	ns	
delay of bus line (40m)	220	ns	
t_{Prop}	600	ns	$= 6 \cdot t_q$
t_{SJW}	100	ns	$= 1 \cdot t_q$
t_{TSeg1}	700	ns	$= t_{Prop} + t_{SJW}$
t_{TSeg2}	200	ns	$=$ 信息处理时间 (IPT) $+ 1 \cdot t_q$
$t_{Sync-Seg}$	100	ns	$= 1 \cdot t_q$
bit time	1000	ns	$= t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$
APB_CLK 容差	0.39	%	$= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ time - PB2)}$ $= \frac{0.1\mu s}{2 \times (13 \times (1\mu s - 0.2\mu s))}$

该例中，连接的位时间参数为 $(2-1)_3 \& (7-1)_4 \& (1-1)_2 \& (1-1)_6$ ，则位定时寄存器设定为 0x1600.

低波特率的位定时示例：

该例中，APB_CLK 的频率为 2 MHz，BRP 为 1，位速率为 100 KBit/s。

t_q	1	$\mu s = 2 \cdot t_{APB_CLK}$
delay of bus driver	200	ns
delay of receiver circuit	80	ns
delay of bus line (40m)	220	ns
t_{Prop}	1	$\mu s = 1 \cdot t_q$
t_{SJW}	4	$\mu s = 4 \cdot t_q$
t_{TSeg1}	5	$\mu s = t_{Prop} + t_{SJW}$
t_{TSeg2}	4	$\mu s = \text{信息处理时间} + 3 \cdot t_q$
$t_{Sync-Seg}$	1	$\mu s = 1 \cdot t_q$
bit time	10	$\mu s = t_{Sync-Seg} + t_{TSeg1} + t_{TSeg2}$
APB_CLK 的容差	1.58	$\% = \frac{\text{Min}(PB1, PB2)}{2 \times 13 \times (\text{bit time} - PB2)}$ $= \frac{4\mu s}{2 \times (13 \times (10\mu s - 4\mu s))}$

该例中，连接的位时间参数为 (4-1)3&(5-1)4&(4-1)2&(2-1)6，位定时寄存器设定为 0x34C1。

5.13.7 寄存器描述

C_CAN 分配了一个 256 字节的地址空间。这些寄存器按照 16-位 寄存器进行组织的。

两组接口寄存器其 (IF1 和 IF2) 控制着软件对报文 RAM 的访问。它们分配数据用于往返 RAM 的传输，避免软件访问和报文接收/发送之间的冲突。

5.13.8 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
CAN0_BA = 0x4018_0000				
CAN_CON	CAN0_BA+0x00	R/W	控制寄存器	0x0000_0001
CAN_STATUS	CAN0_BA+0x04	R/W	状态寄存器	0x0000_0000
CAN_ERR	CAN0_BA+0x08	R	错误计数	0x0000_0000
CAN_BTIME	CAN0_BA+0x0C	R/W	位定时寄存器	0x0000_2301
CAN_IIDR	CAN0_BA+0x10	R	中断标识符寄存器	0x0000_0000
CAN_TEST	CAN0_BA+0x14	R/W	测试寄存器	*(1)
CAN_BRPE	CAN0_BA+0x18	R/W	BRP 扩充寄存器	0x0000_0000
CAN_IF1_CREQ	CAN0_BA+0x20	R/W	IFn ⁽²⁾ 命令请求寄存器	0x0000_0001
CAN_IF2_CREQ	CAN0_BA+0x80	R/W		
CAN_IF1_CMASK	CAN0_BA+0x24	R/W	IFn 命令掩码寄存器	0x0000_0000
CAN_IF2_CMASK	CAN0_BA+0x84	R/W		
CAN_IF1_MASK1	CAN0_BA+0x28	R/W	IFn 掩码 1 寄存器	0x0000_FFFF
CAN_IF2_MASK1	CAN0_BA+0x88	R/W		
CAN_IF1_MASK2	CAN0_BA+0x2C	R/W	IFn 掩码 2 寄存器	0x0000_FFFF
CAN_IF2_MASK2	CAN0_BA+0x8C	R/W		
CAN_IF1_ARB1	CAN0_BA+0x30	R/W	IFn 仲裁 1 寄存器	0x0000_0000
CAN_IF2_ARB1	CAN0_BA+0x90	R/W		
CAN_IF1_ARB2	CAN0_BA+0x34	R/W	IFn 仲裁 2 寄存器	0x0000_0000
CAN_IF2_ARB2	CAN0_BA+0x94	R/W		
CAN_IF1_MCON	CAN0_BA+0x38	R/W	IFn 报文控制寄存器	0x0000_0000
CAN_IF2_MCON	CAN0_BA+0x98	R/W		
CAN_IF1_DAT_An/	CAN0_BA+0x3C~40	R/W	IFn 数据 An ⁽³⁾ and 数据 Bn ⁽³⁾ 寄存器 eg: CAN_IF1_DAT_A1 = CAN_BA+0x3Ch CAN_IF1_DAT_A2 = CAN_BA+0x40h	0x0000_0000
CAN_IF1_DAT_Bn/	CAN0_BA+0x44~48	R/W		
CAN_IF2_DAT_An/	CAN0_BA+0x9C~A0	R/W		
CAN_IF2_DAT_Bn/	CAN0_BA+0xA4~A8	R/W		

CAN_TXREQ1	CAN0_BA+0x100	R	传送请求寄存器 1 & 2	0x0000_0000
CAN_NDAT1	CAN0_BA+0x120	R	新数据寄存器 1 & 2	0x0000_0000
CAN_NDAT2	CAN0_BA+0x124	R		
CAN_IPND1	CAN0_BA+0x140	R	中断挂起寄存器 1 & 2	0x0000_0000
CAN_IPND2	CAN0_BA+0x144	R		
CAN_MVLD1	CAN0_BA+0x160	R	报文有效寄存器 1 & 2	0x0000_0000
CAN_MVLD2	CAN0_BA+0x164	R		
CAN_WU_EN	CAN0_BA+0x168	R/W	唤醒功能使能	0x0000_0000
CAN_WU_STATUS	CAN0_BA+0x16C	R/W	唤醒功能状态	0x0000_0000

注： 1. 0x00 & 0br0000000, 其中 r 表示 CAN_RX 的当前值

2. IFn: 两组报文接口寄存器 – IF1 and IF2, 有相同的功能
3. An/Bn: 两组数据寄存器 – A1, A2 and B1, B2.

5.13.9 CAN 接口复位状态

在硬件复位后, C_CAN 寄存器保持在 [CAN 寄存器映射](#) 寄存器描述的复位值。

此外, busoff 状态复位, CAN_TX 的输出被设置为隐性 (HIGH)。CAN 控制寄存器的值 0x0001 (Init = '1') 使能软件初始化。C_CAN 直到应用软件设置 Init 位为'0'才会影响 CAN 总线。

保存在报文 RAM 中的数据不受硬件复位影响。在上电后, 报文 RAM 中的内容是未定义的。

CAN 寄存器每位的位功能映射

偏移量	寄存器名	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
00h	CAN_CON									Test	CCE	DAR	Res	EIE	IE	Init	
04h	CAN_STATUS									Boff	EWarn	EPass	RxOk	TxOk		LEC	
08h	CAN_ERR	RP														TEC7-0	
0Ch	CAN_BTIME	Res		TSeg2		TSeg1			SJW							BRP	
10h	CAN_IIDR					IntId15-8										IntId7-0	
14h	CAN_TEST					Reserved			Rx	Tx1	Tx0	LBack	Silent	Basic		Reserved	
18h	CAN_BRPE					Reserved										BRPE	
20h	CAN_IF1_CREQ	Busy				Reserved										Message Number	
24h	CAN_IF1_CMASK					Reserved			W/RD	Mask	Arb	Control	CirIntPnd	TxRqst/	Data A	Data B	
28h	CAN_IF1_MASK1								Msk15-0								
2Ch	CAN_IF1_MASK2	MXtd	MDir	Res												Msk28-16	
30h	CAN_IF1_ARB1								ID15-0								
34h	CAN_IF1_ARB2	MsgVal	Xtd	Dir												ID28-16	

Addr offset	Register Name	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0	
38h	CAN_IF1_MCON	NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst	EoB	Reserved			DLC3-0				
3Ch	CAN_IF1_DAT_A1	Data(1)							Data(0)									
40h	CAN_IF1_DAT_A2	Data(3)							Data(2)									
44h	CAN_IF1_DAT_B1	Data(5)							Data(4)									
48h	CAN_IF1_DAT_B2	Data(7)							Data(6)									
80h	CAN_IF2_CREQ	Busy	Reserved							Message Number								
84h	CAN_IF2_CMASK	Reserved							W/R/RD	Mask	Arb	Control	ClrIntPnd	TxRqst	Data A	Data B		
88h	CAN_IF2_MASK1	Msk15-0																
8Ch	CAN_IF2_MASK2	MXtd	MDir	Res.	Msk28-16													
90h	CAN_IF2_ARB1	ID15-0																
94h	CAN_IF2_ARB2	MsgVal	Xtd	Dir	ID28-16													
98h	CAN_IF2_MCON	NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst	EoB	Reserved			DLC3-0				
9Ch	CAN_IF2_DAT_A1	Data(1)							Data(0)									

Addr offset	Register Name	1 5	1 4	1 3	1 2	1 1	1 0	9	8	7	6	5	4	3	2	1	0
A0h	CAN_IF2_DAT_A2	Data(3)										Data(2)					
A4h	CAN_IF2_DAT_B1	Data(5)										Data(4)					
A8h	CAN_IF2_DAT_B2	Data(7)										Data(6)					
100h	CAN_TXREQ1	TxRqst16-1															
104h	CAN_TXREQ2	TxRqst32-17															
120h	CAN_NDAT1	NewDat16-1															
124h	CAN_NDAT2	NewDat32-17															
140h	CAN_IPND1	IntPnd16-1															
144h	CAN_IPND2	IntPnd32-17															
160h	CAN_MVLD1	MsgVal16-1															
164h	CAN_MVLD2	MsgVal32-17															
168h	CAN_WU_EN	Reserved															
16Ch	CAN_WU_STATUS	Reserved															
170h	CAN_RAM_CEN	Reserved															
Others	Reserved	Reserved															

表 5-16 CAN 寄存器每位的位功能映射

注：保留位除了掩码 2 寄存器读取时为‘1’，其余读取为 ‘0’。

Res. = Reserved (保留)

CAN 控制寄存器 (CAN_CON)

寄存器	偏移量	R/W	描述	复位后的值
CAN_CON	CAN0_BA+0x00	R/W	CAN 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
Test	CCE	DAR	Reserved	EIE	SIE	IE	Init

Bits	描述	
[31:8]	Reserved	保留 这些位保留位。 这些位读时总是为 '0'，也必须总是写 '0' 到该些位。
[7]	Test	测试模式使能 1 = 测试模式 0 = 正常操作
[6]	CCE	配置改变使能 1 = 允许写访问到位定时寄存器 (CAN_BTIME & CAN_BRP)。 (当 Init 位 =1)。 0 = 无写访问到位定时寄存器
[5]	DAR	禁用自动重传 1 = 自动重传禁用 0 = 被干扰的报文自动重传使能
[4]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[3]	EIE	错误中断使能 1 = 使能 - 状态寄存器的 BOff 或 EWarn 位的改变将产生中断 0 = 禁用 - 无错误状态中断产生
[2]	SIE	状态改变中断使能 1 = 使能 - 当一条报文传输成功完成或一个 CAN 总线错误被检测到，中断将产生。 0 = 禁用 - 不会产生状态改变中断。



[1]	IE	模块中断使能 1 = 使能 0 = 禁用
[0]	Init	Init 初始化 1 = 初始化开始 0 = 正常操作

注: busoff 恢复次序 (见 CAN 规范 Rev. 2.0) 不能通过设置或复位 Init 位来缩短。如果设备进入 busoff 状态, 它将设置自置位 Init 位, 停止所有总线的活动。一旦 Init 被 CPU 清除了, 设备然后将在继续正常操作之前等待 129 个总线空闲 (129 * 11 连续的隐性位)。在 busoff 恢复序列的最后, 错误管理计数器将复位。

在复位 Init 之后的等待时间过程中, 每次监测到 11 个连续的隐性位, 一个 Bit0Error 码被写入状态寄存器, 使能 CPU 立即去检查 CAN 总线是否被困在显性或连续的干扰状态, 从而监控 busoff 恢复序列的进程。

CAN 状态寄存器 (CAN_STATUS)

寄存器	偏移量	R/W	描述	复位后的值
CAN_STATUS	CAN0_BA+0x04	R/W	CAN 状态寄存器	0x0000_0000

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

Bits	描述	
[31:8]	Reserved	保留 这些位为保留位。读时总是为 '0'，也必须总是写 '0' 到这些位。
[7]	BOff	Busoff 状态 (只读) 1 = The CAN 模块在 busoff 态 0 = The CAN 模块不在 busoff 态
[6]	EWarn	错误警告状态 (只读) 1 = 至少 EML 中的一个错误计数器达到了错误警告 96 的限制。 0 = 所有错误计数器都处于 错误警告 96 的限制以下。
[5]	EPass	被动错误 (只读) 1 = CAN 内核在 CAN 规范定义的被动错误态。 0 = CAN 内核处于主动错误。
[4]	RxOK	成功接收一条报文 1 = 自上次该位被 CPU 复位后，一条报文被成功收到。（独立于接收过滤的结果） 0 = 自上次该位被 CPU 复位后，还没有报文被成功收到。 该位从不被 CAN 内核复位。
[3]	TxOK	成功发送一条报文 1 = 自上次该位被 CPU 复位后，一条报文被成功（至少无错误或被至少一个其他节点响应）发送。 0 = 自上次该位被 CPU 复位后，还没有报文被成功发送。 该位从不被 CAN 内核复位。
[2:0]	LEC	上一次的错误码 (出现在 CAN 总线上的上次的错误类型)

	LEC 域保持一个码，该码指示上一个在 CAN 总线上发生的错误。当一条报文被无错误传输（接收或发送）后，该域将被清为 '0'。未使用的码 '7' 可以被 CPU 写入来确认更新。表 5-17 为错误码的描述。
--	---

错误码	意义
0	无错误
1	填充错误：连续超过 5 个相同的位出现在接收报文的一部分中，这是不允许的
2	格式错误：收到的帧中固定格式的部分出现了错误格式
3	应答错误：CAN 内核传送到报文没有被另一个节点应答。
4	Bit1 错误：在发送一条报文的过程中（除仲裁域之外），设备想发送一个隐性位（逻辑值为 '1' 的位），但是监测到总线的值为显性。
5	Bit0 错误：在发送一条报文（或应答位，或主动错误标志，或超载标志）的过程中，虽然设备想发送一个显性电平（数据或标识符逻辑值为 '0'），但是监测到的总线值为隐性。在 busoff 恢复的过程中，每检测到一个连续的 11 位隐性位，该状态设置一次。这使能 CPU 监控 busoff 恢复序列的进程。（指示总线没有被显性或连续的干扰困住）
6	CRC 错误：收到的报文的 CRC 检查不正确，发过来的报文中接收的 CRC 和 计算收到数据得到的 CRC 不一致。
7	未使用：当 LEC 显示为值 '7'，表示自上次 CPU 写该值到 LEC，没有 CAN 总线事件被检测到。

表 5-17 错误码

状态中断

状态中断由 **BOff** 和 **EWarn**（错误中断）位或 **RxOk**, **TxOk** 和 **LEC**（状态改变中断）位产生（假设 CAN 寄存器中相应的使能位被置位了）。**EPass** 位或对 **RxOk**, **TxOk** 或 **LEC** 的写入不会产生状态中断。

如果该中断挂起，读状态寄存器将清除中断寄存器中的状态中断值 (8000h)，

CAN 错误计数寄存器 (CAN_ERR)

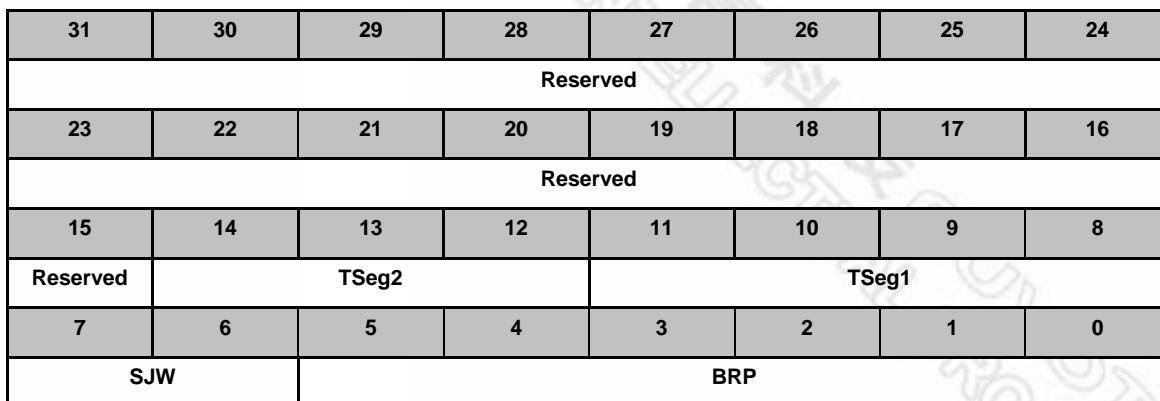
寄存器	偏移量	R/W	描述	复位后的值
CAN_ERR	CAN0_BA+0x08	R	错误计数寄存器	0x0000_0000

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

Bits	描述	
[31:16]	Reserved	保留 这些位为保留位。读时总是为 '0'，也必须总是写 '0' 到这些位。
[15]	RP	接收错误认可 1 = 接收错误计数器已经达到 CAN 规范定义的错误认可水平 0 = 接收错误计数器在错误认可水平之下
[14:8]	REC	接收错误计数 接收错误计数器的当前状态。值在 0 到 127 之间。
[7:0]	TEC	发送错误计数 发送错误计数器的当前状态。值在 0 到 255 之间。

位定时寄存器 (CAN_BTIME)

寄存器	偏移量	R/W	描述	复位后的值
CAN_BTIME	CAN0_BA+0x0C	R/W	位定时寄存器	0x0000_2301

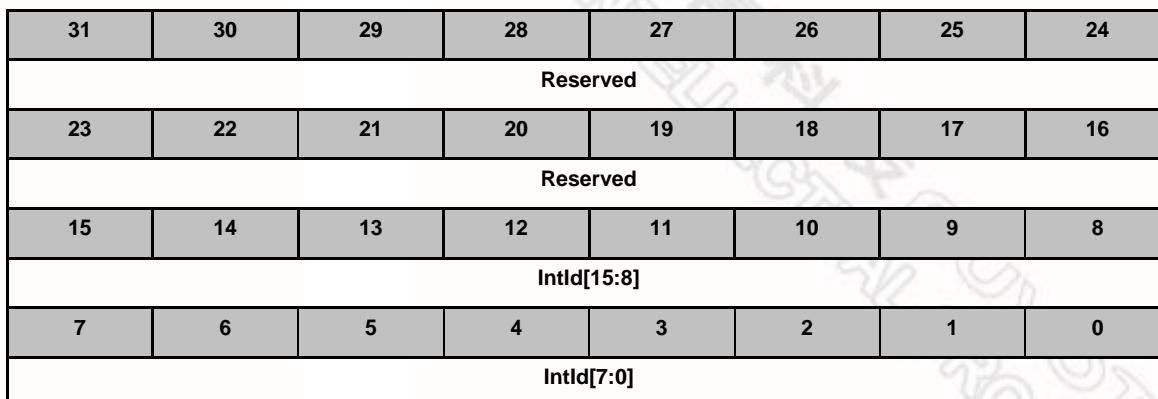


Bits	描述	
[31:15]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[14:12]	TSeg2	采样点后的时间段 0x0-0x7: TSeg2 的有效值为 [0 ... 7]. 该位硬件实际解释的值比该处编程使用的值多 1.
[11:8]	TSeg1	采样点前 - Sync_seg 的时间段 0x01-0x0F: TSeg1 的有效值为 [1 ... 15]. 该位硬件实际解释的值比该处编程使用的值多 1.
[7:6]	SJW	(重新) 同步跳转宽度 0x0-0x3: 有效值为 [0 ... 3]. 该位硬件实际解释的值比该处编程使用的值多 1.
[5:0]	BRP	波特率预分频器 0x01-0x3F: 该值用于振荡器频率除频产生位时间片。位时间是该时间片的倍数累积。波特率分频器的有效值为 [0 ... 63]. 该位硬件实际解释的值比该处编程使用的值多 1.

注：模块时钟 APB_CLK 为 8 MHz，其他的值为 0x2301 时，配置 C_CAN 的位速率为 500 kBit/s。这些寄存器只在 CCE 位和 CAN 控制寄存器中 Init 位被置位后才可写。

中断标识符寄存器 (CAN_IIDR)

寄存器	偏移量	R/W	描述	复位后的值
CAN_IIDR	CAN0_BA+0x10	R	中断标识符寄存器	0x0000_0000



Bits	描述
[15:0]	<p>IntId</p> <p>中断标识符（指示中断源，参考表 5-18）</p> <p>如果一些中断挂起，CAN 中断寄存器将忽略其排序而指向拥有最高优先级的挂起中断。中断将保持挂起直到应用软件清除它。如果 IntId 不为 0x0000 且 IE 置位，则到 IEC 的 IRQ 中断信号被激活。中断保持激活直到 IntId 回复为 0x0000（该中断的原因已复位）或直到 IE 复位。</p> <p>状态中断拥有最高优先级。在报文中断中，报文对象的优先级随着报文号的增加而降低。</p> <p>一个报文中断通过清除该报文对象的 IntPnd 位清除。状态中断通过状态寄存器清除。</p>

IntId 值	意义
0x0000	没有中断挂起
0x0001-0x0020	造成中断的报文对象号
0x0021-0x7FFF	未使用
0x8000	状态中断
0x8001-0xFFFF	未使用

表 5-18 中断源

测试寄存器 (CAN_TEST)

寄存器	偏移量	R/W	描述	复位后的值
CAN_TEST	CAN0_BA+0x14	R/W	测试寄存器	0x0000_00x0

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

Bits	描述	
[31:8]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[7]	Rx	监测 CAN_RX 管脚的当前实际值 (只读) 1 = CAN 总线为隐性 (CAN_RX = '1'). 0 = CAN 总线为显性 (CAN_RX = '0').
[6:5]	Tx[1:0]	Tx[1:0]: CAN_TX 管脚的控制 00 = 复位值, CAN_TX 由 CAN 内核控制 01 = 采样点可以在 CAN_TX 管脚监测 10 = CAN_TX 管脚驱动一个显性 ('0') 值 11 = CAN_TX 管脚驱动一个显性 ('1') 值
[4]	LBack	Loop Back 模式 1 = Loop Back 模式使能 0 = Loop Back 模式禁用
[3]	Silent	Silent 模式 1 = 处于 Silent 模式 0 = 正常操作
[2]	Basic	Basic 模式 1= IF1 寄存器用作 Tx Buffer, IF2 寄存器用作 Rx Buffer。 0 = Basic 模式禁用
[1:0]	Res	保留

		该位为保留位。 该位读时总是为‘0’，也必须总是写‘0’到该位。
--	--	-------------------------------------

复位值: 0000 0000 R000 0000 b (R: RX 管脚的当前值)

通过设置 CAN 控制寄存器的 Test 位来使能测试寄存器的写访问。不同的测试功能可以结合，但是 Tx[1-0] ≠ “00” 干扰报文传输。

波特率分频扩展寄存器 (CAN_BRPE)

寄存器	偏移量	R/W	描述	复位后的值
CAN_BRPE	CAN0_BA+0x18	R/W	波特率分频扩展寄存器	0x0000_0000

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

Bits	描述	
[31:4]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位
[3:0]	BRPE	BRPE: 波特率分频扩展 0x00-0x0F: 通过编程 BRPE , 波特率分频可以被扩展到 1023。该位硬件实际解释的值比编程使用的值 (BRPE (MSBs) 和 BTIME (LSBs)) 多1。

报文接口寄存器组

有 2 组接口寄存器，用来控制 CPU 访问报文 RAM。接口寄存器避免 CPU 访问报文 RAM 和 CAN 报文接收和发送（通过缓存传输）之间的冲突。一个完整的报文对象或报文对象的某些部分可以在报文 RAM 和 IFn 报文缓存寄存器之间一次单独的传输中传输。

除了 Basic 测试模式外，2组接口寄存器的功能是相同的。它们可以这样用：一组寄存器用于到报文 RAM 的数据传输，另一组用于来自报文 RAM 的数据传输，而且允许彼此间的中断。表 5-19 (IF1 和 IF2 报文接口寄存器组) 提供了2组接口寄存器的概述。

每组接口寄存器由它们自身的命令寄存器控制的报文缓存寄存器构成。命令掩码寄存器制定了数据传输的方向和报文对象将要被传输的部分。命令请求寄存器用于选择报文 RAM 中的一个报文对象作为传输的目标或源，并开始命令掩码寄存器中指定的行为。

地址	IF1 寄存器组	地址	IF2 寄存器组
CAN0_BA+0x20	IF1 命令请求	CAN0_BA+0x80	IF2 命令请求
CAN0_BA+0x24	IF1 命令掩码	CAN0_BA+0x84	IF2 命令掩码
CAN0_BA+0x28	IF1 掩码 1	CAN0_BA+0x88	IF2 掩码 1
CAN0_BA+0x2C	IF1 掩码 2	CAN0_BA+0x8C	IF2 掩码 2
CAN0_BA+0x30	IF1 仲裁 1	CAN0_BA+0x90	IF2 仲裁 1
CAN0_BA+0x34	IF1 仲裁 2	CAN0_BA+0x94	IF2 仲裁 2
CAN0_BA+0x38	IF1 报文控制	CAN0_BA+0x98	IF2 报文控制
CAN0_BA+0x3C	IF1 数据 A 1	CAN0_BA+0x9C	IF2 数据 A 1
CAN0_BA+0x40	IF1 数据 A 2	CAN0_BA+0xA0	IF2 数据 A 2
CAN0_BA+0x44	IF1 数据 B 1	CAN0_BA+0xA4	IF2 数据 B 1
CAN0_BA+0x48	IF1 数据 B 2	CAN0_BA+0xA8	IF2 数据 B 2

表 5-19 IF1 和 IF2 报文接口寄存器

IFn 命令请求寄存器 (CAN_IFn_CREQ)

寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_CREQ	CAN0_BA+0x20/0x80	R/W	IFn 命令请求寄存器	0x0000_0001

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

Bits	描述
[15]	Busy 忙标志 1 = 写 Ifn 命令请求寄存器的动作正在进行。该位只能由软件读取 0 = 读/写动作已完成
[14:6]	Reserved 保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[5:0]	Message Number 报文号 0x01-0x20: 有效的报文号，报文 RAM 中被选择用于数据传输的报文对象 0x00: 非有效的报文号，视同 0x20 0x21-0x3F: 非有效的报文号，视同 0x01-0x1F.

一旦应用软件写报文号到命令请求寄存器报文传输就开始了。伴随着这个写操作，Busy 位将自动被置位来通知 CPU 传输正在进行中。在等待 3 到 6 个 APB_CLK 周期后，接口寄存器和报文 RAM 之间的传输完成了。Busy 位被清除。

注：当写入到命令请求寄存器中的报文号非有效时，报文号将被解释转换为一个有效的值，报文对象也将被传输。

IFn 命令掩码寄存器 (CAN_IFn_CMASK)

IFn 命令掩码寄存器指定传输方向和选择哪个 IFn 报文缓存寄存器作为数据传输的源或目标。

寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_CMASK	CAN0_BA+0x24/0x84	R/W	IFn 命令掩码寄存器	0x0000_0000

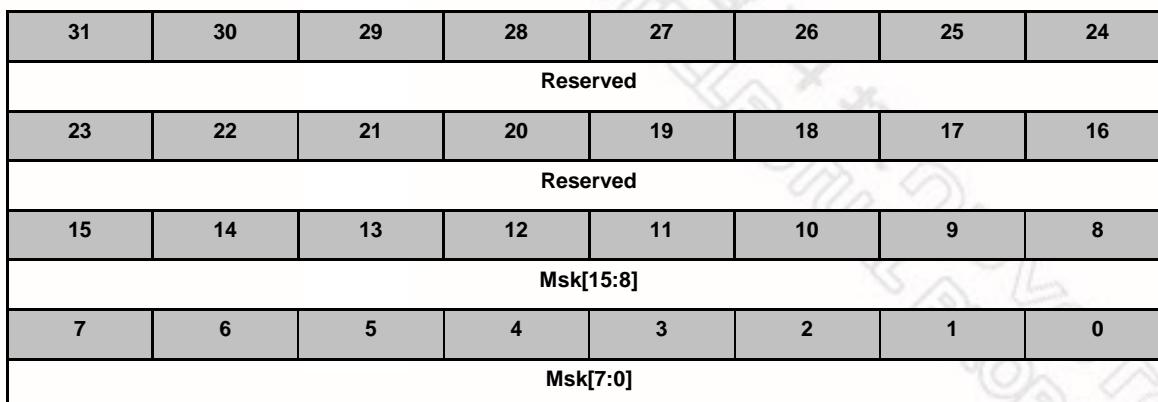
31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
WR/RD	Mask	Arb	Control	CrlntPnd	TxRqst/ NewDat	Data A	Data B

Bits	描述
[31:8]	Reserved 保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[7]	WR/RD 写/读 1 = 写：通过命令请求寄存器从选择的报文缓存寄存器传输数据到已编址的报文对象 0 = 读：通过命令请求寄存器从已编址的报文对象传输数据到选择的报文缓存寄存器
[6]	Mask 访问掩码位 <u>Direction = 写</u> 1 = 传输 Identifier Mask + MDir + MXtd 到报文对象 0: = 掩码位未改变 <u>Direction = 读</u> 1 = 传输 Identifier Mask + MDir + MXtd 到 IFn 报文缓存寄存器 0 = 掩码位未改变
[5]	Arb 访问仲裁位 <u>Direction = 写</u> 1 = 传输 Identifier + Dir + Xtd + MsgVal 到报文对象 0 = 仲裁位未改变 <u>Direction = 读</u> 1 = 传输 Identifier + Dir + Xtd + MsgVal 到 IFn 报文缓存寄存器 0 = 仲裁位未改变

[4]	Control	<p>控制访问控制位</p> <p><u>Direction = 写</u></p> <p>1 = 传输控制位到报文对象 0 = 控制位未改变</p> <p><u>Direction = Read</u></p> <p>1 = 传输控制位到 IFn 报文缓存寄存器 0 = 控制位未改变</p>
[3]	ClrIntPnd	<p>清除中断挂起位</p> <p><u>Direction = 写</u></p> <p>当向报文对象写时，该位忽略。</p> <p><u>Direction = 读</u></p> <p>1 = 清除报文对象中的 IntPnd 位 0 = IntPnd 位保持不变</p>
[2]	TxRqst/NewDat	<p>访问传送请求位 当 <u>Direction = 写</u></p> <p>1 = 设置 TxRqst 位 0 = TxRqst 位未改变</p> <p>注：如果一个传送是通过编程设定 IFn 命令掩码寄存器中的 TxRqst/NewDat 位而被请求的，则 IFn 报文控制寄存器中的 TxRqst 位将被忽略</p> <p>访问新数据位 当 <u>Direction = 读</u></p> <p>1 = 清除报文对象中的 NewDat 位 0 = NewDat 位保持不变</p> <p>注：对一个报文对象的读访问能够和控制位 IntPnd 和 NewDat 的复位结合起来。这些传输到 IFn 报文控制寄存器的值（在复位这些位之前）总是反映其状态。</p>
[1]	Data A	<p>访问数据字节 [3:0]</p> <p><u>Direction = 写</u></p> <p>1 = 传输数据字节[3:0] 到报文对象 0 = 数据字节 [3:0] 未改变</p> <p><u>Direction = 读</u></p> <p>1 = 传输数据字节 [3:0] 到 IFn 报文缓存寄存器 0 = 数据字节 [3:0] 未改变</p>
[0]	Data B	<p>访问数据字节 [7:4]</p> <p><u>Direction = 写</u></p> <p>1 = 传输数据字节 [7:4] 到报文对象 0 = 数据字节 [7:4] 未改变</p> <p><u>Direction = 读</u></p> <p>1 = 传输数据字节 [7:4] 到 IFn 报文缓存寄存器 0 = 数据字节 [7:4] 未改变</p>

IFn 掩码 1 Register (CAN_IFn_MASK1)

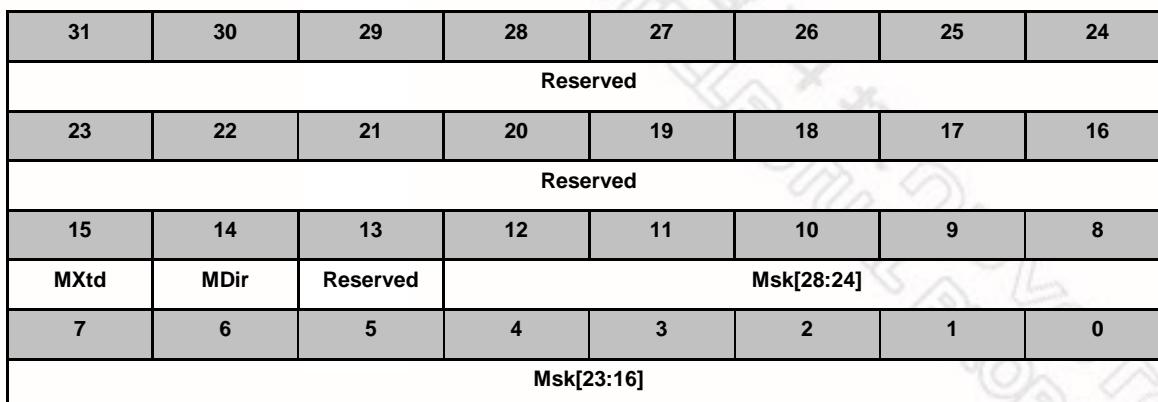
寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_MASK1	CAN0_BA+0x28/0x88	R/W	IFn 掩码 1 寄存器	0x0000_FFFF



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为‘0’，也必须总是写‘0’到该位。
[15:0]	Msk[15:0]	标识符掩码 15-0 1 = 相应的标识符位用于接收过滤 0 = 报文对象标识符的相应位无法禁止接收过滤中的匹配操作

IFn 掩码 2 寄存器 (CAN_IFn_MASK2)

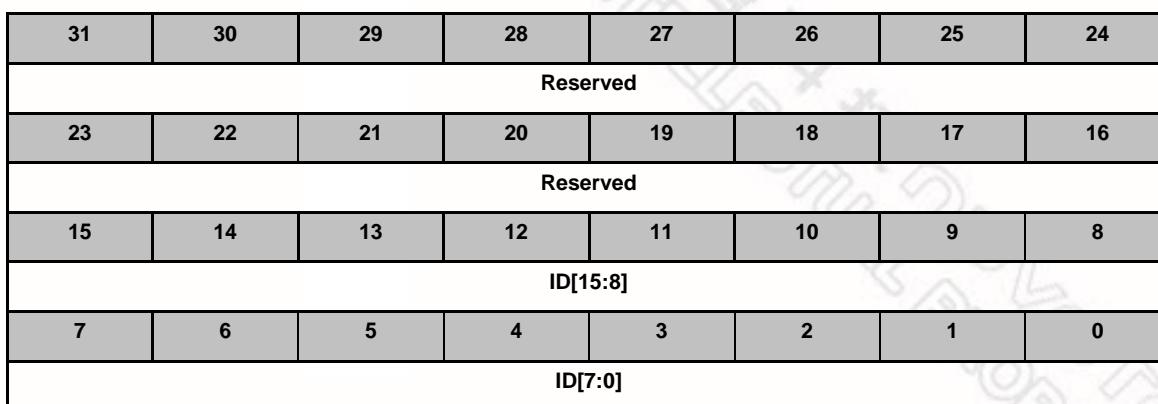
寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_MASK2	CAN0_BA+0x2C/0x8C	R/W	IFn 掩码 2 寄存器	0x0000_FFFF



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为‘0’，也必须总是写‘0’到该位。
[15]	MXtd	掩码扩展标识符 1 = 扩展标识符位 (IDE) 用于接收过滤。 0 = 扩展标识符位 (IDE) 对于接收过滤没有影响。 注： 当 11 位 (“standard”) 标识符用于报文对象时，接收到的数据帧的标识符被写入到位 ID28 到 ID18。对于接收过滤，只有这些位和掩码位 Msk28 到 Msk18 一起时才会被考虑。
[14]	MDir	掩码报文方向 1 = 报文方向位 (Dir) 用于接收过滤。 0 = 报文方向位 (Dir) 对于接收过滤没有影响。
[13]	Reserved	保留 该位为保留位。该位读时总是为‘1’。
[12:0]	Msk[28:16]	标识符掩码 28-16 1 = 相应的标识符位用于接收过滤 0 = 报文对象标识符的相应位无法禁止接收过滤中的匹配操作

IFn 仲裁 1 寄存器 (CAN_IFn_ARB1)

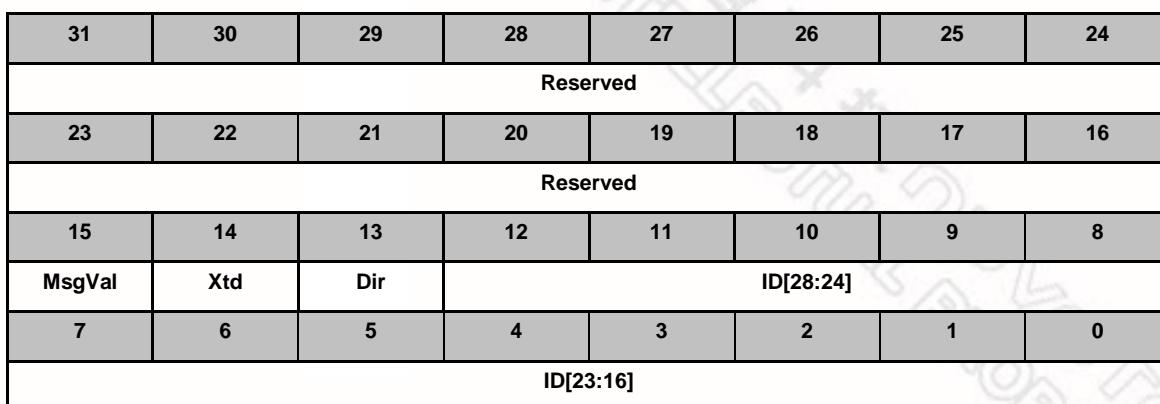
寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_ARB1	CANO_BA+0x30/0x90	R/W	IFn 仲裁 1 寄存器	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为‘0’，也必须总是写‘0’到该位。
[15:0]	ID[15:0]	报文标识符 15-0 ID28 - ID0, 29-位标识符 (“扩展帧”). ID28 - ID18, 11-位标识符 (“标准帧”)

IFn 仲裁 2 寄存器 (CAN_IFn_ARB2)

寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_ARB2	CANO_BA+0x34/0x94	R/W	IFn 仲裁 2 寄存器	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为‘0’，也必须总是写‘0’到该位。
[15]	MsgVal	报文合法 1 = 报文对象被配置，而且必须由报文处理器识别。 0 = 报文对象被报文处理器忽略。 注： 应用软件必须在初始化过程中，复位 CAN 控制寄存器的 Init 位之前，复位所有未使用的报文对象的 MsgVal 位。该位也必须在 Id28-0 , 控制位 Xtd , Dir , 或数据长度码 DLC3-0 被修改之前复位，或者如果报文对象不再必需时。
[14]	Xtd	扩展标识符 1 = 29-位（“扩展”）标识符将被用于报文对象。 0 = 11-位（“标准”）标识符将被用于报文对象。
[13]	Dir	报文方向 1 = 方向为发送 TxRqst 端，各自的报文对象作为数据帧被传输。在收到带有匹配标识符的远程帧时，该报文对象的 TxRqst 位被置位。（如果 RmtEn = 1）。 0 = 方向为接收 TxRqst 端，带有该报文对象标识符的远程帧被传输。在收到带有匹配标识符的数据帧时，报文保存在该报文对象内。
[12:0]	ID[28:16]	报文标识符 28-16 ID28 - ID0, 29-位标识符（“扩展帧”）。 ID28 - ID18, 11-位标识符（“标准帧”）。

IFn 报文控制寄存器 (CAN_IFn_MCON)

寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_MCON	CAN0_BA+0x38/0x98	R/W	IFn 报文控制寄存器	0x0000_0000

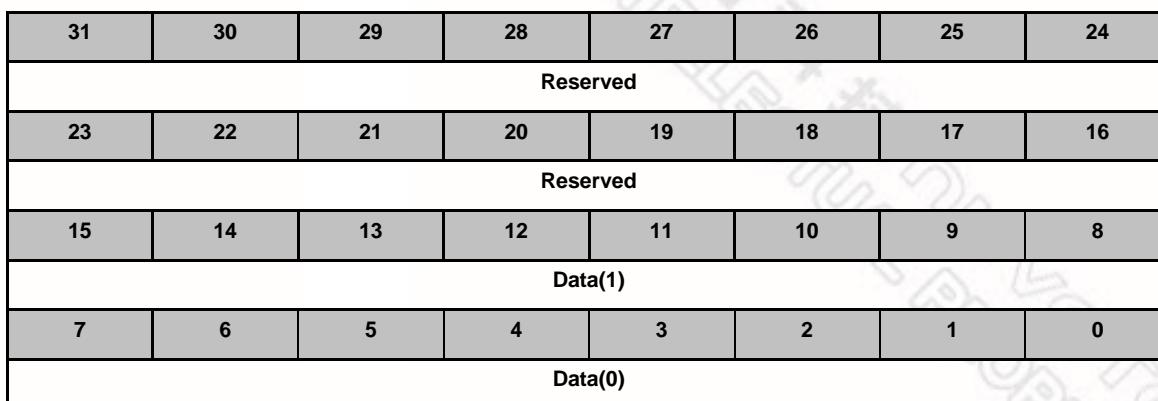
31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst
7	6	5	4	3	2	1	0
EoB	Reserved			DLC[3:0]			

Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15]	NewDat	新数据 1 = 报文处理器或应用软件已经写了新数据到该报文对象的数据部分。 0 = 自上次该位被应用软件清除后，该报文对象的数据部分还没有由报文处理器写入的新数据。
[14]	MsgLst	报文丢失（只对 Message Objects with direction = receive 有效） 1 = 当 NewDat 仍被置位时，报文处理器保存一个新的报文到该对象，CPU 丢失报文。 0 = 自上次该位被 CPU 复位后无报文丢失。
[13]	IntPnd	中断挂起 1 = 该报文对象是中断源。如果没有更高优先级的中断源存在时，中断寄存器中的中断标识符将指向该报文对象。 0 = 该报文对象不是中断源。
[12]	UMask	使用验收掩码 1 = 使用掩码 (Msk28-0, MXtd, and MDir) 用于接收过滤。 0 = 忽略掩码。 注：如果 UMask 位被置为 1，则在报文对象初始化的过程中，MsgVal 位被置为 1 之前，报文对象的掩码位必须被编程设定。
[11]	TxE	传送中断使能 1 = IntPnd 位将在一帧被成功传送后被置位。 0 = IntPnd 位在一帧被成功传送后将保持不变。

[10]	RxE	接收中断使能 1 = IntPnd 将在成功接收一帧后被置位。 0 = IntPnd 在成功接收一帧后将保持不变。
[9]	RmtEn	远程使能 1 = 在收到一远程帧后, TxRqst 被置位。 0 = 在收到一远程帧后, TxRqst 保持不变。
[8]	TxRqst	传送请求 1 = 该报文对象的传送被请求而且还没有完成。 0 = 该报文对象不在等待传送。
[7]	EoB	缓存结束 1 = 单报文对象或 FIFO 缓存的最后一个报文对象 0 = 报文对象属于某 FIFO 缓存, 而且不是该 FIFO 缓存的最后一个报文对象。 注: 该位用于连接 2 个或更多报文对象 (最多到 32) 来组建一个 FIFO 缓存。对于单报文对象 (不属于任一 FIFO 缓存), 该位必须总是置为 1。
[6:4]	Reserved	保留 该位为保留位。该位读时总是为 '0', 也必须总是写 '0' 到该位。
[3:0]	DLC	数据长度码 0-8: 数据帧有 0-8 数据字节。 9-15: 数据帧有 8 数据字节。 注: 报文对象的数据长度码必须被定义和其他节点中所有带有相同标识符的相对对象相同。当报文处理器保持一个数据帧, 它将写 DLC 为接收到的报文给的值。 Data 0: CAN 数据帧的 1st 数据字节 Data 1: CAN 数据帧的 2nd 数据字节 Data 2: CAN 数据帧的 3rd 数据字节 Data 3: CAN 数据帧的 4th 数据字节 Data 4: CAN 数据帧的 5th 数据字节 Data 5: CAN 数据帧的 6th 数据字节 Data 6: CAN 数据帧的 7th 数据字节 Data 7 : CAN 数据帧的 8th 数据字节 注: Data 0 字节是接收过程中, 移入 CAN 内核的移位寄存器的第一个数据字节, Data 7 字节是最后一个。当报文处理器保存一个数据帧, 它将会写所有 8 数据字节到报文对象中。如果数据长度码小于 8, 则报文对象剩下的字节将被非指定值覆盖。

IFn 数据 A1 寄存器 (CAN_IFn_DAT_A1)

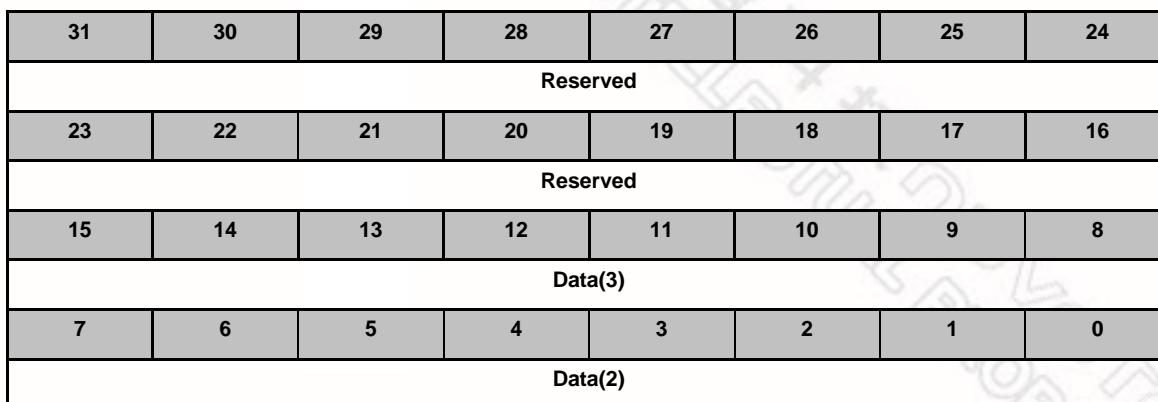
寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_DAT_A1	CANO_BA+0x3C/0x9C	R/W	IFn 数据 A1 寄存器	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:8]	Data (1)	数据字节 1 CAN 数据帧的第 2 个数据字节
[7:0]	Data (0)	数据字节 0 CAN 数据帧的第 1 个数据字节

IFn 数据 A2 寄存器 (CAN_IFn_DAT_A2)

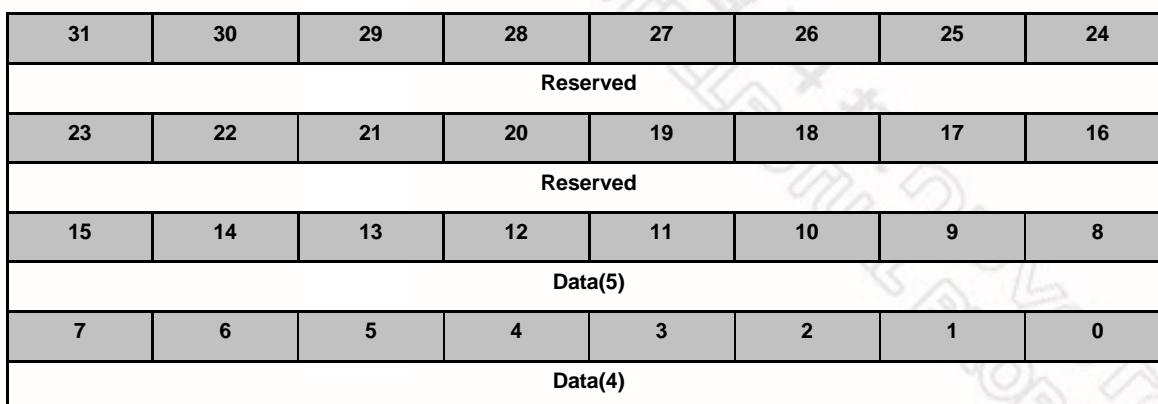
寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_DAT_A2	CAN0_BA+0x40/0xA0	R/W	IFn 数据 A2 寄存器	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:8]	Data (3)	数据字节 3 CAN 数据帧的第 4 个数据字节
[7:0]	Data (2)	数据字节 2 CAN 数据帧的第 3 个数据字节

IFn 数据 B1 寄存器 (CAN_IFn_DAT_B1)

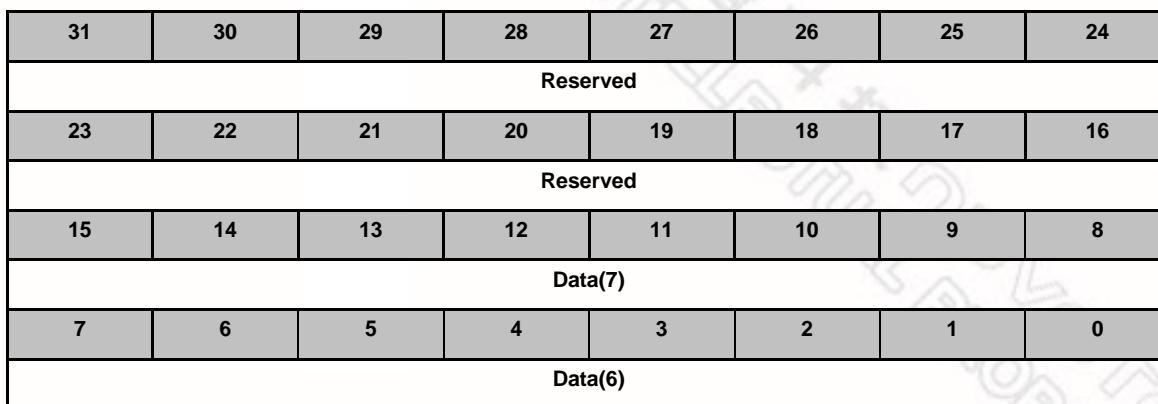
寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_DAT_B1	CAN0_BA+0x44/0xA4	R/W	IFn 数据 B1 寄存器	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:8]	Data (5)	数据字节 5 CAN 数据帧的第 6 个数据字节
[7:0]	Data (4)	数据字节 4 CAN 数据帧的第 5 个数据字节

IFn 数据 B2 寄存器 (CAN_IFn_DAT_B2)

寄存器	偏移量	R/W	描述	复位后的值
CAN_IFn_DAT_B2	CAN0_BA+0x48/0xA8	R/W	IFn 数据 B2 寄存器	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:8]	Data (7)	数据字节 7 CAN 数据帧的第 8 个数据字节。
[7:0]	Data (6)	数据字节 6 CAN 数据帧的第 7 个数据字节。

在一个 CAN 数据帧中，数据(0)是第一个，数据(7)是传送或接收到最后一个字节。在 CAN 的串行位流，每个字节的 MSB 将被先传送。

报文内存中的报文对象

在报文 RAM 中有 32 个报文对象。为了避免应用软件访问报文 RAM 和 CAN 报文的接收发送之间的冲突，CPU 不能直接访问报文对象，这些访问存取都要通过 IF_n 接口寄存器处理。表 5-20 提供了一个报文对象结构的概述。

报文对象												
UMask	Msk [28:0]	MXtd	MDir	EoB	NewDat		MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID [28:0]	Xtd	Dir	DLC [3:0]	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7

表 5-20 报文内存中的报文对象的结构

仲裁寄存器 **ID28-0**, **Xtd** 和 **Dir** 被用来定义传出报文的标识符和类型及用于（和屏蔽寄存器 **Msk28-0**, **MXtd** 和 **MDir**一起）传入报文的接收过滤。一条接收到的报文被保存在具有相匹配标识符的可用报文对象中，其方向 = 接收（数据帧）或方向 = 发送（远程帧）。扩展帧只能被保存在 **Xtd = 1** 的报文对象中，标准帧保存在 **Xtd = 0** 的报文对象中。如果一条接收到的报文（数据帧或远程帧）和1 个以上的可用报文对象匹配，那么它将被保存在最低报文号的报文对象中。

报文处理器寄存器

所有的报文处理器寄存器都是只读。它们的内容（每个报文对象的 **TxRqst**, **NewDat**, **IntPnd** 和 **MsgVal** 位以及众多标识符）是由报文处理器 FSM 提供的状态信息。

传送请求寄存器 1 (CAN_TXREQ1)

这些寄存器保持 32 个报文对象的 **TxRqst** 位。通过读取 **TxRqst** 位，软件可以检查哪个报文对象的传送请求挂起。应用软件（通过 **IFn** 报文接口寄存器）或者报文处理器（在收到一个远程帧后或者一次成功的传送后），可以来置位/复位指定报文对象的 **TxRqst** 位。

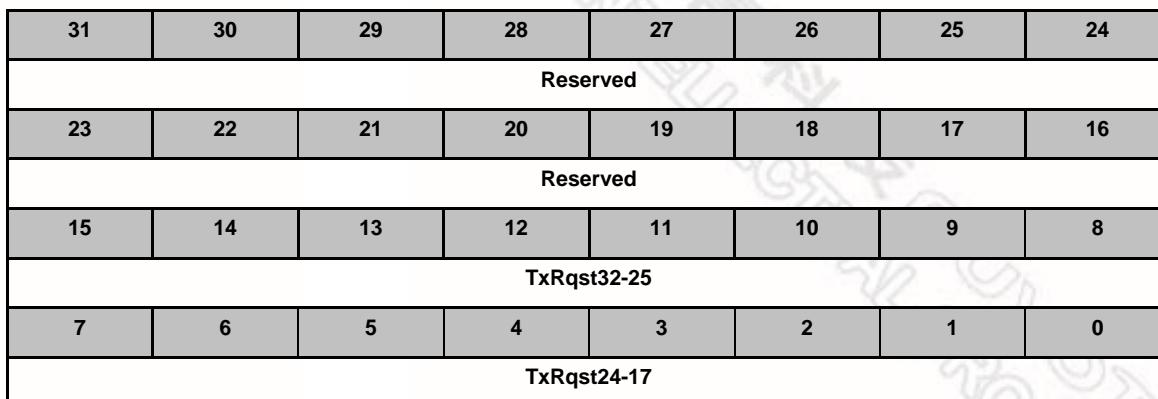
寄存器	偏移量	R/W	描述	复位后的值
CAN_TXREQ1	CAN0_BA+0x100	R	传送请求寄存器 1	0x0000_0000

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

Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:0]	TxRqst 16-1	传送请求位 16-1 (所有报文对象的) 1 = 表示该报文对象的传送请求而且还没有完成。 0 = 该报文对象不在等待传送。 这些位为只读。

传送请求寄存器 2 (CAN_TXREQ2)

寄存器	偏移量	R/W	描述	复位后的值
CAN_TXREQ2	CAN0_BA+0x104	R	传送请求寄存器 2	0x0000_0000

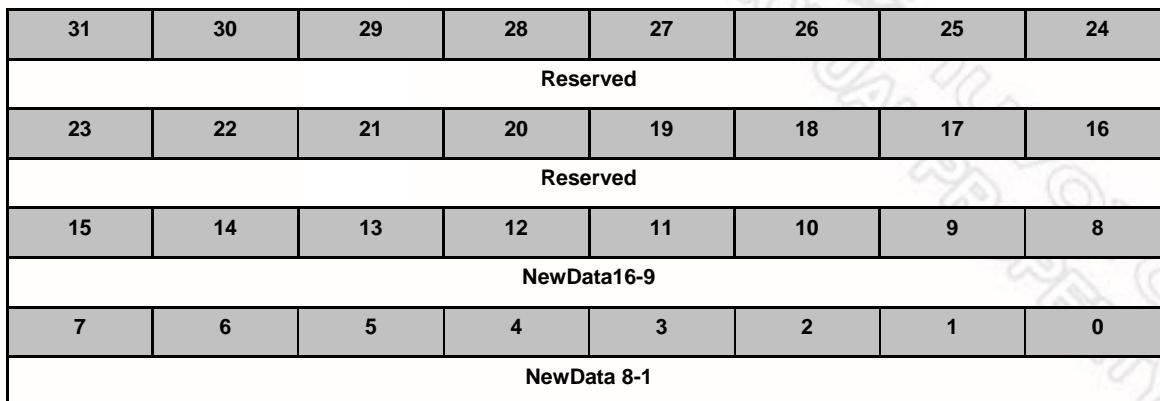


Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:0]	TxRqst 32-17	传送请求位 32-17 (所有报文对象的) 1 = 表示该报文对象的传送请求而且还没有完成。 0 = 该报文对象不在等待传送。 这些位为只读。

新数据寄存器 1 (CAN_NDAT1)

这些寄存器保持 32 个报文对象的 **NewDat** 位。通过读取 **NewDat** 位，软件可以检查哪个报文对象的数据部分被更新了。应用软件（通过 IFn 报文接口寄存器）或者报文处理器（在收到一个数据帧后或者一次成功的传送后），可以来置位/复位指定报文对象的 **NewDat** 位。

寄存器	偏移量	R/W	描述	复位后的值
CAN_NDAT1	CAN0_BA+0x120	R	新数据寄存器 1	0x0000_0000

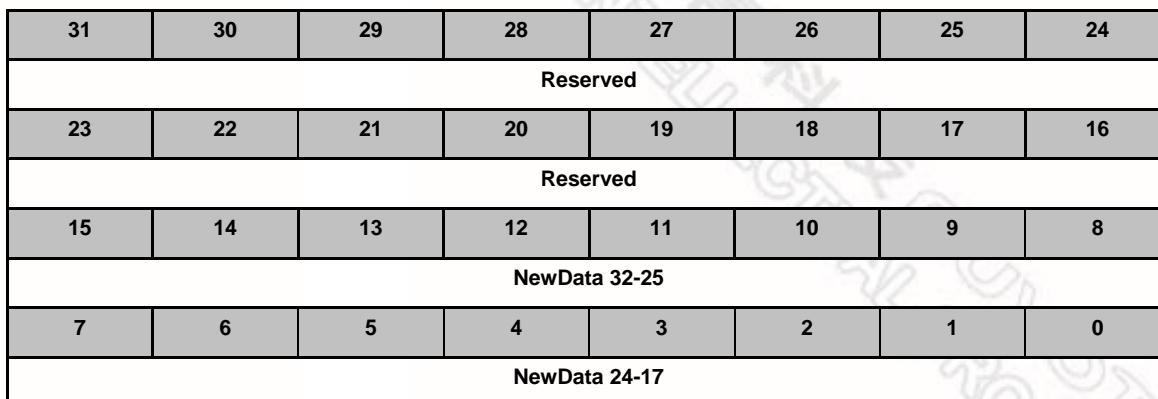


Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:0]	NewData16-1	新数据位 16-1 (所有报文对象的) 1 = 表示报文处理器或应用软件已经写入了新数据到该报文对象的数据部分。 0 = 自最后一次该标志被应用软件清除之后，该报文对象的数据部分没有被报文处理器写入新数据。



新数据寄存器 2 (CAN_NDAT2)

寄存器	偏移量	R/W	描述	复位后的值
CAN_NDAT2	CAN0_BA+0x124	R	新数据寄存器 2	0x0000_0000

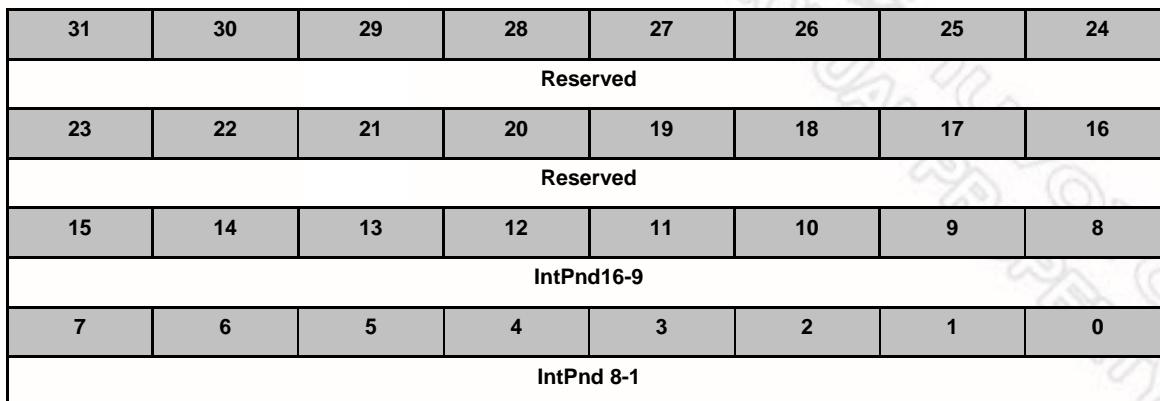


Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:0]	NewData 32-17	新数据位 32-17 (所有报文对象的) 1 = 表示报文处理器或应用软件已经写入了新数据到该报文对象的数据部分。 0 = 自最后一次该标志被应用软件清除之后，该报文对象的数据部分没有被报文处理器写入新数据。

中断挂起寄存器 1 (CAN_IPND1)

这些寄存器包含 32 个报文对象的 **IntPnd** 位。通过读取 **IntPnd** 位，软件可以检查哪个报文对象的中断挂起。应用软件（通过 **IFn** 报文接口寄存器）或者报文处理器（在收到一帧后或者一次成功的帧传送后），可以来置位/复位指定报文对象的 **IntPnd** 位。这也将会影响中断寄存器 **IntId** 的值。

寄存器	偏移量	R/W	描述	复位后的值
CAN_IPND1	CAN0_BA+0x140	R	中断挂起寄存器 1	0x0000_0000

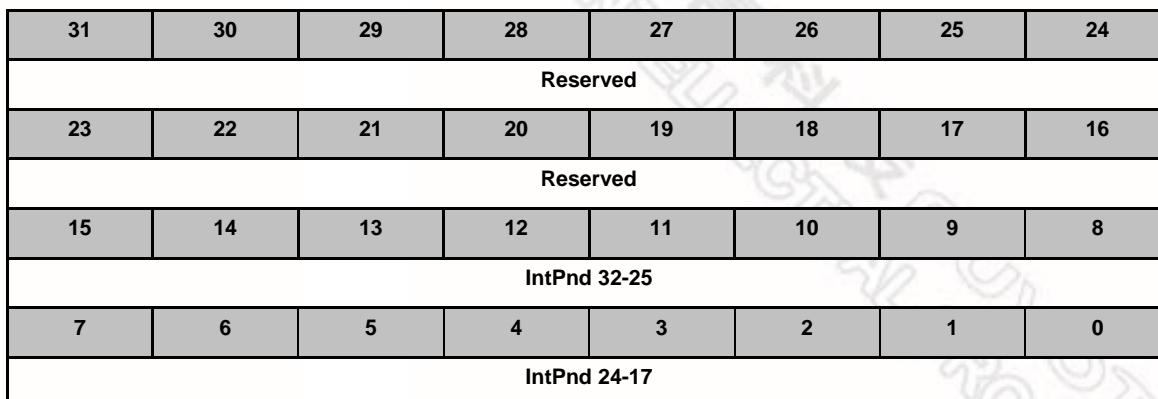


Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:0]	IntPnd16-1	中断挂起位 16-1 (所有报文对象的) 1 = 表示该报文对象是中断源。 0 = 表示该报文对象不是中断源。



中断挂起寄存器 2 (CAN_IPND2)

寄存器	偏移量	R/W	描述	复位后的值
CAN_IPND2	CAN0_BA+0x144	R	中断挂起寄存器 2	0x0000_0000

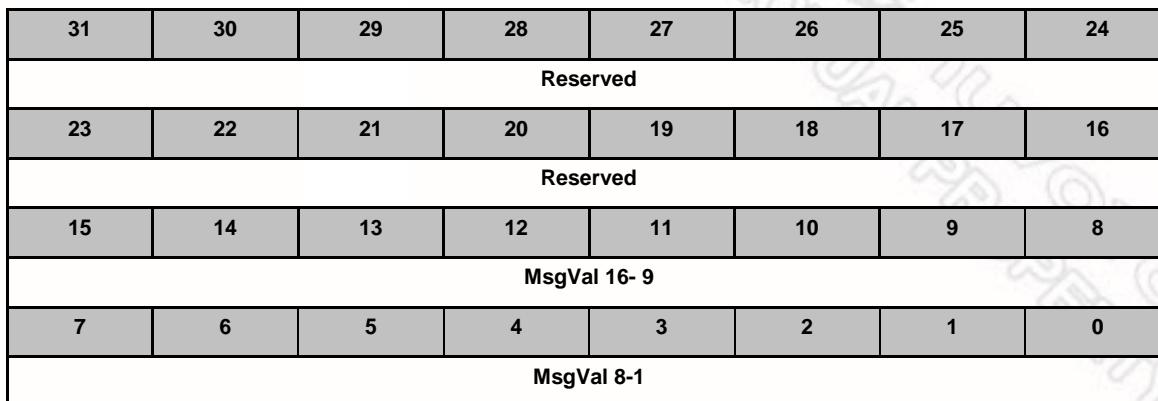


Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:0]	IntPnd 32-17	中断挂起位 32-17 (所有报文对象的) 1 = 表示该报文对象是中断源。 0 = 表示该报文对象不是中断源。

报文有效寄存器 1 (CAN_MVLD1)

这些寄存器保持 32 个报文对象的 **MsgVal** 位。通过读取 **MsgVal** 位，应用软件可以检查哪个报文对象是有效的。应用软件（通过 **IFn** 报文接口寄存器）可以来置位/复位指定报文对象的 **MsgVal** 位。

寄存器	偏移量	R/W	描述	复位后的值
CAN_MVLD1	CAN0_BA+0x160	R	报文有效寄存器 1	0x0000_0000

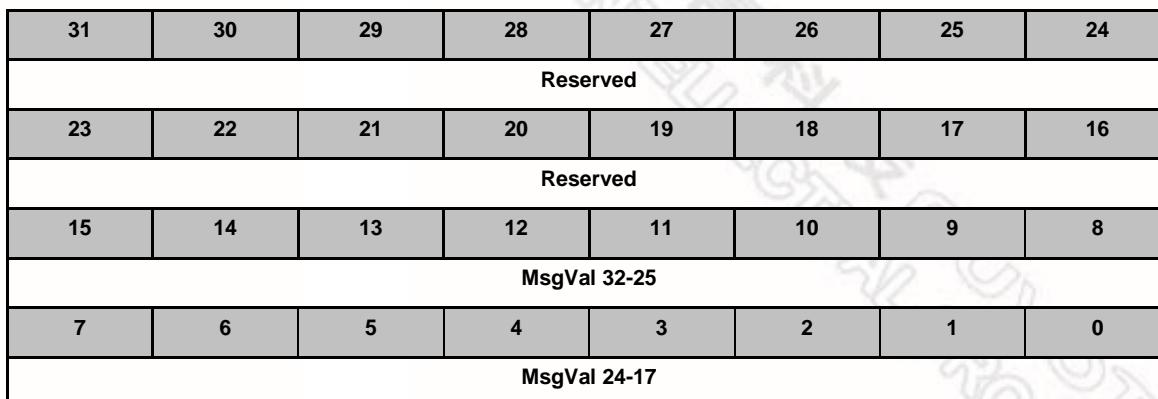


Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:0]	MsgVal 16-1	报文有效位 16-1 (所有报文对象的) (只读) 1 = 表示该报文对象被配置，可以被报文处理器考虑。 0 = 该报文对象被报文处理器忽略。 例如：CAN_MVLD1[0] 表示报文对象 No.1 是否有效。如果 CAN_MVLD1[0] 被置位，则报文对象 No.1 为已被配置的。



报文有效寄存器 2 (CAN_MVLD2)

寄存器	偏移量	R/W	描述	复位后的值
CAN_MVLD2	CAN0_BA+0x164	R	报文有效寄存器 2	0x0000_0000



Bits	描述	
[31:16]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[15:0]	MsgVal 32-17	报文有效位 32-17 (所有报文对象的) (只读) 1 = 表示该报文对象被配置，可以被报文处理器考虑。 0 = 该报文对象被报文处理器忽略。 例如：CAN_MVLD2[15] 表示报文对象 No.32 是否有效。如果 CAN_MVLD2[15] 被置位，则报文对象 No.32 为已被配置的。



唤醒使能寄存器 (CAN_WU_EN)

寄存器	偏移量	R/W	描述	复位后的值
CAN_WU_EN	CAN0_BA+0x168	R/W	唤醒使能寄存器	0x0000_0000

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

Bits	描述	
[31:1]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[0]	WAKUP_EN	唤醒使能 1 = 唤醒功能使能。 0 = 唤醒功能禁用。 注： 当在 CAN_Rx 有一个下降沿时，用户可以唤醒系统。



唤醒状态寄存器 (CAN_WU_STATUS)

寄存器	偏移量	R/W	描述	复位后的值
CAN_WU_STATUS	CAN0_BA+0x16C	R/W	唤醒状态寄存器	0x0000_0000

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

Bits	描述	
[31:1]	Reserved	保留 该位为保留位。该位读时总是为 '0'，也必须总是写 '0' 到该位。
[0]	WAKUP_STS	唤醒状态 1 = 发生了唤醒事件。 0 = 没有唤醒事件发生。 注：该位可以写 '0' 清除。



5.14 PS/2 设备控制器 (PS2D)

5.14.1 概述

PS/2 设备控制器为 PS/2 通讯提供基本的时序控制。所有在设备和主机之间的通讯都是通过 CLK 和 DATA 管脚管理的。不同于 PS/2 键盘和鼠标设备控制器，接收/传送代码需要固件进行代码转换成有意义的代码。在接收到发送请求后设备控制器启动发送 CLK 信号，但是在通信过程中主机拥有最终的控制权。主机发送到设备的数据 (DATA) 是在上升沿读取，设备发送到主机的数据在上升沿之后被改变。设备向主机发送数据。一个 16 字节的 FIFO 被应用于减少 CPU 的介入。S/W 可选择 1 ~ 16 字节的连续传输。

5.14.2 特征

- 主机通讯禁止和发送侦测请求
- 接收帧错误侦测
- 可编程的 1~16 字节传送缓存用以减少 CPU 干扰
- 双数据缓冲功能
- S/W override 总线

5.14.3 框图

PS/2 设备控制器包括 APB 接口和时序控制逻辑，用于 DATA 和 CLK 线。

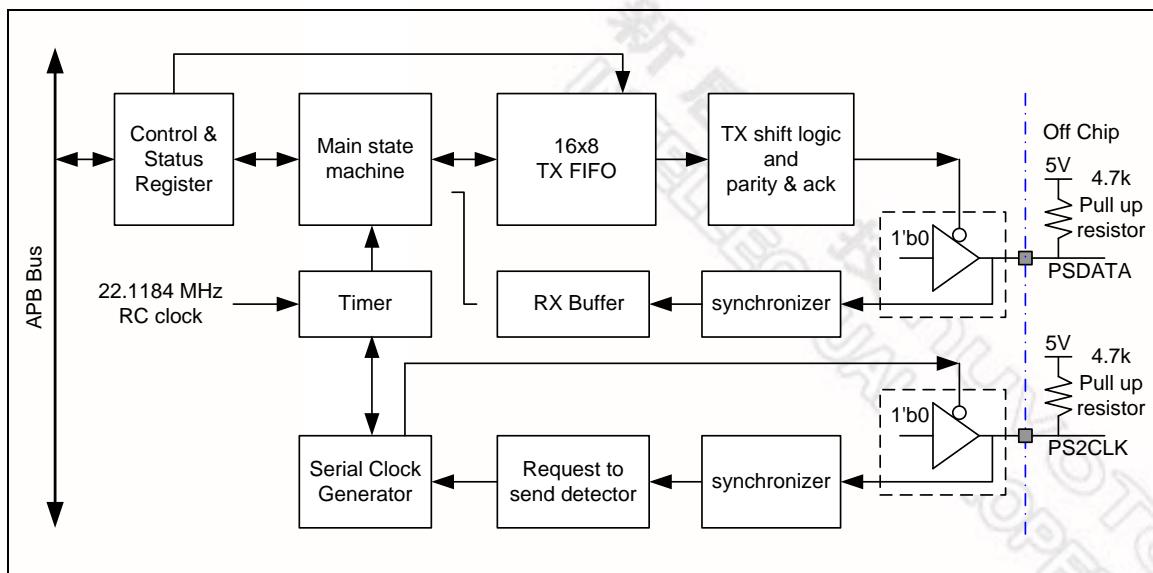


图 5-87 PS/2 设备框图

5.14.4 功能描述

5.14.4.1 通信

PS/2 设备具有双向同步串行协议。当两条线都为高 (open-collector) 时，总线为 "Idle"。该状态为设备允许开始传送 DATA 的唯一状态。主机在总线上有最终的控制权，并且任何时刻都可以通过拉低 CLK 线来禁止通讯。

CLK 信号由 PS/2 设备产生。如果主机需要发送 DATA，主机必须先拉低 CLK 来禁止通信。然后主机拉低 DATA 并释放 CLK。这是 "Request-to-Send" 状态，通知设备开始产生 CLK 脉冲。

DATA	CLK	总线状态
High	High	Idle
High	Low	禁止通讯
Low	High	主机请求发送

所有数据每次传输 1 字节，每字节被以包含 11 或 12 位的帧的形式发送。这些位包括：

- 1 开始位。一直为 0。
- 8 DATA 位，最低有效位优先
- 1 奇偶校验位（奇数校验）
- 1 停止位。一直为 1。
- 1 应答位（只在主机到设备的通讯中）

如果数据位中有偶数个 1，则奇偶校验位被置位，如果数据位中有奇数个 1，则奇偶校验位被清为零。数据位中 1 的个数加上奇偶校验位总是为奇数。这用于错误检测。设备必须检查该位，如果该位错误，则设备必须像收到了一个无效的命令般进行应答。

主机可以在任何时刻通过拉低 CLK 至少 100 微秒来禁止通讯。如果在第 11 个时钟脉冲之前，传输被禁止了，则设备必须放弃当前的传输并准备在当主机释放时钟时重新传输当前数据。为了保留足够的时间用于 S/W 来解码主机命令，传送逻辑被 RXINT 位封锁，S/W 必须清除 RXINT 位来开始重新传输。在必要时，S/W 能够写 CLRFIFO 为 1 来复位 FIFO 指针。

设备向主机传输：

设备使用 11-位帧的串行协议，如下：

- 1 开始位。一直为 0。
- 8 DATA 位，最低有效位优先
- 1 奇偶校验位（奇数校验）
- 1 停止位。一直为 1。

当 CLK 为高时，设备写一位在 DATA 线上，当 CLK 为低时，主机读该数据。图 5-88 描述该过程：

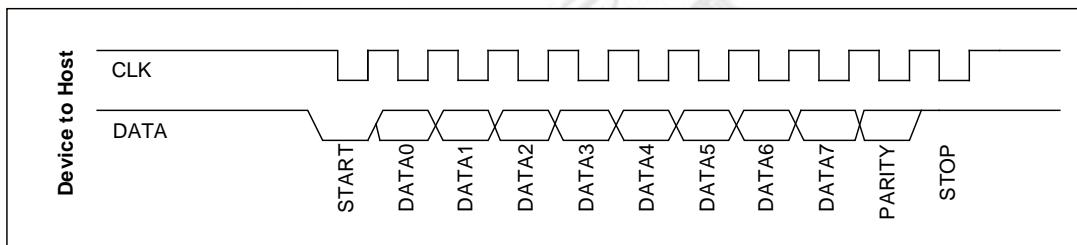


图 5-88 设备向主机传输的数据格式

主机向设备传输:

首先，PS/2 设备总是产生 CLK 信号。如果主机希望发送 DATA，首先必须设定 CLK 和 DATA 线在 "Request-to-send" 状态，如下：

- 拉低 CLK 至少 100 微秒来禁止通讯
- 拉低 DATA 应用 "Request-to-send"，然后释放 CLK

设备每隔一段不超过 10 ms 的时间间隔进行检测该状态。当设备检测到该状态，将开始产生 CLK 信号和 8 个数据位和一个停止位。当 CLK 为低时，主机改变 DATA 线，当 CLK 为高时，设备读取 DATA。

在接收到停止位后，设备将通过使 DATA 线为低并且产生最后一个 CLK 脉冲的方式，来应答收到的字节。如果主机在第 11 个 CLK 脉冲后未释放 DATA 线，则设备将继续产生 CLK 脉冲直到 DATA 线被释放。

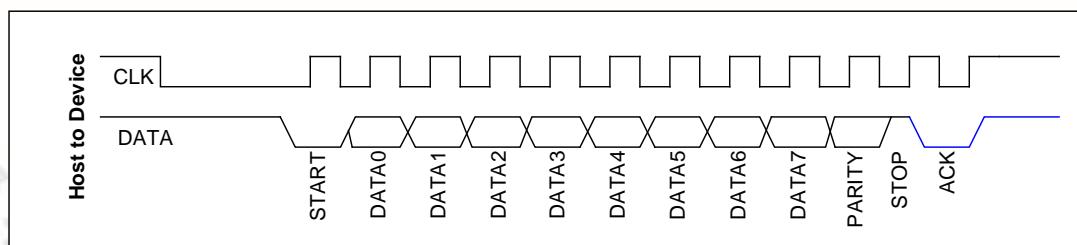


图 5-89 主机向设备传输的数据格式

主机和设备通讯的 DATA 和 CLK 详细的时序框图如下：

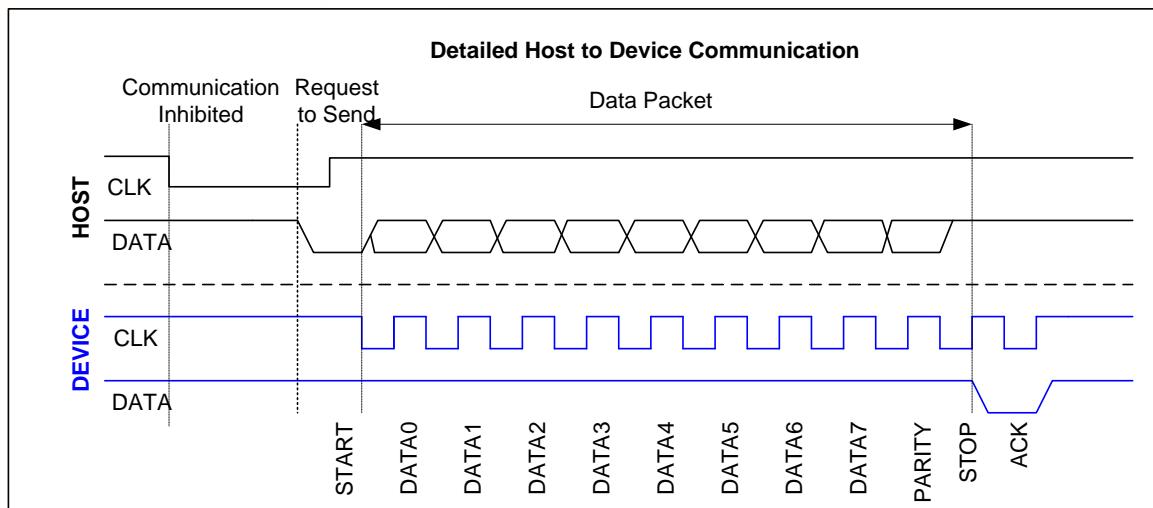


图 5-90 PS/2 位数据格式

5.14.4.2 PS/2 总线时序规范

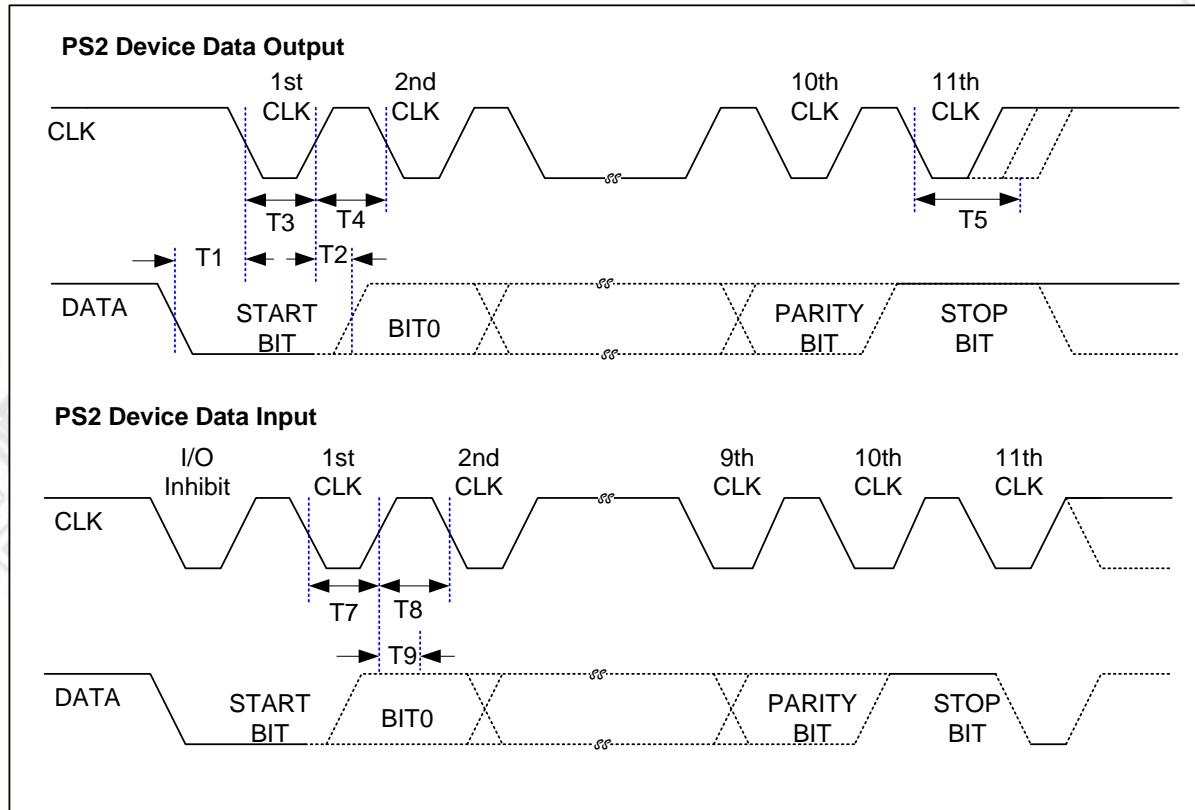


图 5-91 PS/2 总线时序



新唐科技 NUVOTON
INTELLECTUAL PROPERTY

新唐科技 NUVOTON
INTELLECTUAL PROPERTY



符号	时间参数	最小值.	最大值
T1	DATA 过渡 到 CLK 下降沿	5us	25us
T2	CLK 上升沿到 DATA 过渡	5us	T4-5us
T3	CLK 无效的持续时间	30us	50us
T4	CLK 有效的持续时间	30us	50us
T5	第11 个时钟后保证辅助设备不会开始另一个传输, 辅助设备禁止的时间	>0	50us
T7	CLK 无效的持续时间	30us	50us
T8	CLK 有效的持续时间	30us	50us
T9	CLK 从无效到有效转变的时间, 辅助设备取样 DATA 时间	5us	25us

5.14.4.3 TX FIFO 操作

写 PS2TXDATA0 寄存器开始设备到主机的传输。在写传送数据到 TX FIFO 之前，S/W 需要定义 TXFIFO 的长度。在 S/W 写 TX FIFO 后的 100us，第一个 START 位被发送的到 PS/2 总线。如果要传送的数据超过四个字节，S/W 在第四个字节数据传输完成前可写剩余数据到 PS2TXDATA1-3。2 个连续字节之间将增加100us的延时。

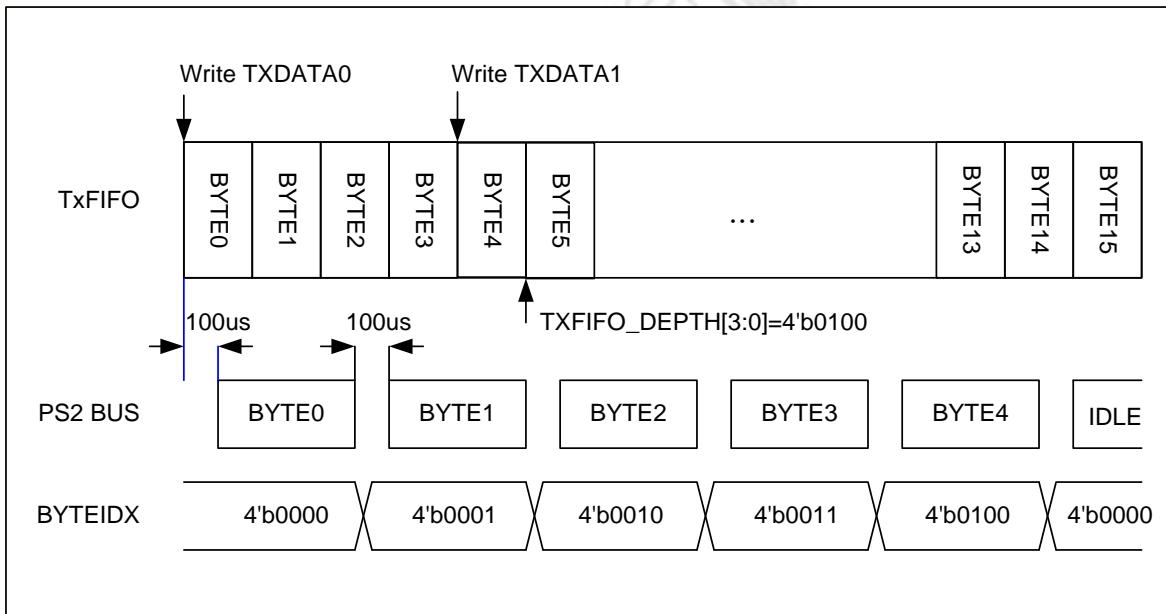


图 5-92 PS/2 数据格式



5.14.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
PS2_BA: 0x4010_0000				
PS2CON	PS2_BA+0x00	R/W	PS/2 控制寄存器	0x0000_0000
PS2TXDATA0	PS2_BA+0x04	R/W	PS/2 传输数据寄存器 0	0x0000_0000
PS2TXDATA1	PS2_BA+0x08	R/W	PS/2 传输数据寄存器 1	0x0000_0000
PS2TXDATA2	PS2_BA+0x0C	R/W	PS/2 传输数据寄存器 2	0x0000_0000
PS2TXDATA3	PS2_BA+0x10	R/W	PS/2 传输数据寄存器 3	0x0000_0000
PS2RXDATA	PS2_BA+0x14	R	PS/2 接收数据寄存器	0x0000_0000
PS2STATUS	PS2_BA+0x18	R/W	PS/2 状态寄存器	0x0000_0083
PS2INTID	PS2_BA+0x1C	R/W	PS/2 中断标识寄存器	0x0000_0000



5.14.6 寄存器描述

PS/2 控制寄存器 (PS2CON)

寄存器	偏移量	R/W	描述	复位后的值
PS2CON	PS2_BA+0x00	R/W	PS/2 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				FPS2DAT	FPS2CLK	OVERRIDE	CLRFIFO
7	6	5	4	3	2	1	0
ACK	TXFIFO_DEPTH				RXINTEN	TXINTEN	PS2EN

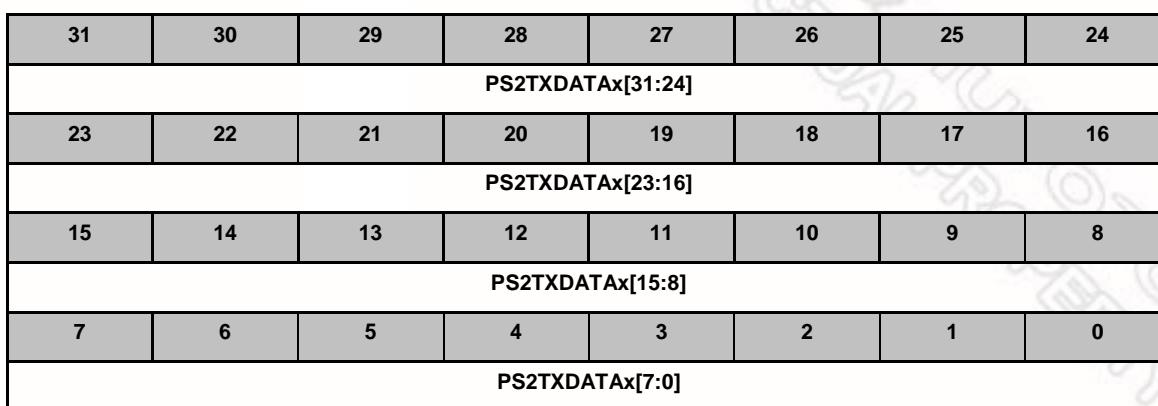
Bits	描述	
[31:12]	Reserved	保留
[11]	FPS2DAT	强制 PS2DATA Line 如果 OVERRIDE 被置为高，该位将强制 PS2DATA 为高或低，而不管设备控制器的内部状态。 1 = 强制 PS2DATA 为高 0 = 强制 PS2DATA 为低
[10]	FPS2CLK	强制 PS2CLK Line 如果 OVERRIDE 被置为高，该位将强制 PS2CLK line 为高或低，而不管设备控制器的内部状态。 1 = 强制 PS2CLK line 为高 0 = 强制 PS2CLK line 为低
[9]	OVERRIDE	软件无视 PS/2 CLK/DATA 管脚状态 1 = PS2CLK 和 PS2DATA 管脚由 S/W 控制。 0 = PS2CLK 和 PS2DATA 管脚由内部状态机控制。
[8]	CLRFIFO	清 TX FIFO 写 1 到该位终止设备到主机的传输。PS2STATUS 的 TXEMPTY 位将被置 1，并且指针 BYTEINDEX 被复位为0，而不管缓存中是否有残留的数据。缓存的内容将不会被清除。 1 = 清除 FIFO 0 = 无效
[7]	ACK	应答使能 1 = 如果奇偶错误或停止位未被正确接收，则在第 12 个时钟不会发送应答位。

		0 = 在主机到设备的通讯的第 12 个时钟总是发送应答位。
[6:3]	TXFIFODIPTH	<p>数据传输 FIFO 长度</p> <p>16 字节缓存用于数据传输。S/W 可根据应用定义 FIFO 的长度从 1 到 16 字节。</p> <p>0 = 1 字节 1 = 2 字节 ... 14 = 15 字节 15 = 16 字节</p>
[2]	RXINTEN	<p>使能接收中断</p> <p>1 = 使能数据接收完成中断 0 = 禁用数据接收完成中断</p>
[1]	TXINTEN	<p>使能传输中断</p> <p>1 = 使能数据传输完成中断 0 = 禁用数据传输完成中断</p>
[0]	PS2EN	<p>使能 PS/2 设备</p> <p>使能 PS/2 设备控制器</p> <p>1 = 使能 0 = 禁用</p>



PS/2 TX 数据寄存器 0-3 (PS2TXDATA0-3)

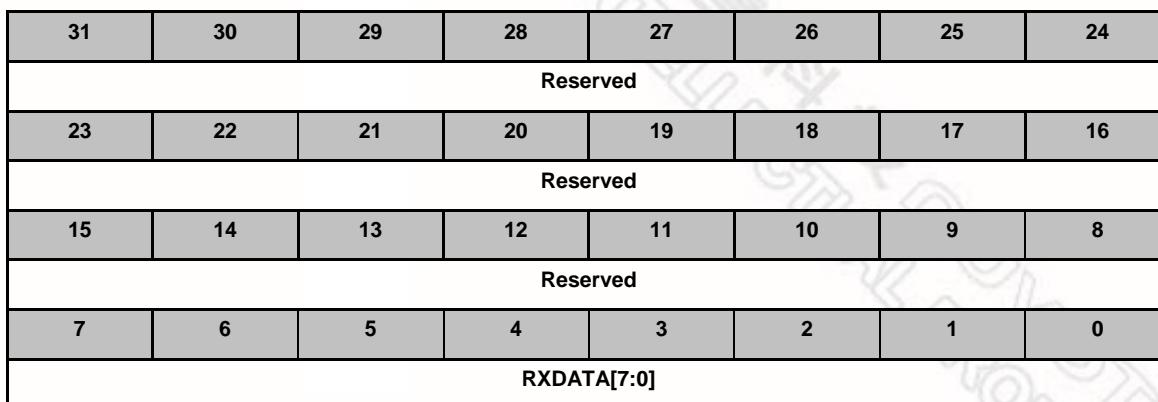
寄存器	偏移量	R/W	描述	复位后的值
PS2TXDATA0	PS2_BA+0x04	R/W	PS/2 数据传输寄存器 0	0x0000_0000
PS2TXDATA1	PS2_BA+0x08	R/W	PS/2 数据传输寄存器 1	0x0000_0000
PS2TXDATA2	PS2_BA+0x0C	R/W	PS/2 数据传输寄存器 2	0x0000_0000
PS2TXDATA3	PS2_BA+0x10	R/W	PS/2 数据传输寄存器 3	0x0000_0000



Bits	描述	
[31:0]	PS2TXDATAx	传输数据 当总线在 IDLE 状态时，写数据到该寄存器开始设备到主机的通讯。S/W 必须在写数据到 TX 缓存之前使能 PS2EN。

PS/2 接收数据寄存器 (PS2RXDATA)

寄存器	偏移量	R/W	描述	复位后的值
PS2RXDATA	PS2_BA+0x14	R	PS/2 接收数据寄存器	0x0000_0000



Bits	描述	
[31:8]	Reserved	保留
[7:0]	PS2RXDATA	接收数据 在主机到设备传输中，在应答位发送后，接收数据将从移位寄存器拷贝到 PS2RXDATA 寄存器。CPU 需在下一字节接收完成前读取此寄存器，否则数据将被改写而无效，并且 RXOVF 位 (PS2STATUS[6]) 将被置 1。

PS/2 状态寄存器 (PS2STATUS)

寄存器	偏移量	R/W	描述	复位后的值
PS2STATUS	PS2_BA+0x18	R/W	PS/2 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				BYTEIDX[3:0]			
7	6	5	4	3	2	1	0
TXEMPTY	RXOVF	TXBUSY	RXBUSY	RXPARTY	FRAMERR	PS2DATA	PS2CLK

Bits	描述						
[31:12]	Reserved	保留					
[11:8]	BYTEIDX	字节索引 该位表示哪个数据字节在传输数据移位寄存器。当 FIFO 中的所有数据传输完毕时，该位将被清为 0。 该位为只读位。					
		BYTEIDX	DATA Transmit	BYTEIDX			
		0000	TXDATA0[7:0]	1000			
		0001	TXDATA0[15:8]	1001			
		0010	TXDATA0[23:16]	1010			
		0011	TXDATA0[31:24]	1011			
		0100	TXDATA1[7:0]	1100			
		0101	TXDATA1[15:8]	1101			
		0110	TXDATA1[23:16]	1110			
		0111	TXDATA1[31:24]	1111			
[7]	TXEMPTY	TX FIFO 空 如果 PS2EN 使能，当 S/W 写任何数据到 PS2TXDATA0-3 时，TXEMPTY 位将立即被清为 0。当传输的数据字节数和 FIFODEPTH 相等，则 TXEMPTY 位将置 1。 1 = FIFO 为空 0 = 有数据被传输 只读位					

[6]	RXOVF	RX Buffer 改写 1 = PS2RXDATA 寄存器中的数据被新来的数据改写了 0 = 无改写 写 1 清除该位。
[5]	TXBUSY	发送忙 该位表示 PS/2 设备当前正在发送数据。 1 = 当前正在发送数据 0 = 空闲 只读位
[4]	RXBUSY	接收忙 该位表示 PS/2 设备当前正在接收数据。 1 = 当前正在接收数据 0 = 空闲 只读位。
[3]	RXPARITY	接收奇偶 该位反映最后接收到的数据字节的奇偶位（奇数奇偶校验）。 只读位。
[2]	FRAMERR	帧错误 对于主机到设备的通讯，如果未收到 STOP 位（逻辑1）则表示帧错误。如果帧错误发生了，DATA 线在第 12 个时钟之后仍保持低状态。此时，S/W 无视 PS2CLK 一直发送时钟直到 PS2DATA 释放成高状态。然后，设备发送一个“Resend”命令给主机。 1 = 发生帧错误 0 = 未发生帧错误 写 1 清除该位。
[1]	PS2DATA	DATA 管脚状态 该位反映在同步和采样后 PS2DATA 线的状态。
[0]	PS2CLK	CLK 管脚状态 该位反映在同步后 PS2CLK 线的状态。

PS/2 中断标识寄存器 (PS2INTID)

寄存器	偏移量	R/W	描述	复位后的值
PS2INTID	PS2_BA+0x1C	R/W	PS/2 中断标识寄存器	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						TXINT	RXINT

Bits	描述	
[31:3]	Reserved	保留
[1]	TXINT	<p>传送中断 在 STOP 位传送后, 该位被置为 1。如果 TXINTEN 位被置为1, 则中断产生。 1 = 传送中断发生 0 = 无中断 写 1 清该位为 0。</p>
[0]	RXINT	<p>接收中断 在主机向设备传输时, 当应答位发送, 该位被置为 1。如果 RXINTEN 位被置为1, 则中断产生。 1 = 接收中断发生 0 = 无中断 写 1 清该位为 0。</p>



5.15 I²S 控制器 (I²S)

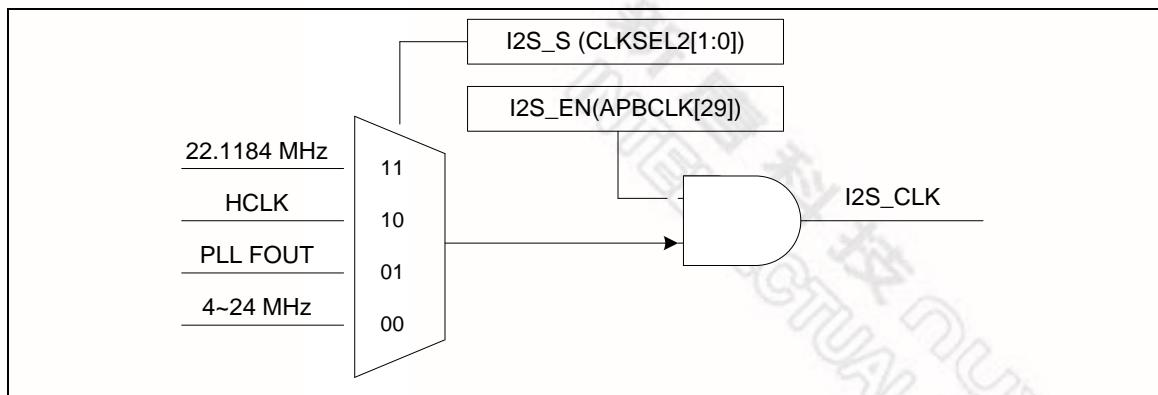
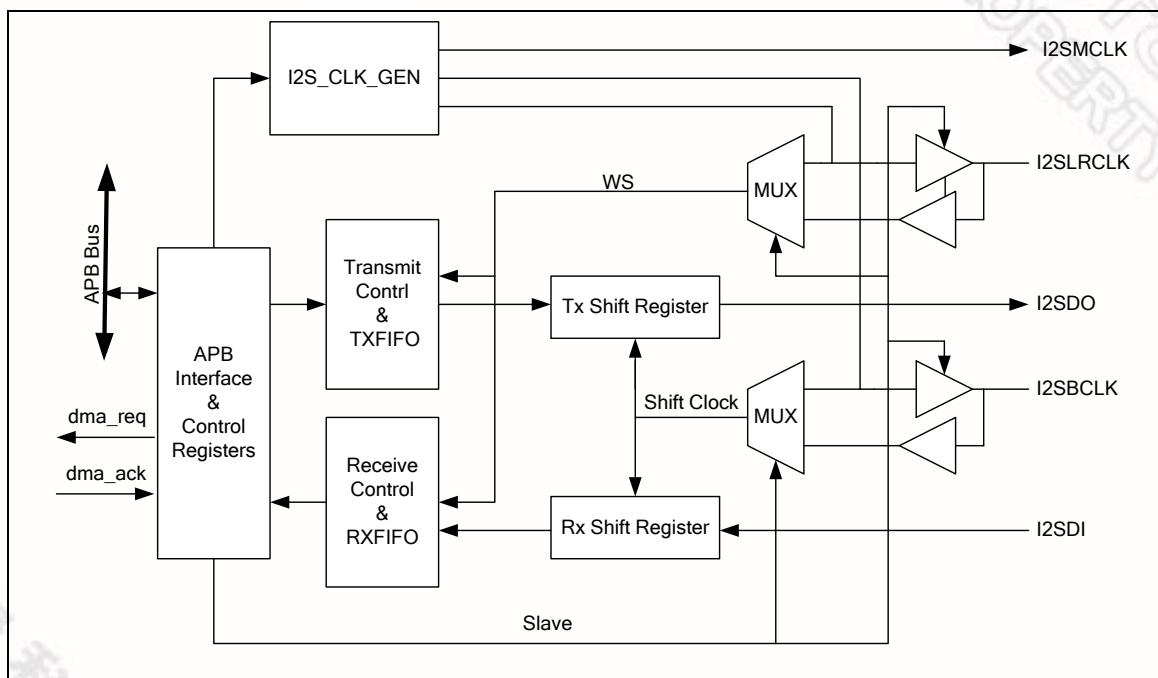
5.15.1 概述

I²S 控制器由 IIS 协议与外部音频 CODEC 接口组成。两个 8 字的 FIFO 分别用于读和写通道，可以处理 8 ~ 32 位字大小。DMA 控制器处理 FIFO 和内存之间的数据移动。

5.15.2 特征

- I²S 可工作与主机模式或从机模式
- 可处理 8-, 16-, 24- 和 32-位字大小
- 支持单声道和立体声的音频数据
- 支持 I²S 和 MSB 校正数据格式
- 提供两个 8 字的 FIFO 数据缓存，一个用于发送，一个用于接收
- 当缓存超过可编程边界时，产生中断请求
- 两个 DMA 请求，一个用于发送，一个用于接收

5.15.3 框图

图 5-93 I²S 时钟控制框图图 5-94 I²S 控制器框图

5.15.4 功能描述

5.15.4.1 I²S 操作

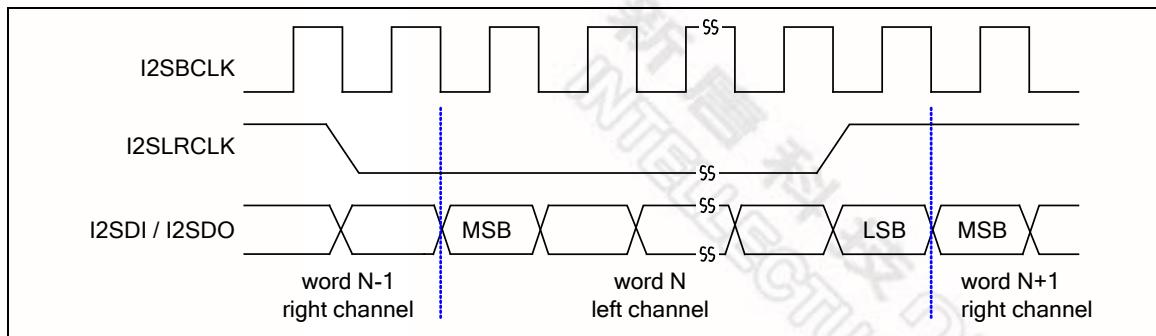


图 5-95 I²S 总线时序图 (Format=0)

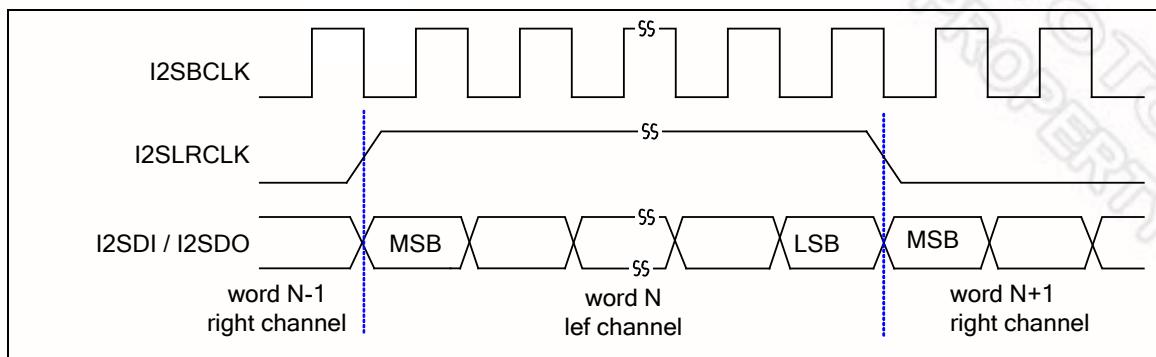


图 5-96 MSB 校正 (MSB Justified) 时序图 (Format=1)

5.15.4.2 FIFO 操作

Mono 8-bit data mode								
<table border="1"> <tr> <td>N+3 7</td> <td>0</td> <td>7</td> <td>N+2 0</td> <td>7</td> <td>N+1 0</td> <td>7</td> <td>N 0</td> </tr> </table>	N+3 7	0	7	N+2 0	7	N+1 0	7	N 0
N+3 7	0	7	N+2 0	7	N+1 0	7	N 0	
Stereo 8-bit data mode								
<table border="1"> <tr> <td>LEFT+1 7</td> <td>0</td> <td>7</td> <td>RIGHT+1 0</td> <td>7</td> <td>LEFT 0</td> <td>7</td> <td>RIGHT 0</td> </tr> </table>	LEFT+1 7	0	7	RIGHT+1 0	7	LEFT 0	7	RIGHT 0
LEFT+1 7	0	7	RIGHT+1 0	7	LEFT 0	7	RIGHT 0	
Mono 16-bit data mode								
<table border="1"> <tr> <td colspan="2">N+1 15</td> <td>0</td> <td>15</td> <td>N 0</td> </tr> </table>	N+1 15		0	15	N 0			
N+1 15		0	15	N 0				
Stereo 16-bit data mode								
<table border="1"> <tr> <td colspan="2">LEFT 15</td> <td>0</td> <td>15</td> <td>RIGHT 0</td> </tr> </table>	LEFT 15		0	15	RIGHT 0			
LEFT 15		0	15	RIGHT 0				
Mono 24-bit data mode								
<table border="1"> <tr> <td>23</td> <td>N 0</td> </tr> </table>	23	N 0						
23	N 0							
Stereo 24-bit data mode								
<table border="1"> <tr> <td>23</td> <td>LEFT 0</td> <td>N</td> </tr> </table>	23	LEFT 0	N					
23	LEFT 0	N						
<table border="1"> <tr> <td>23</td> <td>RIGHT 0</td> <td>N+1</td> </tr> </table>	23	RIGHT 0	N+1					
23	RIGHT 0	N+1						
Mono 32-bit data mode								
<table border="1"> <tr> <td>31</td> <td>N 0</td> </tr> </table>	31	N 0						
31	N 0							
Stereo 32-bit data mode								
<table border="1"> <tr> <td>31</td> <td>LEFT 0</td> <td>N</td> </tr> </table>	31	LEFT 0	N					
31	LEFT 0	N						
<table border="1"> <tr> <td>31</td> <td>RIGHT 0</td> <td>N+1</td> </tr> </table>	31	RIGHT 0	N+1					
31	RIGHT 0	N+1						

图 5-97 不同 I²S 模式的 FIFO 内容



5.15.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
I2S_BA = 0x401A_0000				
I2S_CON	I2S_BA+0x00	R/W	I ² S 控制寄存器	0x0000_0000
I2S_CLKDIV	I2S_BA+0x04	R/W	I ² S 时钟分频寄存器	0x0000_0000
I2S_IE	I2S_BA+0x08	R/W	I ² S 中断使能寄存器	0x0000_0000
I2S_STATUS	I2S_BA+0x0C	R/W	I ² S 状态寄存器	0x0014_1000
I2S_TXFIFO	I2S_BA+0x10	R/W	I ² S 传送 FIFO 寄存器	0x0000_0000
I2S_RXFIFO	I2S_BA+0x14	R/W	I ² S 接收 FIFO 寄存器	0x0000_0000



5.15.6 寄存器描述

I²S Control Register (I2S_CON)

寄存器	偏移量	R/W	描述	复位后的值
I2S_CON	I2S_BA+0x00	R/W	I ² S 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	Reserved	RXDMA	TXDMA	CLR_RXFIFO	CLR_TXFIFO	LCHZCEN	RCHZCEN
15	14	13	12	11	10	9	8
MCLKEN	RXTH[2:0]			TXTH[2:0]			SLAVE
7	6	5	4	3	2	1	0
FORMAT	MONO	WORDWIDTH		MUTE	RXEN	TXEN	I2SEN

Bits	描述	
[31:22]	Reserved	保留
[21]	RXDMA	使能接收 DMA 当 RX DMA 使能时, 如果 FIFO 非空, I ² S 请求 DMA 从接收 FIFO 向 SRAM 传输数据。 1 = 使能 RX DMA 0 = 禁用 RX DMA
[20]	TXDMA	使能传送 DMA 当 TX DMA 使能时, 如果 FIFO 没满, I ² S 请求 DMA 从 SRAM 向发送 FIFO 传输数据。 1 = 使能 TX DMA 0 = 禁用 TX DMA
[19]	CLR_RXFIFO	清除接收 FIFO 写 1 清接收 FIFO, 内部指针复位指向 FIFO 的起始位置, RXFIFO_LEVEL[3:0] 返回 0, 接收 FIFO 变为空。 该位由硬件自动清零, 读时返回0。
[18]	CLR_TXFIFO	清除发送 FIFO 写 1 清发送 FIFO, 内部指针复位指向 FIFO 的起始位置, TXFIFO_LEVEL[3:0] 返回 0, 发送 FIFO 变为空, 但在发送 FIFO 中的数据不变。 该位由硬件自动清零, 读时返回0。

[17]	LCHZCEN	使能左声道过零检测 如果该位置 1，当左声道数据符号位改变或下一个移位数据位都为0，则寄存器 I2S_STATUS 的 LZCF 标志被置为 1。 1 = 使能左声道过零检测 0 = 禁用左声道过零检测
[16]	RCHZCEN	使能右声道过零检测 如果该位置 1，当右声道数据符号位改变或下一个移位数据位都为0，则寄存器 I2S_STATUS 的 RZCF 标志被置为 1。 1 = 使能右声道过零检测 0 = 禁用右声道过零检测
[15]	MCLKEN	使能主时钟 如果 NuMicro™ NUC100 系列，外部晶振时钟频率为 $2^N \times 256\text{fs}$ ，则软件可以编程寄存器 I2S_CLKDIV 的 MCLK_DIV[2:0] 位，以得到音频 CODEC 芯片所需的256fs 时钟。 1 = 使能主时钟 0 = 禁用主时钟
[14:12]	RXTH[2:0]	接收 FIFO 阈值水平 当缓存中的接收字等于或大于阈值水平时，RXTHF 标志置位。 000 = 接收 FIFO 中有 1 字数据 001 = 接收 FIFO 中有 2 字数据 010 = 接收 FIFO 中有 3 字数据 011 = 接收 FIFO 中有 4 字数据 100 = 接收 FIFO 中有 5 字数据 101 = 接收 FIFO 中有 6 字数据 110 = 接收 FIFO 中有 7 字数据 111 = 接收 FIFO 中有 8 字数据
[11:9]	TXTH[2:0]	发送 FIFO 阈值水平 如果发送 FIFO 中剩下的数据字 (32 bits) 等于或小于阈值水平，则 TXTHF 标志置位。 000 = 发送 FIFO 中有 0 个字数据 001 = 发送 FIFO 中有 1 个字数据 010 = 发送 FIFO 中有 2 个字数据 011 = 发送 FIFO 中有 3 个字数据 100 = 发送 FIFO 中有 4 个字数据 101 = 发送 FIFO 中有 5 个字数据 110 = 发送 FIFO 中有 6 个字数据 111 = 发送 FIFO 中有 7 个字数据

[8]	SLAVE	从机模式 I ² S 可工作于主机模式或从机模式。主机模式下，I2S_BCLK 和 I2S_LRCLK 管脚都为输出模式并作为 NuMicro™ NUC100 系列发送时钟到音频 CODEC 芯片。从机模式下，I2S_BCLK 和 I2S_LRCLK 管脚都为输入模式，接收来自外部音频 CODEC 芯片的 I2S_BCLK 和 I2S_LRCLK 信号。 1 = 从机模式 0 = 主机模式
[7]	FORMAT	数据格式 1 = MSB 校正数据格式 0 = I ² S 数据格式
[6]	MONO	单声道数据 1 = 数据为单声道格式 0 = 数据为立体声格式
[5:4]	WORDWIDTH	字宽度 00 = 数据为 8-位 01 = 数据为 16-位 10 = 数据为 24-位 11 = 数据为 32-位
[3]	MUTE	使能发送静音 1= 发送通道数据为 0 0 = 发送数据由缓存移动
[2]	RXEN	接收使能 1 = 使能数据接收 0 = 禁用数据接收
[1]	TXEN	发送使能 1 = 使能数据发送 0 = 禁用数据发送
[0]	I2SEN	使能 I²S 控制器 1 = 使能 0 = 禁用

I²S 时钟分频 (I2S_CLKDIV)

寄存器	偏移量	R/W	描述	复位后的值
I2S_CLKDIV	I2S_BA+0x04	R/W	I ² S 时钟分频控制寄存器	0x0000_0000

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

Bits	描述	
[31:16]	Reserved	保留
[15:8]	BCLK_DIV [7:0]	<p>位时钟分频 如果 I²S 工作于主机模式，位时钟由 NuMicro™ NUC100 系统提供。软件可以编程这些位产生采样率时钟频率。 $F_{BCLK} = F_{I2SCLK} / (2 \times (BCLK_DIV + 1))$</p>
[7:3]	Reserved	保留
[2:0]	MCLK_DIV[2:0]	<p>主时钟分频 如果芯片的外部晶振频率为 $(2 \times MCLK_DIV) \times 256\text{fs}$，则软件可编程这些位产生 256fs 时钟频率到音频 CODEC 芯片。如果 MCLK_DIV 被置为 0，则 MCLK 与外部时钟输入相同。 例如，采样率为 24 kHz，芯片的外部晶振时钟为 12.288 MHz，设置 MCLK_DIV=1。 $F_{MCLK} = F_{I2SCLK} / (2 \times (MCLK_DIV))$ (When MCLK_DIV is ≥ 1) $F_{MCLK} = F_{I2SCLK}$ (When MCLK_DIV is set to 0)</p>

I²S 中断使能寄存器 (I2S IE)

寄存器	偏移量	R/W	描述	复位后的值
I2S_IE	I2S_BA+0x08	R/W	I ² S 中断使能寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			LZCIE	RZCIE	TXTHIE	TXOVFIE	TXUDFIE
7	6	5	4	3	2	1	0
Reserved					RXTHIE	RXOVFIE	RXUDFIE

Bits	描述	
[31:13]	Reserved	保留
[12]	LZCIE	左声道过零检测中断使能 如果该位设为 1 且 左声道有过零信号，则中断产生。 1 = 使能中断 0 = 禁用中断
[11]	RZCIE	右声道过零检测中断使能 1 = 使能中断 0 = 禁用中断
[10]	TXTHIE	发送 FIFO 阈值水平中断使能 如果该位为1，而且发送 FIFO 中的数据字小于 TXTH[2:0]，则中断发生。 1 = 使能中断 0 = 禁用中断
[9]	TXOVFIE	接收 FIFO 溢出中断使能 如果该位为1，而且发送 FIFO 的溢出标志设为 1，则中断发生。 1 = 使能中断 0 = 禁用中断
[8]	TXUDFIE	发送 FIFO 下溢中断使能 如果该位为1，而且发送 FIFO 的下溢标志设为 1，则中断发生。 1 = 使能中断 0 = 禁用中断

[7:3]	Reserved	保留
[2]	RXTHIE	<p>接收 FIFO 阈值水平中断使能</p> <p>当接收 FIFO 中的数据字等于或大于 RXTH[2:0]，则 RXTHF 位被置为 1。如果 RXTHIE 位使能，则中断发生。</p> <p>1 = 使能中断 0 = 禁用中断</p>
[1]	RXOVFIE	<p>接收 FIFO 溢出中断使能</p> <p>1 = 使能中断 0 = 禁用中断</p>
[0]	RXUDFIE	<p>接收 FIFO 下溢中断使能</p> <p>如果软件读取接收 FIFO，当为空时，I2SSTATUS 寄存器的 RXUDF 位被置为 1。</p> <p>1 = 使能中断 0 = 禁用中断</p>

I²S 状态寄存器 (I2S_STATUS)

寄存器	偏移量	R/W	描述	复位后的值
I2S_STATUS	I2S_BA+0x0C	R/W	I ² S 状态寄存器	0x0014_1000

31	30	29	28	27	26	25	24
TX_LEVEL[3:0]				RX_LEVEL[3:0]			
23	22	21	20	19	18	17	16
LZCF	RZCF	TXBUSY	TXEMPTY	TXFULL	TXTHF	TXOVF	TXUDF
15	14	13	12	11	10	9	8
Reserved			RXEMPTY	RXFULL	RXTHF	RXOVF	RXUDF
7	6	5	4	3	2	1	0
Reserved				RIGHT	I2STXINT	I2SRXINT	I2SINT

Bits	描述	
[31:28]	TX_LEVEL	发送 FIFO 水平 这些位表示发送 FIFO 中数据字的数目。 0000 = 无数据 0001 = 发送 FIFO 有 1 个字 1000 = 发送 FIFO 有 8 个字
[27:24]	RX_LEVEL	接收 FIFO 水平 这些位表示接收 FIFO 中数据字的数目。 0000 = 无数据 0001 = 接收 FIFO 有 1 个字 1000 = 接收 FIFO 有 8 个字
[23]	LZCF	左声道过零标志 该位表示左声道下一个采样数据符号位已经改变或所有数据位为 0。 1 = 检测到左声道过零 0 = 没有过零 软件可以写 1 清除该位为 0。

[22]	RZCF	右声道过零标志 该位表示左声道下一个采样数据符号位已经改变或所有数据位为 0。 1 = 检测到右声道过零 0 = 没有过零 软件可以写 1 清除该位为 0。
[21]	TXBUSY	发送忙 当发送 FIFO 中的所有数据和移位缓存都被移出时，该位清零。当第一个数据加载到移位缓存时，该位置 1。 1 = 发送移位缓存忙 0 = 发送移位缓存为空 该位只读。
[20]	TXEMPTY	发送 FIFO 为空 该位反映发送 FIFO 中的数据字号为 0. 1 = 空 0 = 非空 该位只读。
[19]	TXFULL	发送 FIFO 满 该位反映发送 FIFO 中的数据字号为8。 1 = 满 0 = 没满. 该位只读。
[18]	TXTHF	发送 FIFO 阈值标志 当发送 FIFO 中的数据字等于或低于 TXTH[2:0] 中设定的阈值，则 TXTHF 位变为 1。该位将保持 1 直到软件写 TXFIFO 寄存器后 TXFIFO_LEVEL[3:0] 高于 TXTH[1:0]。 1 = FIFO 中的数据字等于或低于阈值水平 0 = FIFO 中的数据字高于阈值水平 该位只读。
[17]	TXOVF	发送 FIFO 溢出标志 当发送 FIFO 已满时写数据到该 FIFO，该位被置为 1。 1 = 溢出 0 = 没有溢出 软件写 1 清除该位为 0。
[16]	TXUDF	发送 FIFO 下溢标志 当发送 FIFO 为空且移位逻辑硬件从数据 FIFO 读数据,会导致该位被置为1。 1 = 下溢 0 = 没有下溢 软件写 1 清除该位为 0。

[15:13]	Reserved	保留
[12]	RXEMPTY	<p>接收 FIFO 为空</p> <p>该位表示接收 FIFO 中的数据字号为0 1 = 空 0 = 非空 该位只读。</p>
[11]	RXFULL	<p>接收 FIFO 满</p> <p>该位表示接收 FIFO 中的数据字号为8 1 = 满 0 = 未满 该位只读。</p>
[10]	RXTHF	<p>接收 FIFO 阈值标志</p> <p>当接收 FIFO 中数据字等于或高于 RXTH[2:0] 中所设定的阈值时，RXTHF 位 变为 1。 该位将保持 1直到软件读 RXFIFO 寄存器后 RXFIFO_LEVEL[3:0] 低于 RXTH[1:0]。 1 = FIFO 中数据字等于或高于阈值水平 0 = FIFO 中数据字低于阈值水平 该位只读。</p>
[9]	RXOVF	<p>接收 FIFO 溢出标志</p> <p>当接收 FIFO 满了时，接收硬件还尝试写数据到接收 FIFO，则该位被置为1，第一个缓存中数据被覆盖。 1 = 发生溢出 0 = 没发生溢出 软件写 1 清除该位为 0。</p>
[8]	RXUDF	<p>接收 FIFO 下溢标志</p> <p>当接收 FIFO 为空时，读接收 FIFO，则该位置 1表示发生下溢。 1 = 发生下溢 0 = 没有发生下溢 软件写 1 清除该位为 0。</p>
[7:4]	Reserved	保留
[3]	RIGHT	<p>右声道</p> <p>该位表示当前的发送数据属于右声道。 1 = 右声道 0 = 左声道 该位只读。</p>

[2]	I2STXINT	I²S 发送中断 1 = 发送中断 0 = 无发送中断 该位只读。
[1]	I2SRXINT	I²S 接收中断 1 = 接收中断 0 = 无接收中断 该位只读。
[0]	I2SINT	I²S 中断标志 1 = I ² S 中断 0 = 无 I ² S 中断 该位为 I2STXINT 和 I2SRXINT 位的“或”。 该位只读。

I²S 发送 (I2S_TXFIFO)

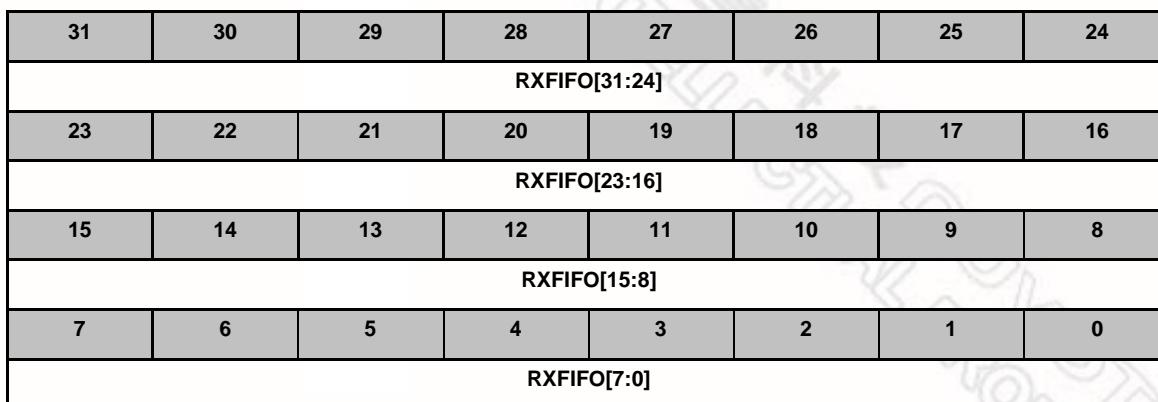
寄存器	偏移量	R/W	描述	复位后的值
I2S_TXFIFO	I2S_BA+0x10	R/W	I ² S 发送 FIFO	0x0000_0000

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

Bits	描述	
[31:0]	TXFIFO	I ² S 包含 8个字 (8x32 bit) 的数据缓存用于数据传送。写数据到该寄存器准备用于传送的数据。剩余的字数由 I2S_STATUS 的TX_LEVEL[3:0] 指示。

I²S 接收 FIFO (I²S_RXFIFO)

寄存器	偏移量	R/W	描述	复位后的值
I ² S_RXFIFO	I ² S_BA+0x14	R/W	I ² S 接收 FIFO	0x0000_0000



Bits	描述	
[31:0]	RXFIFO	<p>接收 FIFO 寄存器</p> <p>I²S 包含 8 个字 (8x32 bit) 的数据缓存用于数据接收。读该寄存器获取 FIFO 中的数据。</p> <p>剩下的数据字数目由 I²S_STATUS 寄存器的 RX_LEVEL[3:0] 指示。</p>



5.16 模拟数字转换 (ADC)

5.16.1 概述

NuMicro™ NUC100 系列包含一个 12-位 8 通道逐次逼近式的模拟 - 数字转换器 (SAR A/D converter)。A/D 转换器支持三种操作模式：单一 (single)，单周期扫描 (single-cycle scan) 和连续扫描模式 (continuous scan mode)。A/D 转换器可由软件和外部 STADC 管脚开启。

5.16.2 特征

- 模拟输入电压范围：0~Vref (Max to 5.0 V)
- 12-位分辨率和 10-位精确度保证
- 多达 8 路单端模拟输入通道或 4 路差分模拟输入通道
- 最大 ADC 时钟频率为 16 MHz
- 高达 700K SPS 转换速率
- 三种操作模式
 - 单一模式：A/D 转换在指定通道完成一次
 - 单周期扫描模式：A/D 转换在所有指定通道完成一个周期，转换顺序从最小号通道到最大号通道
 - 连续扫描模式：A/D 转换器持续执行单周期扫描直到软件停止 A/D 转换
- A/D 转换开始条件
 - 软件向 ADST 位写 1
 - 外部管脚 STADC
- 每个通道转换结果存储在数据寄存器内，并带有 valid/overrun 标志
- 转换结果可和指定的值相比较，当转换值和比较寄存器中的设定值相等时，用户可以选择是否产生一个中断请求
- 通道 7 支持 3 输入源：外部模拟电压，内部带隙电压和内部温度传感器输出
- 支持自校正功能以最小化转换误差

5.16.3 框图

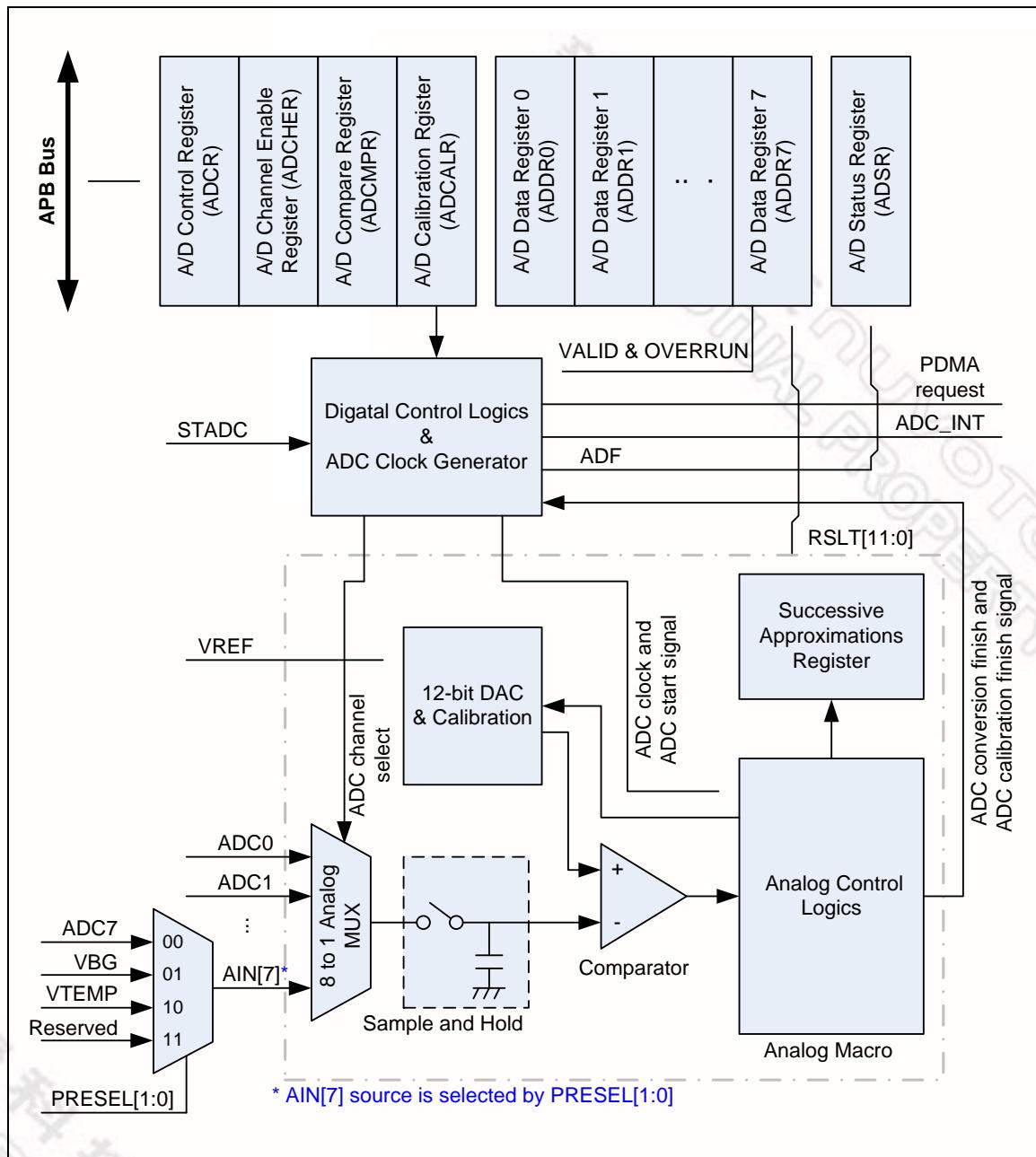


图 5-98 ADC 控制器框图

5.16.4 功能描述

A/D 转换器为 12 位逐次逼近式。A/D 转换器具有自校正功能以最小化转换误差，用户可写 1 到 ADCALR 寄存器的 CALEN 位使能自校正功能，当内部校正完成时，CAL_DONE 位将会被硬件置为高。ADC 有 3 种操作模式：单一模式、单周期扫描模式和连续扫描模式。当改变运行模式或模拟输入通道时，为了防止错误的操作，软件必须清 ADCR 寄存器中的 ADST 位为 0。

5.16.4.1 自校正

当芯片上电或者 ADC 的输入类型在单端输入与差分输入之间转换时，就需要做 ADC 自校正以最小化转换误差。用户写 1 到 CALEN 位（ADCALR 寄存器）开始自校正功能。完成校正需要 127 ADC 时钟，完成校正后内部硬件将会置 CAL_DONE 位为 1。其详细的时序图如下：

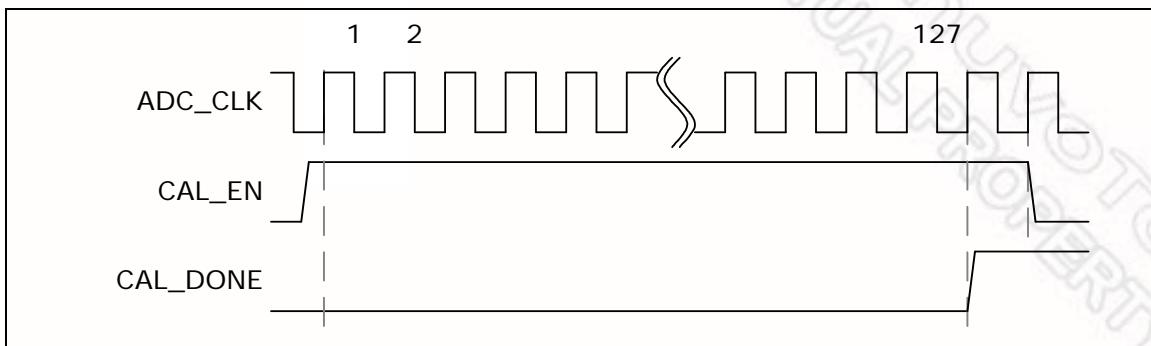


图 5-99 ADC 转换器自校正时序图

5.16.4.2 ADC 时钟发生器

最大采样率达 700K SPS。ADC 有四个时钟源，通过 2 位 ADC_S (CLKSEL[3:2]) 进行选择，ADC 时钟频率按如下公式进行 8 位预分频：

$$\text{ADC 时钟频率} = (\text{ADC 时钟源频率}) / (\text{ADC_N} + 1);$$

其中 8 位 ADC_N 在寄存器 CLKDIV[23:16] 中设置。

如果时钟源来自 HCLK，则 ADC_N 不能为 0。通常来说，软件可设置 ADC_S 和 ADC_N 获得 16 MHz 或稍低于 16MHz 的频率。

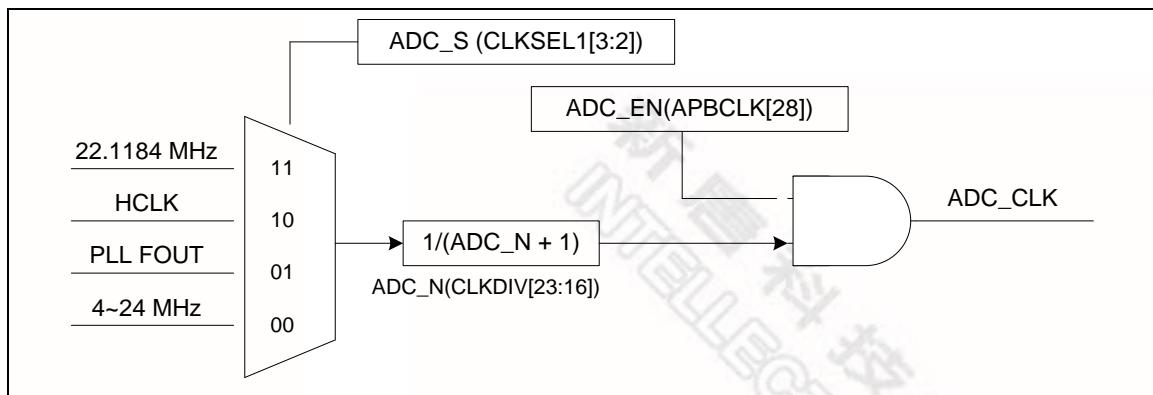


图 5-100 ADC 时钟控制

5.16.4.3 单一模式

在单一模式下，A/D 转换仅仅在指定的单一通道中执行一次，运作流程如下：

1. 当 ADCR 的 ADST 位被软件或者外部触发输入置位时，A/D 转换开始。
2. 当 A/D 转换完成，转换值将存储在与通道相对应的 A/D 数据寄存器。
3. A/D 转换完成，ADSR 的 ADF 位置位。若此时 ADCR 寄存器的 ADIE 位被置位，则将产生 ADC 中断请求。
4. 在 A/D 转换期间，ADST 位保持为 1。当 A/D 转换结束，ADST 位自动清 0，A/D 转换器进入 idle 状态。ADST 位被清为 0 后，ADST 位保持为 0 至少要一个 ADC 时钟周期，才能再将 ADST 位置为 1，否则 A/D 转换器可能无法工作。

注：如果在单一模式下，软件使能多于一个通道，则编号最小的通道将被选中，其他通道将被忽略。

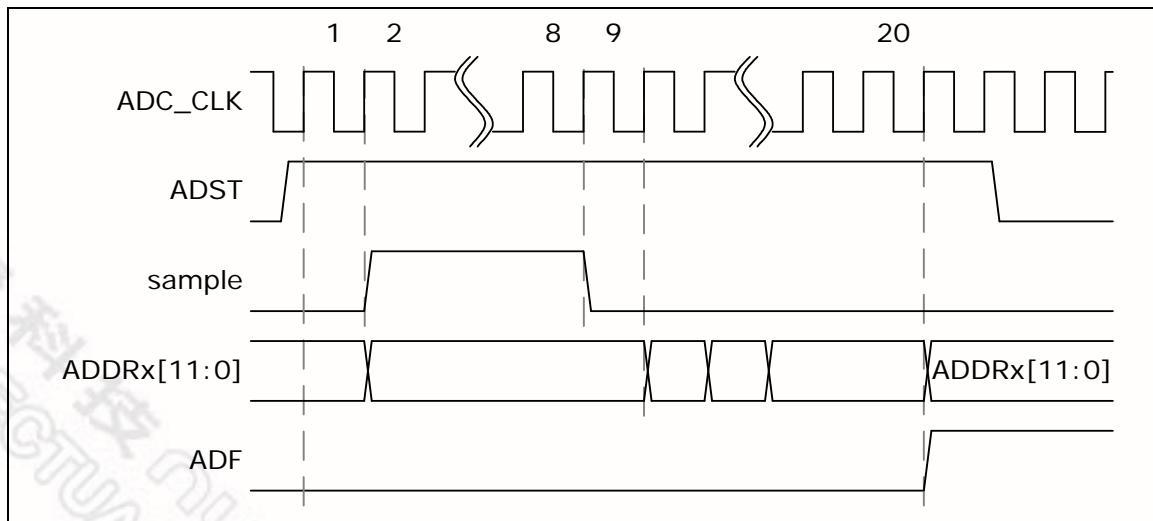


图 5-101 单一模式转换时序图

5.16.4.4 单周期扫描模式

在单周期扫描模式下，ADC 会对所有指定的通道进行一次采样与转换，且从编号最小的通道到编

号最大的通道依次开始。

1. 当 ADCR 的 ADST 被软件或者外部触发输入置位时, A/D 转换从最小编号的通道开始。
2. 每路 A/D 转换完成后, A/D 转换数值将传输到相应的 A/D 数据寄存器中。
3. 当所有被选择的通道转换都完成后, ADF 位 (ADSR 寄存器) 置1。若此时 ADC 中断功能使能, 则将产生 ADC 中断。
4. A/D 转换结束后, ADST 位自动清零, A/D 转换器进入 idle 状态。如果在所有使能 ADC 通道转换完成之前, ADST 位被清除为零, 则 ADC 控制器将完成当前转换, 最低使能 ADC 通道的结果将不可预测。ADST 位被清为 0 后, ADST 位保持为 0 至少要一个 ADC 时钟周期, 才能再将 ADST 位置为1, 否则 A/D 转换器可能无法工作。

一个使能通道 (0, 2, 3 and 7) 的单周期扫描模式时序图示例如下:

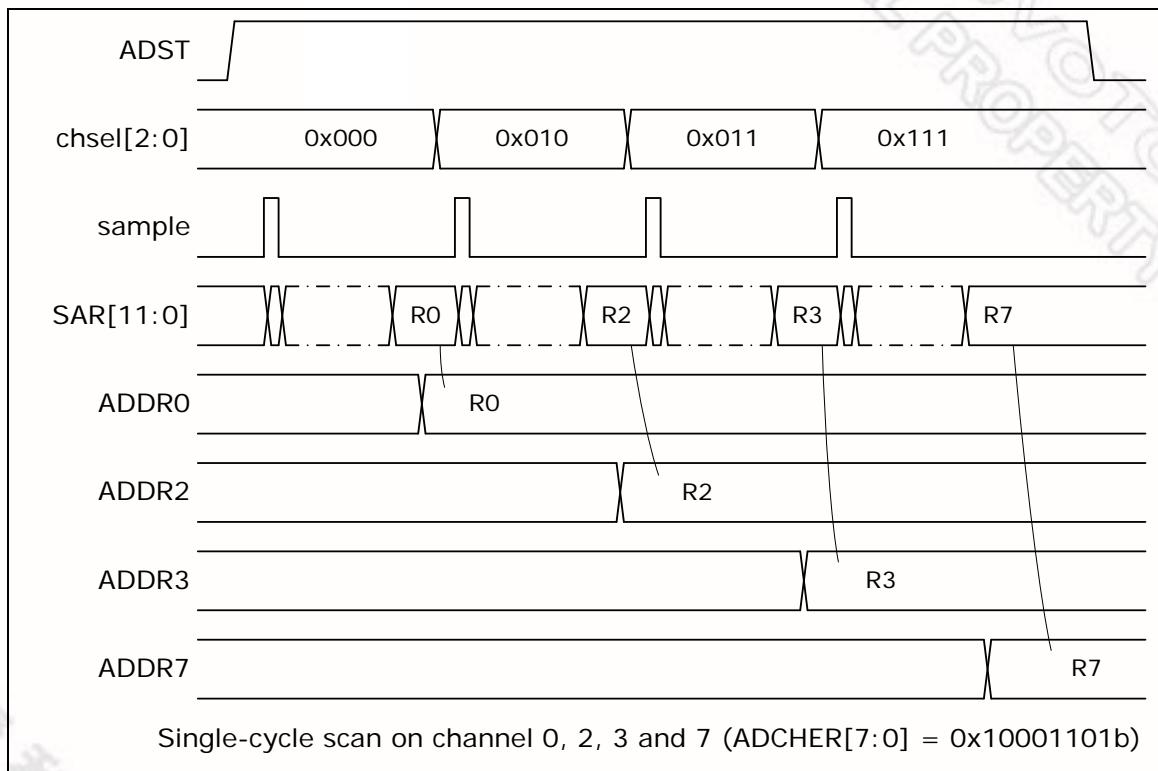


图 5-102 使能通道上的单周期扫描模式时序图

5.16.4.5 连续扫描模式

在连续扫描模式，A/D 转换器会连续对所有由 CHEN 位（ADCHER 寄存器）使能指定的通道（最多 8 ADC 通道）执行转换。操作如下：

1. 当 ADCR 的 ADST 被软件或者外部触发输入置位时，A/D 转换从最小编号的通道开始。
2. 每路使能的 A/D 转换完成后，A/D 转换数值将装载到相应的 A/D 数据寄存器中。
3. 当所有被使能的通道依次完成一次 A/D 转换时，ADF 位 (ADSR[0]) 将被置位。如果此时 ADC 中断功能已使能，则产生 ADC 中断。如果软件没有清除 ADST 位，则使能的最小编号通道将再次开始新的转换。
4. 只要 ADST 保持为 1，就重复步骤 2 到步骤 3。当清 ADST 为 0，ADC 控制器将完成当前转换，最小编号使能通道的转换结果数据将不可预测。

一个使能通道 (0, 2, 3 and 7) 的连续扫描模式时序图示例如下：

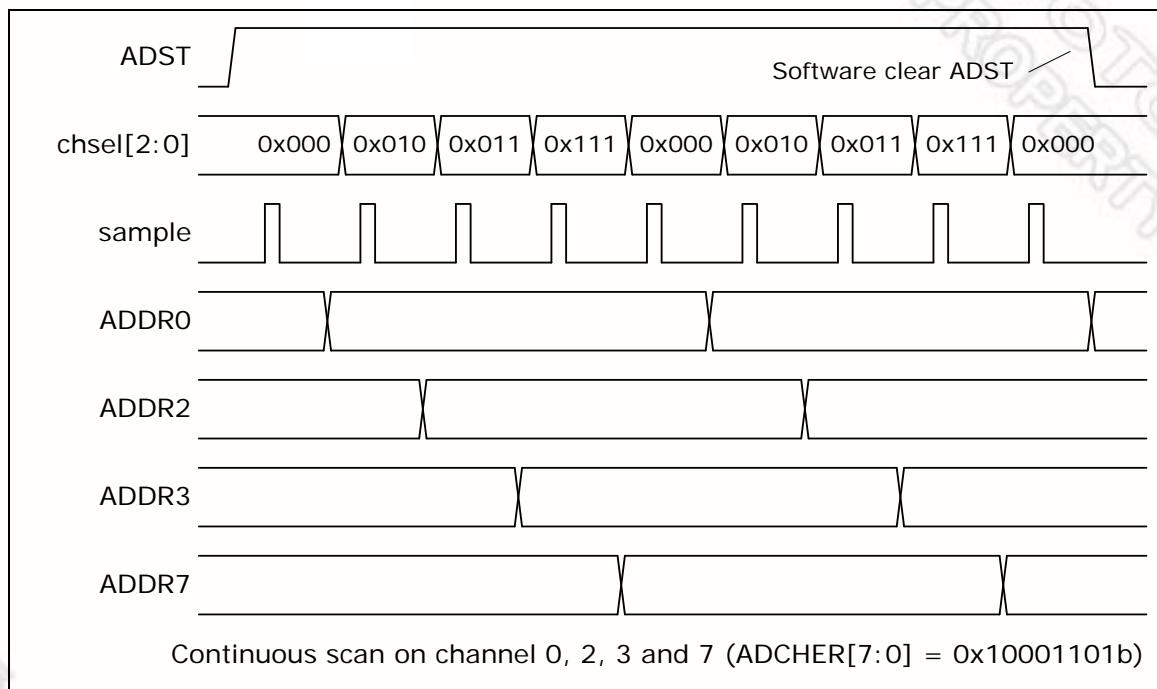


图 5-103 使能通道上的连续扫描模式时序图

5.16.4.6 外部触发输入采样和 A/D 转换时间

在单周期扫描模式下，A/D 转换可由外部管脚请求触发。当 ADCR 的 TRGEN 置为高来使能 ADC 外部触发功能，通过设定 TRGS[1:0] 位为 00b 选择从 STADC 管脚进行外部触发输入，并可藉由软件设定 TRGCOND[1:0] 选择上升/下降沿或低/高电平触发。若选择电平触发 (level trigger) 条件，STADC 管脚需要保持默认状态至少 8 个 PCLKs。ADST 位将在第九个 PCLK 时被置 1 并且开始转换。在电平触发模式状态下，如果外部触发输入维持在有效状态，转换将会持续进行。仅当外部触发条件消失才停止。若选择边缘触发模式，高和低状态至少需保持 4 个 PCLKs，低于该值的脉冲将被忽略。

5.16.4.7 比较功能监控的转换结果

ADC 控制器提供两组比较寄存器 ADCMPRO 和 ADCMPR1，用于监控 A/D 转换控制器（最多支持）2 路通道的转换结果值，可参考图 5-104。软件可通过设定 CMPCH(ADCMPRx[5:0]) 来选择监控哪路通道，而 CMPCOND 位被用来检查转换值小于指定值或大于（等于）CMPD[11:0] 的指定值。当 CMPCH 指定的通道转换完成时，比较行为将会被自动的触发一次。当比较结果和设定值相匹配，比较匹配计数器将加 1，否则，比较匹配计数器将会被清 0。当计数器的值和设定值 (CMPPATCNT+1) 匹配，CMPPF 位将置 1。如果 CMPIE 置位，将产生 ADC_INT 中断请求。在扫描模式下无需软件介入就可用其监控外部模拟输入管脚电压变化。具体逻辑框图如下：

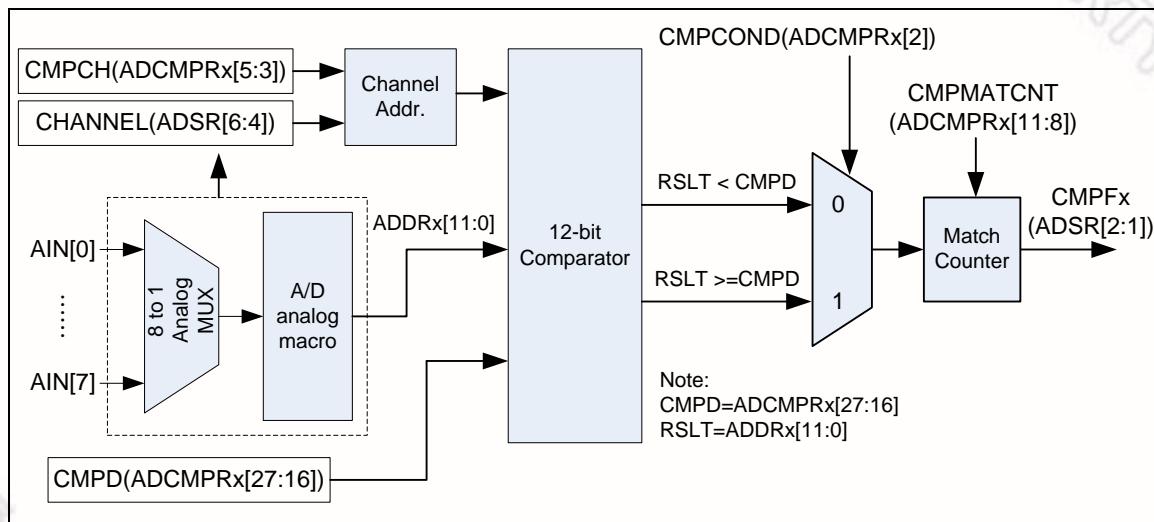


图 5-104 A/D 转换结果监控逻辑图

5.16.4.8 中断源

A/D 转换器有三个中断源。当某个 ADC 操作模式结束其转换时，A/D 转换的结束标志 ADF 位将被置 1。CMPF0 和 CMPF1 是比较功能的比较标志。当 A/D 转换结果同 ADCMPR0/1 寄存器设定值相匹配时，相应的比较标志 (CMPF0/1) 会被置 1。当 ADF、CMPF0 和 CMPF1 中任一个标志被置 1，且其相应中断使能位 ADIE (ACDR 寄存器) 及 CMPIE (ADCMR0/1 寄存器) 置 1 时，将产生 ADC 中断。软件可藉由清除标志位以撤销中断请求。

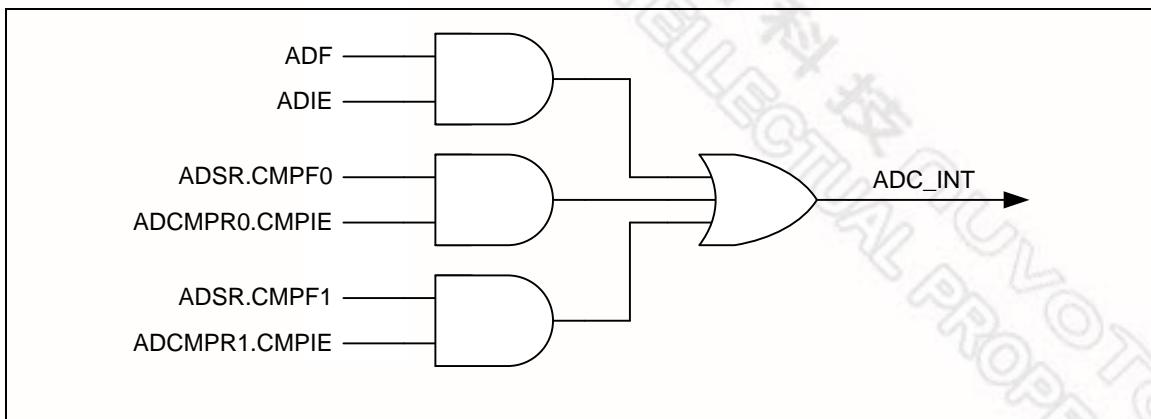


图 5-105 A/D 控制器中断

5.16.4.9 外设 DMA 请求

当 A/D 转换完成时，结果将被装载到 ADDR 寄存器且 VALID 位被置 1。如果 ACDR 寄存器的 PTEN 位被置位，ADC 控制器将产生请求到 PDMA。用户可使用 PDMA 将转换结果传输到用户指定的内存空间，而无需 CPU 参与。不管选择哪个通道，PDMA 操作的源地址都为 ADPDMA。如果 ADC 工作于单周期或连续扫描模式，当 PDMA 传输转换结果时，ADC 将继续转换所选择的下一个通道。用户可以通过读寄存器 ADPDMA 监控当前 PDMA 传输数据。如果 ADC 完成所选通道的转换，且相同通道上次的转换结果还没有被 PDMA 传输，则相应通道的 OVERUN 位将置位，上次 ADC 转换结果将被新的 ADC 转换结果覆盖。PDMA 将传输所选通道的最后数据到用户指定的目的地址。

5.16.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
ADC_BA = 0x400E_0000				
ADDR0	ADC_BA+0x00	R	A/D 数据寄存器 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D 数据寄存器 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D 数据寄存器 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D 数据寄存器 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D 数据寄存器 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D 数据寄存器 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D 数据寄存器 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D 数据寄存器 7	0x0000_0000
ADCR	ADC_BA+0x20	R/W	A/D 控制寄存器	0x0000_0000
ADCHER	ADC_BA+0x24	R/W	A/D 通道使能寄存器	0x0000_0000
ADCMPR0	ADC_BA+0x28	R/W	A/D 比较寄存器 0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	A/D 比较寄存器 1	0x0000_0000
ADSR	ADC_BA+0x30	R/W	A/D 状态寄存器	0x0000_0000
ADCALR	ADC_BA+0x34	R/W	A/D 校准寄存器	0x0000_0000
ADPDMA	ADC_BA+0x40	R	ADC PDMA 当前传输数据	0x0000_0000



5.16.6 寄存器描述

A/D 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
Reserved							
23	22	21	20	19	18	17	16
Reserved						VALID	OVERRUN
15	14	13	12	11	10	9	8
RSLT [15:8]							
7	6	5	4	3	2	1	0
RSLT[7:0]							

Bits	描述	
[31:18]	Reserved	保留
[17]	VALID	<p>有效标志 1 = RSLT[15:0] 中的数据有效 0 = RSLT[15:0] 中的数据无效 当相应的模拟通道转换完成后，该位被置位，读 ADDR 寄存器后，该位由硬件清除。 该位只读。</p>
[16]	OVERRUN	<p>Over Run 标志 1 = RSLT[15:0] 数据被覆盖 0 = RSLT[15:0] 数据为最新转换结果 如果在新的转换结果载入到该寄存器之前，已经在 RSLT[15:0] 的转换结果还没有被读取，则 OVERRUN 将被置 1，之前的转换结果也会丢失。在读 ADDR 寄存器之后，该位由硬件清除。 该位只读。</p>

[15:0]	RSLT	<p>A/D 转换结果</p> <p>该域包含了 ADC 的转换结果。</p> <p>当 DMOF 位 (ADCR[31]) 被置为 0, 12-位 ADC 转换结果是无符号格式的, 其结果将存放在 RSLT[11:0], 而 RSLT[15:12] 用 0 填充。</p> <p>当 DMOF 位 (ADCR[31]) 被置为 1, 12-位 ADC 转换结果是二进制的补码 (2'complement) 格式, 其结果将存放在 RSLT[11:0], 符号位存放在 RSLT[15:12]。</p>
--------	------	--

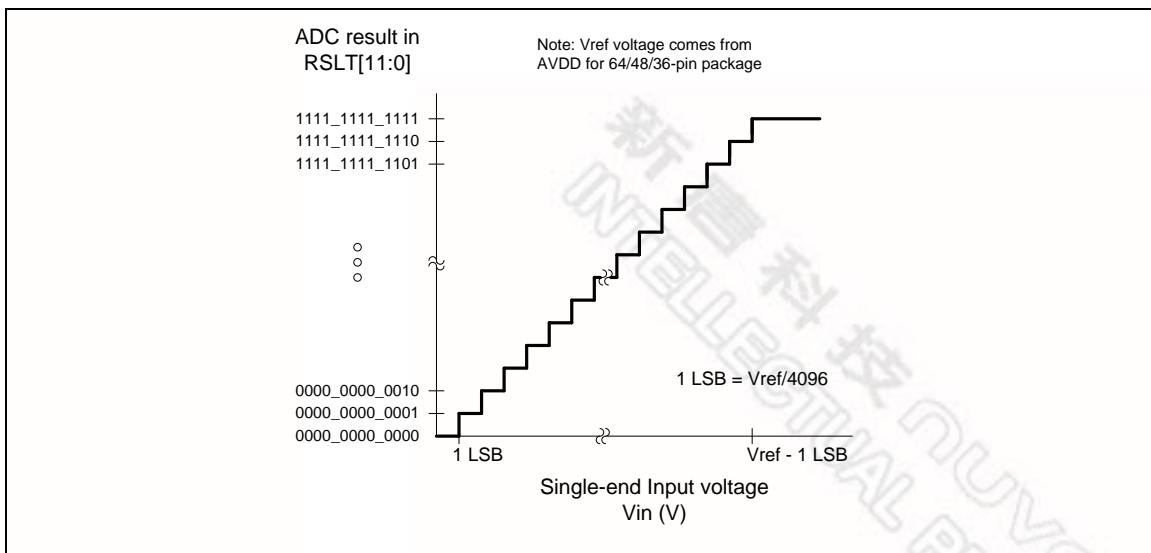


图 5-106 ADC 单端输入转换电压和转换结果图

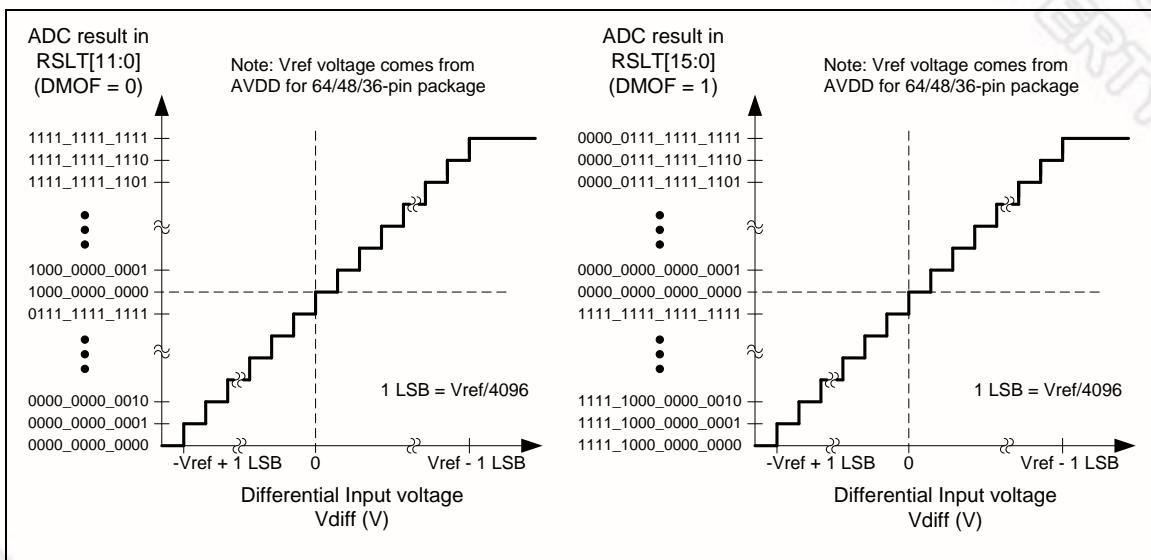


图 5-107 ADC 差分输入转换电压和转换结果图

A/D 控制寄存器 (ADCR)

寄存器	偏移量	R/W	描述	复位后的值
ADCR	ADC_BA+0x20	R/W	ADC 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
DMOF	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				ADST	DIFFEN	PTEN	TRGEN
7	6	5	4	3	2	1	0
TRGCOND		TRGS		ADMD		ADIE	ADEN

Bits	描述	
[31]	DMOF	A/D 差分输入模式的输出格式 1 = A/D转换结果存储在 ADDR _x 寄存器的 RSLT，格式为二进制补码格式 (2'complement format)。 0 = A/D 转换结果存储在 ADDR _x 寄存器的 RSLT，格式为无符号格式 (unsigned format)。
[30:12]	Reserved	保留
[11]	ADST	A/D 转换开始 1 = 转换开始 0 = 转换停止，A/D 转换器进入空闲状态 ADST 位置位有下列 2 种方式：软件设定和外部管脚 STADC 触发。单一模式和单周期模式下，在转换结束后，ADST 将被硬件自动清除。在连续扫描模式下，A/D 转换将一直进行直到软件向该位写 0 或芯片复位。

[10]	DIFFEN	差分输入模式使能 1 = 差分模拟输入模式 0 = 单端模拟输入模式															
		<table border="1"> <thead> <tr> <th rowspan="2">差分输入对通道</th><th colspan="2">ADC 模拟输入</th></tr> <tr> <th>V_{plus}</th><th>V_{minus}</th></tr> </thead> <tbody> <tr> <td>0</td><td>ADC0</td><td>ADC1</td></tr> <tr> <td>1</td><td>ADC2</td><td>ADC3</td></tr> <tr> <td>2</td><td>ADC4</td><td>ADC5</td></tr> <tr> <td>3</td><td>ADC6</td><td>ADC7</td></tr> </tbody> </table> <p>差分输入电压 (V_{diff}) = $V_{plus} - V_{minus}$, 其中 V_{plus} 是模拟输入; V_{minus} 是反向的模拟输入。 在差分输入模式下, 只需要在 ADCHER 使能两个相应通道中的偶数通道。转换结果将放置于相应的使能通道的寄存器里。</p>	差分输入对通道	ADC 模拟输入		V_{plus}	V_{minus}	0	ADC0	ADC1	1	ADC2	ADC3	2	ADC4	ADC5	3
差分输入对通道	ADC 模拟输入																
	V_{plus}	V_{minus}															
0	ADC0	ADC1															
1	ADC2	ADC3															
2	ADC4	ADC5															
3	ADC6	ADC7															
[9]	PTEN	PDMA 传输使能 1 = 使能 PDMA 传输 ADDR 0~7 中数据 0 = 禁用 PDMA 数据传输 当 A/D 转换完成后, 转换的结果载入到 ADDR 0~7, 软件可使能该位来产生 PDMA 数据传输请求。 当 PTEN=1, 软件必须设定 ADIE=0 来禁用中断。															
[8]	TRGEN	外部触发使能 通过外部 STADC 管脚使能或禁用 A/D 转换的触发。 1= 使能 0= 禁用 ADC 外部触发功能只在单周期扫描模式下支持。															
[7:6]	TRGCOND	外部触发条件 该 2 位决定外部管脚 STADC 触发状态为电平或边沿。该信号必须保持至少 8 PCLKs 的稳定状态用于电平触发, 4 PCLKs 的高和低状态用于边沿触发。 00 = 低电平 01 = 高电平 10 = 下降沿 11 = 上升沿															
[5:4]	TRGS	硬件触发源 00 = A/D 转换由外部 STADC 管脚触发开始。 其他 = 保留 改变 TRGS 之前, 软件必须禁用 TRGEN 和 ADST。 在硬件触发模式下, ADST 位由来自 STADC 的外部触发设定。															

[3:2]	ADMD	A/D 转换操作模式 00 = 单一转换 01 = 保留 10 = 单周期扫描 11 = 连续扫描 当改变操作模式时，软件将首先禁止 ADST 位。
[1]	ADIE	A/D 中断使能 1 = 使能 A/D 中断功能 0 = 禁用 A/D 中断功能 如果 ADIE 位被置为1，则 A/D 转换结束将产生中断请求。
[0]	ADEN	A/D 转换使能 1 = 使能 0 = 禁用 在开始 A/D 转换功能前，该位需置位。清除该位为 0 将禁用 A/D 转换模拟电路从而节省功耗。

A/D 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	A/D 通道使能	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						PRESEL[1:0]	
7	6	5	4	3	2	1	0
CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0

Bits	描述	
[31:10]	Reserved	保留
[9:8]	PRESEL	模拟输入通道 7 选择 00 = 外部模拟输入 01 = 内部带隙电压 10 = 内部温度传感器 11 = 保留 注： 当软件选择带隙电压作为 ADC 通道 7 的模拟输入源时，ADC 时钟速率必须被限制在低于 300 KHz。
[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 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 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 1 = 使能 0 = 禁用
[1]	CHEN1	模拟输入通道 1 使能 1 = 使能 0 = 禁用
[0]	CHEN0	模拟输入通道 0 使能 1 = 使能 0 = 禁用

A/D 比较寄存器 0/1 (ADCMR0/1)

寄存器	偏移量	R/W	描述	复位后的值
ADCMR0	ADC_BA+0x28	R/W	A/D 比较寄存器 0	0x0000_0000
ADCMR1	ADC_BA+0x2C	R/W	A/D 比较寄存器 1	0x0000_0000

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

Bits	描述	
[31:28]	Reserved	保留
[27:16]	CMPD	比较数值 此 12-位数据用于和指定通道的转换结果进行比较。 当 DMOF 位被置为 0, ADC 比较器比较 CMPD 与无符号格式的转换结果。CMPD 也必须填写无符号格式。 当 DMOF 位被置为 1, ADC 比较器比较 CMPD 与二进制补码格式 (2'complement) 的转换结果。CMPD 也必须填写二进制补码格式 (2'complement)。
[15:12]	Reserved	保留
[11:8]	CMPMATCNT	比较匹配计数 当指定 A/D 通道的模拟转换值和比较条件 CMPCOND[2] 相匹配, 内部计数器将加 1。 当内部计数器的值达到设定值 (CMPMATCNT +1) 时, 将置位 CMPFx 位。
[7:6]	Reserved	保留

[5:3]	CMPCH	比较通道选择 000 = 选择通道 0 转换结果进行比较 001 = 选择通道 1 转换结果进行比较 010 = 选择通道 2 转换结果进行比较 011 = 选择通道 3 转换结果进行比较 100 = 选择通道 4 转换结果进行比较 101 = 选择通道 5 转换结果进行比较 110 = 选择通道 6 转换结果进行比较 111 = 选择通道 7 转换结果进行比较
[2]	CMPCOND	比较条件 1 = 设备比较条件，即当 12- 位 A/D 转换结果大于或等于 12- 位 CMPD (ADCMPRx[27:16])，内部匹配计数器将加 1。 0 = 设备比较条件，即当 12-位 A/D 转换结果小于 12-位 CMPD (ADCMPRx[27:16])，内部匹配计数器将加 1。 注：当内部计数器达到 (CMPMATCNT +1)，CMPPFx 位将被置位。
[1]	CMPIE	比较中断使能 1 = 使能比较功能中断 0 = 禁用比较功能中断 如果比较功能使能，而且比较条件和 CMPCOND 及 CMPMATCNT 的设定匹配，则 CMPPF 位将被置位，同时，如果 CMPIE 为1，将产生比较中断请求。
[0]	CMPEN	比较使能 1 = 使能比较功能 0 = 禁用比较功能 当转换的数据载入到 ADDR 寄存器时，设置该位为1 使能 ADC 控制器去比较 CMPD[11:0] 和指定通道的转换结果。.

A/D 状态寄存器 (ADSR)

寄存器	偏移量	R/W	描述	复位后的值
ADSR	ADC_BA+0x30	R/W	ADC 状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
OVERRUN							
15	14	13	12	11	10	9	8
VALID							
7	6	5	4	3	2	1	0
Reserved	CHANNEL			BUSY	CMPF1	CMPF0	ADF

Bits	描述	
[31:24]	Reserved	保留
[23:16]	OVERRUN	Over Run 标志 该位是 ADDR _x 中 OVERRUN 位的镜像。 该位只读。
[15:8]	VALID	数据有效标志 该位是 ADDR _x 中 VALID 位的镜像。 该位只读。
[7]	Reserved	保留
[6:4]	CHANNEL	当前转换通道 当 BUSY = 1 时，该域反映当前转换通道。当 BUSY=0 时，它表示下一个要转换的通道号。 该位只读。
[3]	BUSY	BUSY/IDLE 1 = A/D 转换器正在忙着转换 0 = A/D 转换器空闲 该位是 ADCR 中 ADST 位的镜像。 该位只读。

[2]	CMPF1	比较标志 当所选择的通道的 A/D 转换结果和 ADCMPR1 的设定条件匹配时，该位置1。该位写1清除。 1 = ADDR 中的转换结果和 ADCMPR1 的设定值匹配 0 = ADDR 中的转换结果和 ADCMPR1 的设定值不匹配
[1]	CMPF0	比较标志 当所选择的通道的 A/D 转换结果和 ADCMPR0 的设定条件匹配时，该位置1。该位写1清除。 1 = ADDR 中的转换结果和 ADCMPR0 的设定值匹配 0 = ADDR 中的转换结果和 ADCMPR0 的设定值不匹配
[0]	ADF	A/D 转换结束标志 该状态标志指示 A/D 转换的结束。 ADF在如下两种条件下被置 1。 1. 当在单一模式下的 A/D 转换结束 2. 当在扫描模式下，所有指定的通道的 A/D 转换结束 该标志写 1 清除。

A/D 校准寄存器 (ADCALR)

寄存器	偏移量	R/W	描述	复位后的值
ADCALR	ADC_BA+0x34	R/W	A/D 校准寄存器	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						CALDONE	CALEN

Bits	描述	
[31:2]	Reserved	保留
[1]	CALDONE	<p>校准完成 1 = A/D 转换器的自校正完成。 0 = A/D 转换器没校正或校正正在进行中（如果 CALEN 位置位）。 当写 0 到 CALEN 位，CALDONE 位会立即由硬件清除。该位为只读位。</p>
[0]	CALEN	<p>自校正使能 1 = 使能自校正 0 = 禁用自校正 软件可置位该位使能 A/D 转换执行自校正功能。完成校正功能需要 127 个 ADC 时钟。 CALDONE 置位后该位需保持为高，清该位将禁止自身校准 功能。</p>

A/D PDMA 当前传输数据寄存器 (ADPDMA)

寄存器	偏移量	R/W	描述	复位后的值
ADPDMA	ADC_BA+0x40	R	A/D PDMA 当前传输数据寄存器	0x0000_0000

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

Bits	描述	
[31:12]	Reserved	保留
[11:0]	AD_PDMA	ADC PDMA 当前传输数据寄存器 当 PDMA 传输时，读该寄存器可以监测当前 PDMA 的传输数据。 此为只读寄存器。



5.17 模拟比较器 (CMP)

5.17.1 概述

NuMicro™ NUC100 系列包含 2 路模拟比较器。该比较器可应用于不同的配置。当正极输入大于负极输入时，比较器输出为逻辑 1，反之为逻辑 0。当比较器输出值改变时，每一路比较器都可通过配置产生中断。该模块系统框图见图 5-108。

5.17.2 特征

- 模拟输入电压范围：0~5.0 V
- 支持迟滞功能 (Hysteresis function)
- 2 路模拟比较器，其负极可选内部参考电压输入
- 2 路模拟比较器共享同一个中断向量

5.17.3 框图

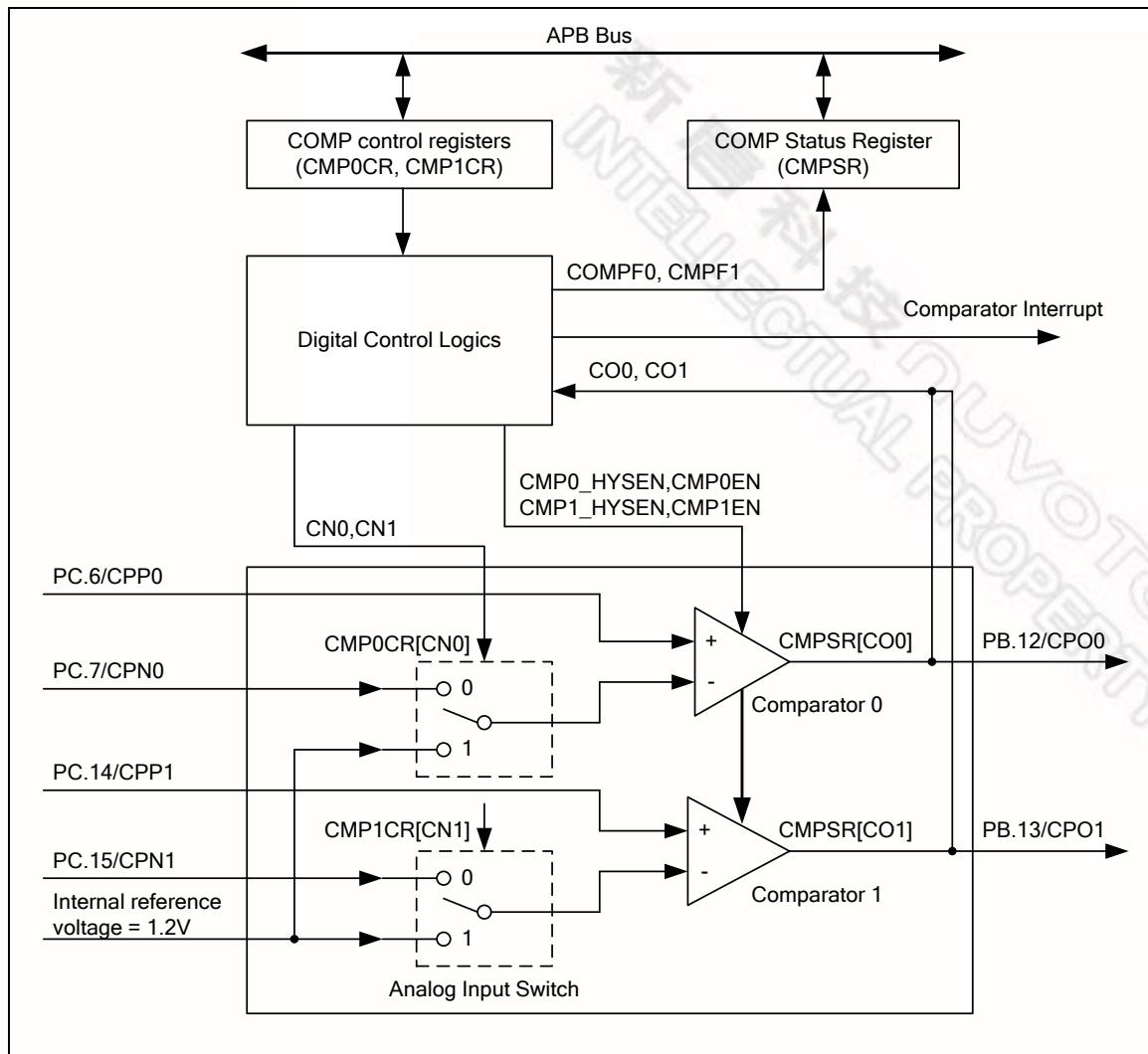


图 5-108 模拟比较器框图

5.17.4 功能描述

5.17.4.1 中断源

比较器的输出通过 PCLK 采样后，将结果写入寄存器 CMPSR 的 CO1 和 CO2。如果 CMP0CR/CMP1CR 的 CMP0IE/CMP1IE 置位，则模拟比较器的中断将被使能。当比较器的输出) 状态改变时，则会产生比较器中断请求，相应的比较器标志 CMPF0 或 CMPF1 将被置位。软件可写 1 到标志位清除该位。

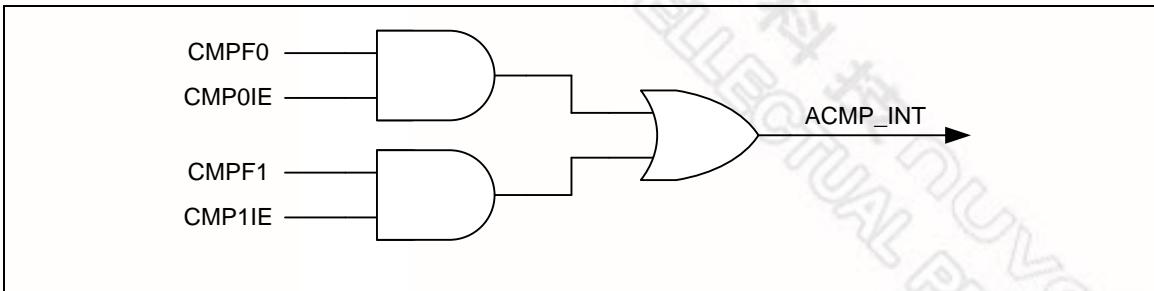


图 5-109 模拟比较器控制器中断源



5.17.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
CMP_BA = 0x400D_0000				
CMP0CR	CMP_BA+0x00	R/W	比较器 0 控制寄存器	0x0000_0000
CMP1CR	CMP_BA+0x04	R/W	比较器 1 控制寄存器	0x0000_0000
CMPSR	CMP_BA+0x08	R/W	比较器状态寄存器	0x0000_0000



5.17.6 寄存器描述

CMP0 控制寄存器 (CMP0CR)

寄存器	偏移量	R/W	描述	复位后的值
CMP0CR	CMP_BA+0x00	R/W	比较器 0 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			CN0	Reserved	CMP0_HYSEN	CMP0IE	CMP0EN

Bits	描述	
[31:5]	Reserved	保留
[4]	CN0	比较器 0 负极输入选择 1 = 选择内部带隙基准电压 ($V_{BG} = 1.26 V$) 作为比较器的负极输入源 0 = 选择 CPN0 管脚作为比较器的负极输入源
[3]	Reserved	保留
[2]	CMP0_HYSEN	比较器 0 迟滞 (Hysteresis) 使能 1 = 使能迟滞功能 (典型范围为 20mV)。 0 = 禁用迟滞功能 (Default)。
[1]	CMP0IE	比较器 0 中断使能 1 = 使能中断功能 0 = 禁用中断功能
[0]	CMP0EN	比较器 0 使能 1 = 使能 0 = 禁用 CMP0EN 置位后，比较器输出需等待 2 us 后稳定。

CMP1 控制寄存器 (CMP1CR)

寄存器	偏移量	R/W	描述	复位后的值
CMP1CR	CMP_BA+0x04	R/W	比较器 1 控制寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			CN1	Reserved	CMP1_HYSEN	CMP1IE	CMP1EN

Bits	描述	
[31:5]	Reserved	保留
[4]	CN1	比较器 1 负极输入选择 1 = 选择内部带隙基准电压 ($V_{BG} = 1.26 V$) 作为比较器的负极输入源 0 = 选择 CPN1 管脚作为比较器的负极输入源
[3]	Reserved	保留
[2]	CMP1_HYSEN	比较器 1 迟滞 (Hysteresis) 使能 1 = 使能迟滞功能 (典型范围为 20mV)。 0 = 禁用迟滞功能 (Default)。
[1]	CMP1IE	比较器 1 中断使能 1 = 使能中断功能 0 = 禁用中断功能
[0]	CMP1EN	比较器 1 使能 1 = 使能 0 = 禁用 CMP1EN 置位后，比较器输出需等待 2 us 后稳定。

CMP 状态寄存器 (CMPSR)

寄存器	偏移量	R/W	描述	复位后的值
CMPSR	CMP_BA+0x08	R/W	比较器状态寄存器	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CO1	CO0	CMPF1	CMPF0

Bits	描述	
[31:4]	Reserved	保留
[3]	CO1	比较器 1 输出 同步 APB 时钟从而允许软件访问。当比较器禁用 (CMP1EN = 0) 时清该位。
[2]	CO0	比较器 0 输出 同步 APB 时钟从而允许软件访问。当比较器禁用 (CMP0EN = 0) 时清该位。
[1]	CMPF1	比较器 1 标志 每当比较器 1 输出状态改变时，该位将被硬件置位。如果 CMP1IE 置位，则产生中断。 写 1 清该位为 0。
[0]	CMPF0	比较器 0 标志 每当比较器 0 输出状态改变时，该位将被硬件置位。如果 CMP0IE 置位，则产生中断。 写 1 清该位为 0。



5.18 PDMA 控制器 (PDMA)

5.18.1 概述

NuMicro™ NUC130/NUC140 包括一个外设直接存储器存取 (PDMA) 控制器，用于 APB 设备和存储器之间的数据传输。PDMA 有 9 路 DMA 通道（外设到存储器或存储器到外设或存储器到存储器）。每路 PDMA 通道 (PDMA CH0~CH8)，在外围 APB 设备和存储器之间有一个字大小的缓存作为传输缓存。

通过设定禁用 PDMA [PDMACEN] 位，软件可以停止 PDMA 的操作。通过软件轮询或者收到内部的 PDMA 中断，CPU 可以识别 PDMA 运作的完成。PDMA 控制器可增加源和目的地址或者也能将其固定。

注：部分 NuMicro™ NUC130/NUC140 仅有 1 PDMA 通道 (channel 0)。

5.18.2 特征

- 支持 9 DMA 通道。每个通道能支持一个单向传输
- AMBA AHB 主机/从机接口兼容，用于数据传输和寄存器读/写
- 支持源地址与目的地址增加模式或固定模式
- 硬件通道优先级。DMA 通道 0 拥有最高优先级，通道 8 拥有最低优先级

5.18.3 框图

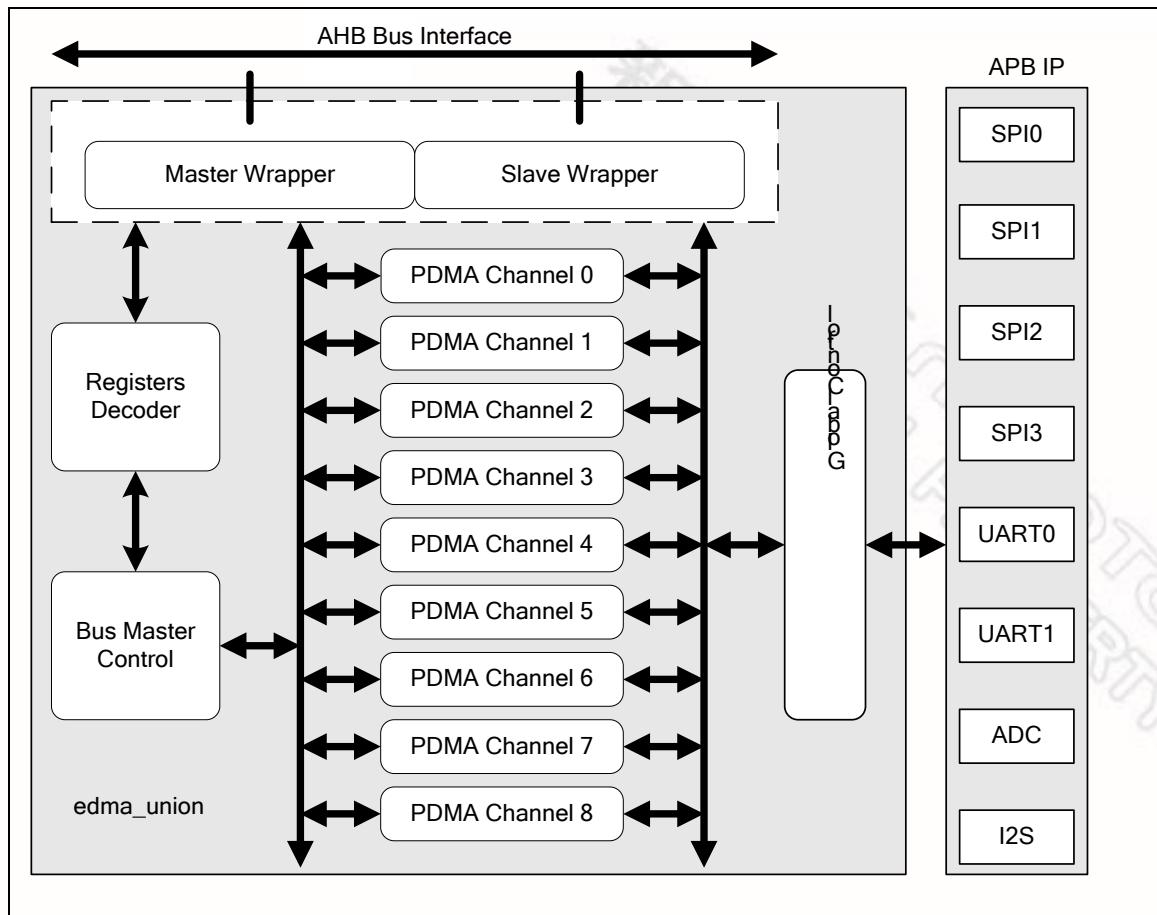


图 5-110 PDMA 控制器框图



5.18.4 功能描述

PDMA 控制器具有 9 路相关联 DMA，应用于外设-到-存储器、存储器-到-外设 或 存储器-到-存储器。每路 PDMA 通道，在外围 APB IP 和存储器之间有一个字的传输缓存。

CPU 通过软件轮询或接收到内部 PDMA 中断，可识别 PDMA 操作的完成。对于源和目的地址而言，PDMA 控制器具有两种模式：增加模式和固定模式。

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

软件必须使能 DMA 通道 PDMA [PDMACEN] 并且写有效的源地址到 PDMA_SARx 寄存器，写目的地址到 PDMA_DARx 寄存器，传输数量到 PDMA_BCRx 寄存器。然后触发 DMA_CSRx PDMA [TRIG_EN]。PDMA 将继续传输直到 PDMA_CBCRx 降为 0。如果在 PDMA 操作期间有错误发生，通道停止直到软件清除错误条件并设置 PDMA_CSRx [SW_RST] 复位 PDMA 通道，并设置 PDMA_CSRx [PDMACEN] 和 [TRIG_EN] 位重新开始。

在 PDMA（外设-到-存储器、存储器-到-外设）模式下，DMA 可以外设 APB IP (ex: UART, SPI, ADC....) 和 存储器间传输数据。

5.18.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
PDMA_BA_ch0 = 0x5000_8000	PDMA_BA_ch1 = 0x5000_8100		PDMA_BA_ch2 = 0x5000_8200	
PDMA_BA_ch3 = 0x5000_8300	PDMA_BA_ch4 = 0x5000_8400		PDMA_BA_ch5 = 0x5000_8500	
PDMA_BA_ch6 = 0x5000_8600	PDMA_BA_ch7 = 0x5000_8700		PDMA_BA_ch8 = 0x5000_8800	
PDMA_CSRx	PDMA_BA_chx+0x00	R/W	PDMA 控制寄存器	0x0000_0000
PDMA_SARx	PDMA_BA_chx+0x04	R/W	PDMA 源地址寄存器	0x0000_0000
PDMA_DARx	PDMA_BA_chx+0x08	R/W	PDMA 目的地址寄存器	0x0000_0000
PDMA_BCRx	PDMA_BA_chx+0x0C	R/W	PDMA 传输字节计数寄存器	0x0000_0000
PDMA_POINTx	PDMA_BA_chx+0x10	R	PDMA 内部缓存指针	0xXXXX_0000
PDMA_CSARx	PDMA_BA_chx+0x14	R	PDMA 当前源地址寄存器	0x0000_0000
PDMA_CDARx	PDMA_BA_chx+0x18	R	PDMA 当前目的地址寄存器	0x0000_0000
PDMA_CBCRx	PDMA_BA_chx+0x1C	R	PDMA 当前传输字节计数寄存器	0x0000_0000
PDMA_IERx	PDMA_BA_chx+0x20	R/W	PDMA 中断使能寄存器	0x0000_0001
PDMA_ISRx	PDMA_BA_chx+0x24	R/W	PDMA 中断状态寄存器	0x0000_0000
PDMA_SBUF0_cx	PDMA_BA_chx+0x80	R	PDMA 共享缓存 FIFO 0	0x0000_0000
PDMA_BA_GCR = 0x5000_8F00				
PDMA_GCRCR	PDMA_BA_GCR+0x00	R/W	PDMA 全局控制寄存器	0x0000_0000
PDMA_PDSSR0	PDMA_BA_GCR+0x04	R/W	PDMA 服务选择控制寄存器 0	0xFFFF_FFFF
PDMA_PDSSR1	PDMA_BA_GCR+0x08	R/W	PDMA 服务选择控制寄存器 1	0xFFFF_FFFF
PDMA_GCRISR	PDMA_BA_GCR+0x0C	R/W	PDMA 全局中断寄存器	0x0000_0000
PDMA_PDSSR2	PDMA_BA_GCR+0x10	R/W	PDMA 服务选择控制寄存器 2	0x0000_00FF



5.18.6 寄存器描述



PDMA 控制和状态寄存器 (PDMA_CSRx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_CSR0	PDMA_BA_ch0+0x00	R/W	PDMA 控制和状态寄存器 CH0	0x0000_0000
PDMA_CSR1	PDMA_BA_ch1+0x00	R/W	PDMA 控制和状态寄存器 CH1	0x0000_0000
PDMA_CSR2	PDMA_BA_ch2+0x00	R/W	PDMA 控制和状态寄存器 CH2	0x0000_0000
PDMA_CSR3	PDMA_BA_ch3+0x00	R/W	PDMA 控制和状态寄存器 CH3	0x0000_0000
PDMA_CSR4	PDMA_BA_ch4+0x00	R/W	PDMA 控制和状态寄存器 CH4	0x0000_0000
PDMA_CSR5	PDMA_BA_ch5+0x00	R/W	PDMA 控制和状态寄存器 CH5	0x0000_0000
PDMA_CSR6	PDMA_BA_ch6+0x00	R/W	PDMA 控制和状态寄存器 CH6	0x0000_0000
PDMA_CSR7	PDMA_BA_ch7+0x00	R/W	PDMA 控制和状态寄存器 CH7	0x0000_0000
PDMA_CSR8	PDMA_BA_ch8+0x00	R/W	PDMA 控制和状态寄存器 CH8	0x0000_0000

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

Bits	描述	
[31:24]	Reserved	保留
[23]	TRIG_EN	<p>TRIG_EN</p> <p>1 = 使能 PDMA 数据读或写传输 0 = 不起作用</p> <p>注：当 PDMA 传输完成时，该位将被自动清除。 如果总线错误出现，则所有的 DMA 传输将停止。软件必须复位所有的 PDMA 通道，然后再次触发。</p>
[22:21]	Reserved	保留

[20:19]	APB_TWS	外设传输宽度选择 00 = 每个 PDMA 操作传输一个字 (32-bit) 01 = 每个 PDMA 操作传输一个字节 (8-bit) 10 = 每个 PDMA 操作传输一个半字 (16-bit) 11 = 保留 注：当且仅当 MODE_SEL 是外设到存储器模式 (Peripheral-to-Memory) 或 存储器到外设模式 (Memory-to-Peripheral) 时，该域才有意义。
[18:8]	Reserved	保留
[7:6]	DAD_SEL	传输目的地址方向选择 00 = 传输目的地址持续增加 01 = 保留。 10 = 传输目的地址固定（该特性可被用于当数据从多个源地址到一个单独的目的地址传输的情况） 11 = 保留。
[5:4]	SAD_SEL	传输源地址方向选择 00 = 传输源地址持续增加 01 = 保留 10 = 传输源地址固定（该特性可被用于当数据从一个单独的源地址到多个目的地址传输的情况） 11 = 保留
[3:2]	MODE_SEL	PDMA 模式选择 00 = 存储器到存储器模式 (Memory-to-Memory). 01 = 外设到存储器模式 (Peripheral-to-Memory). 10 = 存储器到外设模式 (Memory-to-Peripheral).
[1]	SW_RST	软件引擎复位 0 = 写 0 到该位无效 1 = 写 1 到该位将复位内部状态机，指针和内部缓存。控制寄存器的内部不会被清除。 该位在几个时钟周期之后将自动清除。
[0]	PDMACEN	PDMA 通道使能 设置该位为 1 使能 PDMA 的操作。如果该位被清除，PDMA 将忽略所有的 PDMA 请求并强制总线主机进入 IDLE 状态。 注： SW_RST(PDMA_CSRx[1], x= 0~8) 将清除该位。

PDMA 传输源地址寄存器 (PDMA_SARx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_SAR0	PDMA_BA_ch0+0x04	R/W	PDMA 传输源地址寄存器 CH0	0x0000_0000
PDMA_SAR1	PDMA_BA_ch1+0x04	R/W	PDMA 传输源地址寄存器 CH1	0x0000_0000
PDMA_SAR2	PDMA_BA_ch2+0x04	R/W	PDMA 传输源地址寄存器 CH2	0x0000_0000
PDMA_SAR3	PDMA_BA_ch3+0x04	R/W	PDMA 传输源地址寄存器 CH3	0x0000_0000
PDMA_SAR4	PDMA_BA_ch4+0x04	R/W	PDMA 传输源地址寄存器 CH4	0x0000_0000
PDMA_SAR5	PDMA_BA_ch5+0x04	R/W	PDMA 传输源地址寄存器 CH5	0x0000_0000
PDMA_SAR6	PDMA_BA_ch6+0x04	R/W	PDMA 传输源地址寄存器 CH6	0x0000_0000
PDMA_SAR7	PDMA_BA_ch7+0x04	R/W	PDMA 传输源地址寄存器 CH7	0x0000_0000
PDMA_SAR8	PDMA_BA_ch8+0x04	R/W	PDMA 传输源地址寄存器 CH8	0x0000_0000

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

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

PDMA 传输目的地址寄存器 (PDMA_DARx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_DAR0	PDMA_BA_ch0+0x08	R/W	PDMA 传输目的地址寄存器 CH0	0x0000_0000
PDMA_DAR1	PDMA_BA_ch1+0x08	R/W	PDMA 传输目的地址寄存器 CH1	0x0000_0000
PDMA_DAR2	PDMA_BA_ch2+0x08	R/W	PDMA 传输目的地址寄存器 CH2	0x0000_0000
PDMA_DAR3	PDMA_BA_ch3+0x08	R/W	PDMA 传输目的地址寄存器 CH3	0x0000_0000
PDMA_DAR4	PDMA_BA_ch4+0x08	R/W	PDMA 传输目的地址寄存器 CH4	0x0000_0000
PDMA_DAR5	PDMA_BA_ch5+0x08	R/W	PDMA 传输目的地址寄存器 CH5	0x0000_0000
PDMA_DAR6	PDMA_BA_ch6+0x08	R/W	PDMA 传输目的地址寄存器 CH6	0x0000_0000
PDMA_DAR7	PDMA_BA_ch7+0x08	R/W	PDMA 传输目的地址寄存器 CH7	0x0000_0000
PDMA_DAR8	PDMA_BA_ch8+0x08	R/W	PDMA 传输目的地址寄存器 CH8	0x0000_0000

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

Bits	描述	
[31:0]	PDMA_DAR	PDMA 传输目的地址寄存器 该域表示一 32-位 PDMA 目的地址 注：目的地址必须字对齐

PDMA 传输字计数寄存器 (PDMA_BCRx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_BCR0	PDMA_BA_ch0+0x0C	R/W	PDMA 传输字节计数寄存器 CH0	0x0000_0000
PDMA_BCR1	PDMA_BA_ch1+0x0C	R/W	PDMA 传输字节计数寄存器 CH1	0x0000_0000
PDMA_BCR2	PDMA_BA_ch2+0x0C	R/W	PDMA 传输字节计数寄存器 CH2	0x0000_0000
PDMA_BCR3	PDMA_BA_ch3+0x0C	R/W	PDMA 传输字节计数寄存器 CH3	0x0000_0000
PDMA_BCR4	PDMA_BA_ch4+0x0C	R/W	PDMA 传输字节计数寄存器 CH4	0x0000_0000
PDMA_BCR5	PDMA_BA_ch5+0x0C	R/W	PDMA 传输字节计数寄存器 CH5	0x0000_0000
PDMA_BCR6	PDMA_BA_ch6+0x0C	R/W	PDMA 传输字节计数寄存器 CH6	0x0000_0000
PDMA_BCR7	PDMA_BA_ch7+0x0C	R/W	PDMA 传输字节计数寄存器 CH7	0x0000_0000
PDMA_BCR8	PDMA_BA_ch8+0x0C	R/W	PDMA 传输字节计数寄存器 CH8	0x0000_0000

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

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

PDMA 内部缓存指针寄存器(PDMA_POINTx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_POINT0	PDMA_BA_ch0+0x10	R	PDMA 内部缓存指针寄存器 CH0	0xXXXX_0000
PDMA_POINT1	PDMA_BA_ch1+0x10	R	PDMA 内部缓存指针寄存器 CH1	0xXXXX_0000
PDMA_POINT2	PDMA_BA_ch2+0x10	R	PDMA 内部缓存指针寄存器 CH2	0xXXXX_0000
PDMA_POINT3	PDMA_BA_ch3+0x10	R	PDMA 内部缓存指针寄存器 CH3	0xXXXX_0000
PDMA_POINT4	PDMA_BA_ch4+0x10	R	PDMA 内部缓存指针寄存器 CH4	0xXXXX_0000
PDMA_POINT5	PDMA_BA_ch5+0x10	R	PDMA 内部缓存指针寄存器 CH5	0xXXXX_0000
PDMA_POINT6	PDMA_BA_ch6+0x10	R	PDMA 内部缓存指针寄存器 CH6	0xXXXX_0000
PDMA_POINT7	PDMA_BA_ch7+0x10	R	PDMA 内部缓存指针寄存器 CH7	0xXXXX_0000
PDMA_POINT8	PDMA_BA_ch8+0x10	R	PDMA 内部缓存指针寄存器 CH8	0xXXXX_0000

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

Bits	描述	
[31:2]	Reserved	保留
[1:0]	PDMA_POINT	PDMA 内部缓存指针寄存器（只读） 该域表示内部缓存指针。

PDMA 当前源地址寄存器 (PDMA_CSARx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_CSAR0	PDMA_BA_ch0+0x14	R	PDMA 当前源地址寄存器 CH0	0x0000_0000
PDMA_CSAR1	PDMA_BA_ch1+0x14	R	PDMA 当前源地址寄存器 CH1	0x0000_0000
PDMA_CSAR2	PDMA_BA_ch2+0x14	R	PDMA 当前源地址寄存器 CH2	0x0000_0000
PDMA_CSAR3	PDMA_BA_ch3+0x14	R	PDMA 当前源地址寄存器 CH3	0x0000_0000
PDMA_CSAR4	PDMA_BA_ch4+0x14	R	PDMA 当前源地址寄存器 CH4	0x0000_0000
PDMA_CSAR5	PDMA_BA_ch5+0x14	R	PDMA 当前源地址寄存器 CH5	0x0000_0000
PDMA_CSAR6	PDMA_BA_ch6+0x14	R	PDMA 当前源地址寄存器 CH6	0x0000_0000
PDMA_CSAR7	PDMA_BA_ch7+0x14	R	PDMA 当前源地址寄存器 CH7	0x0000_0000
PDMA_CSAR8	PDMA_BA_ch8+0x14	R	PDMA 当前源地址寄存器 CH8	0x0000_0000

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

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

PDMA 当前目的地址寄存器 (PDMA_CDARx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_CDAR0	PDMA_BA_ch0+0x18	R	PDMA 当前目的地址寄存器 CH0	0x0000_0000
PDMA_CDAR1	PDMA_BA_ch1+0x18	R	PDMA 当前目的地址寄存器 CH1	0x0000_0000
PDMA_CDAR2	PDMA_BA_ch2+0x18	R	PDMA 当前目的地址寄存器 CH2	0x0000_0000
PDMA_CDAR3	PDMA_BA_ch3+0x18	R	PDMA 当前目的地址寄存器 CH3	0x0000_0000
PDMA_CDAR4	PDMA_BA_ch4+0x18	R	PDMA 当前目的地址寄存器 CH4	0x0000_0000
PDMA_CDAR5	PDMA_BA_ch5+0x18	R	PDMA 当前目的地址寄存器 CH5	0x0000_0000
PDMA_CDAR6	PDMA_BA_ch6+0x18	R	PDMA 当前目的地址寄存器 CH6	0x0000_0000
PDMA_CDAR7	PDMA_BA_ch7+0x18	R	PDMA 当前目的地址寄存器 CH7	0x0000_0000
PDMA_CDAR8	PDMA_BA_ch8+0x18	R	PDMA 当前目的地址寄存器 CH8	0x0000_0000

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

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

PDMA 当前字节计数寄存器 (PDMA_CBCRx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_CBCR0	PDMA_BA_ch0+0x1C	R	PDMA 当前字节计数寄存器 CH0	0x0000_0000
PDMA_CBCR1	PDMA_BA_ch1+0x1C	R	PDMA 当前字节计数寄存器 CH1	0x0000_0000
PDMA_CBCR2	PDMA_BA_ch2+0x1C	R	PDMA 当前字节计数寄存器 CH2	0x0000_0000
PDMA_CBCR3	PDMA_BA_ch3+0x1C	R	PDMA 当前字节计数寄存器 CH3	0x0000_0000
PDMA_CBCR4	PDMA_BA_ch4+0x1C	R	PDMA 当前字节计数寄存器 CH4	0x0000_0000
PDMA_CBCR5	PDMA_BA_ch5+0x1C	R	PDMA 当前字节计数寄存器 CH5	0x0000_0000
PDMA_CBCR6	PDMA_BA_ch6+0x1C	R	PDMA 当前字节计数寄存器 CH6	0x0000_0000
PDMA_CBCR7	PDMA_BA_ch7+0x1C	R	PDMA 当前字节计数寄存器 CH7	0x0000_0000
PDMA_CBCR8	PDMA_BA_ch8+0x1C	R	PDMA 当前字节计数寄存器 CH8	0x0000_0000

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

Bits	描述	
[31:16]	Reserved	保留
[15:0]	PDMA_CBCR	PDMA 当前字节计数寄存器 (只读) 该域指示 PDMA 当前剩下的字节计数。 注: SW_RST 将清除该寄存器值。

PDMA 中断使能控制寄存器 (PDMA_IERx)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_IER0	PDMA_BA_ch0+0x20	R/W	PDMA 中断使能控制寄存器 CH0	0x0000_0001
PDMA_IER1	PDMA_BA_ch1+0x20	R/W	PDMA 中断使能控制寄存器 CH1	0x0000_0001
PDMA_IER2	PDMA_BA_ch2+0x20	R/W	PDMA 中断使能控制寄存器 CH2	0x0000_0001
PDMA_IER3	PDMA_BA_ch3+0x20	R/W	PDMA 中断使能控制寄存器 CH3	0x0000_0001
PDMA_IER4	PDMA_BA_ch4+0x20	R/W	PDMA 中断使能控制寄存器 CH4	0x0000_0001
PDMA_IER5	PDMA_BA_ch5+0x20	R/W	PDMA 中断使能控制寄存器 CH5	0x0000_0001
PDMA_IER6	PDMA_BA_ch6+0x20	R/W	PDMA 中断使能控制寄存器 CH6	0x0000_0001
PDMA_IER7	PDMA_BA_ch7+0x20	R/W	PDMA 中断使能控制寄存器 CH7	0x0000_0001
PDMA_IER8	PDMA_BA_ch8+0x20	R/W	PDMA 中断使能控制寄存器 CH8	0x0000_0001

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

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

PDMA 中断状态寄存器 (PDMA_ISR_x)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_ISR0	PDMA_BA_ch0+0x24	R/W	PDMA 中断状态寄存器 CH0	0x0X0X_0000
PDMA_ISR1	PDMA_BA_ch1+0x24	R/W	PDMA 中断状态寄存器 CH1	0x0X0X_0000
PDMA_ISR2	PDMA_BA_ch2+0x24	R/W	PDMA 中断状态寄存器 CH2	0x0X0X_0000
PDMA_ISR3	PDMA_BA_ch3+0x24	R/W	PDMA 中断状态寄存器 CH3	0x0X0X_0000
PDMA_ISR4	PDMA_BA_ch4+0x24	R/W	PDMA 中断状态寄存器 CH4	0x0X0X_0000
PDMA_ISR5	PDMA_BA_ch5+0x24	R/W	PDMA 中断状态寄存器 CH5	0x0X0X_0000
PDMA_ISR6	PDMA_BA_ch6+0x24	R/W	PDMA 中断状态寄存器 CH6	0x0X0X_0000
PDMA_ISR7	PDMA_BA_ch7+0x24	R/W	PDMA 中断状态寄存器 CH7	0x0X0X_0000
PDMA_ISR8	PDMA_BA_ch8+0x24	R/W	PDMA 中断状态寄存器 CH8	0x0X0X_0000

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

Bits	描述	
[31:2]	Reserved	保留
[1]	BLKD_IF	Block 传输完成中断标志 该位指示 PDMA 已经完成了所有传输。 1 = 完成 0 = 还没完成 软件可写 1 清除该位为 0。
[0]	TABORT_IF	PDMA 读/写 目标中止中断标志 1 = 收到总线错误应答 0 = 没有收到总线错误应答 软件可写 1 清除该位为 0。

注：PDMA_ISR [TABORT_IF] 指示总线主机是否受到 ERROR 响应。如果总线主机收到了 ERROR 响应，则意味着目标中止发生了。PDMAC 将停止传输和响应该事件到软件，然后进入 IDLE 状态。当目标中止发生，软件必须复位 PDMA，再次

文件发布时间：2014年7月3日



传输那些数据。

PDMA 共享缓存 FIFO 0 (PDMA_SBUF0_cx)

Register	Offset	R/W	Description	Reset Value
PDMA_SBUF0_c0	PDMA_BA_ch0+0x080	R	PDMA 共享缓存 FIFO 0 寄存器 CH0	0x0000_0000
PDMA_SBUF0_c1	PDMA_BA_ch1+0x180	R	PDMA 共享缓存 FIFO 0 寄存器 CH1	0x0000_0000
PDMA_SBUF0_c2	PDMA_BA_ch2+0x280	R	PDMA 共享缓存 FIFO 0 寄存器 CH2	0x0000_0000
PDMA_SBUF0_c3	PDMA_BA_ch3+0x380	R	PDMA 共享缓存 FIFO 0 寄存器 CH3	0x0000_0000
PDMA_SBUF0_c4	PDMA_BA_ch4+0x480	R	PDMA 共享缓存 FIFO 0 寄存器 CH4	0x0000_0000
PDMA_SBUF0_c5	PDMA_BA_ch5+0x580	R	PDMA 共享缓存 FIFO 0 寄存器 CH5	0x0000_0000
PDMA_SBUF0_c6	PDMA_BA_ch6+0x680	R	PDMA 共享缓存 FIFO 0 寄存器 CH6	0x0000_0000
PDMA_SBUF0_c7	PDMA_BA_ch7+0x780	R	PDMA 共享缓存 FIFO 0 寄存器 CH7	0x0000_0000
PDMA_SBUF0_c8	PDMA_BA_ch8+0x880	R	PDMA 共享缓存 FIFO 0 寄存器 CH8	0x0000_0000

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

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

PDMA 全局控制寄存器 (PDMA_GCRCSR)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_GCRCSR	PDMA_BA_GCR+0x00	R/W	PDMA 全局控制寄存器	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
CLK7_EN	CLK6_EN	CLK5_EN	CLK4_EN	CLK3_EN	CLK2_EN	CLK1_EN	CLK0_EN
7	6	5	4	3	2	1	0
Reserved							

Bits	描述	
[31:17]	Reserved	保留
[16]	CLK8_EN	PDMA 控制器通道 8 时钟使能控制 0 = 禁用 1 = 使能
[15]	CLK7_EN	PDMA 控制器通道 7 时钟使能控制 0 = 禁用 1 = 使能
[14]	CLK6_EN	PDMA 控制器通道 6 时钟使能控制 0 = 禁用 1 = 使能
[13]	CLK5_EN	PDMA 控制器通道 5 时钟使能控制 0 = 禁用 1 = 使能
[12]	CLK4_EN	PDMA 控制器通道 4 时钟使能控制 0 = 禁用 1 = 使能
[11]	CLK3_EN	PDMA 控制器通道 3 时钟使能控制 0 = 禁用 1 = 使能



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

PDMA 服务选择控制寄存器 0 (PDMA_PDSSR0)

寄存器	地址	R/W	描述	复位后的值
PDMA_PDSSR0	PDMA_BA_GCR+0x04	R/W	PDMA 服务选择控制寄存器 0	0xFFFF_FFFF

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

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

[3:0]	SPI0_RXSEL	PDMA SPI0 RX 选择 该域定义了哪一个 PDMA 通道和片上外设 SPI0 RX 相连。软件可以通过 SPI0_RXSEL 配置 RX 通道。 4'b0000: CH0 4'b0001: CH1 4'b0010: CH2 4'b0011: CH3 4'b0100: CH4 4'b0101: CH5 4'b0110: CH6 4'b0111: CH7 4'b1000: CH8 其他: 保留 注: 例如: SPI0_RXSEL = 4'b0110, 则 SPI0_RX 和 PDMA_CH6 相连。
-------	-------------------	---

PDMA 服务选择控制寄存器 1 (PDMA_PDSSR1)

寄存器	地址	R/W	描述	复位后的值
PDMA_PDSSR1	PDMA_BA_GCR+0x08	R/W	PDMA 服务选择控制寄存器 1	0xFFFF_FFFF

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

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

[3:0]	UART0_RXSEL	该域定义了哪一个 PDMA 通道和片上外设 UART0 RX 相连。软件可以通过 UART0_RXSEL 配置 RX 通道。 4'b0000: CH0 4'b0001: CH1 4'b0010: CH2 4'b0011: CH3 4'b0100: CH4 4'b0101: CH5 4'b0110: CH6 4'b0111: CH7 4'b1000: CH8 其他: 保留 注: 例如: UART0_RXSEL = 4'b0110, 则 UART0_RX 和 PDMA_CH6 相连。
-------	--------------------	---

PDMA 全局中断状态寄存器 (PDMA_GCRISR)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_GCRISR	PDMA_BA_GCR+0x0C	R	PDMA 全局中断状态寄存器	0x0000_0000

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

Bits	描述	
[31]	INTR	中断管脚状态 该位是 PDMA 控制器的中断状态。 注：该位只读
[30:9]	Reserved	保留
[8]	INTR8	通道 8 中断管脚状态 该位是 PDMA 通道 8 的中断状态。 注：该位只读
[7]	INTR7	通道 7 中断管脚状态 该位是 PDMA 通道 7 的中断状态。 注：该位只读
[6]	INTR6	通道 6 中断管脚状态 该位是 PDMA 通道 6 的中断状态。 注：该位只读
[5]	INTR5	通道 5 中断管脚状态 该位是 PDMA 通道 5 的中断状态。 注：该位只读
[4]	INTR4	通道 4 中断管脚状态 该位是 PDMA 通道 4 的中断状态。 注：该位只读



[3]	INTR3	通道 3 中断管脚状态 该位是 PDMA 通道 3 的中断状态。 注：该位只读
[2]	INTR2	通道 2 中断管脚状态 该位是 PDMA 通道 2 的中断状态。 注：该位只读
[1]	INTR1	通道 1 中断管脚状态 该位是 PDMA 通道 1 的中断状态。 注：该位只读
[0]	INTR0	通道 0 中断管脚状态 该位是 PDMA 通道 0 的中断状态。 注：该位只读

PDMA 服务选择控制寄存器 2 (PDMA_PDSSR2)

寄存器	偏移量	R/W	描述	复位后的值
PDMA_PDSSR2	PDMA_BA_GCR+0x10	R/W	PDMA 服务选择控制寄存器 2	0x0000_00FF

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
I2S_TXSEL				I2S_RXSEL			

Bits	描述	
[31:8]	Reserved	保留
[7:4]	I2S_TXSEL	<p>PDMA I²S TX 选择</p> <p>该域定义了哪一个 PDMA 通道和片上外设 I²S TX 相连。软件可以通过 I2S_TXSEL 配置 TX 通道。该通道配置和 I2S_RXSEL 域相同，请参考 I2S_RXSEL 的说明。</p>
[3:0]	I2S_RXSEL	<p>PDMA I²S RX 选择</p> <p>该域定义了哪一个 PDMA 通道和片上外设 I²S RX 相连。软件可以通过 I2S_RXSEL 配置 RX 通道。</p> <p>4'b0000: CH0 4'b0001: CH1 4'b0010: CH2 4'b0011: CH3 4'b0100: CH4 4'b0101: CH5 4'b0110: CH6 4'b0111: CH7 4'b1000: CH8 其他: 保留</p> <p>注: 例如: I2S_RXSEL = 4'b0110, 则 I2S_RX 和 PDMA_CH6 相连。</p>



5.19 外部总线接口 (EBI)

5.19.1 概述

NuMicro™ NUC130/NUC140 LQFP-64 和 LQFP-100 封装配备了一个外部总线接口 (EBI)，以供外部设备使用。

为节省外部设备与芯片的连接线，EBI 支持地址总线与数据总线多路复用的模式，而且地址锁存使能 (ALE) 信号支持地址与数据周期的差别。

5.19.2 特征

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)
- 支持可变的外部总线基本时钟 (MCLK)
- 支持 8-位 或 16-位 数据宽度
- 支持可变的数据访问时间 (tACC)，地址锁存使能时间 (tALE) 和地址保持时间 (tAHD)
- 支持地址总线和数据总线多路复用以节省地址管脚
- 支持可配置的空闲周期以用于不同访问条件：写命令完成 (W2X)，连续读 (R2R)

5.19.3 框图

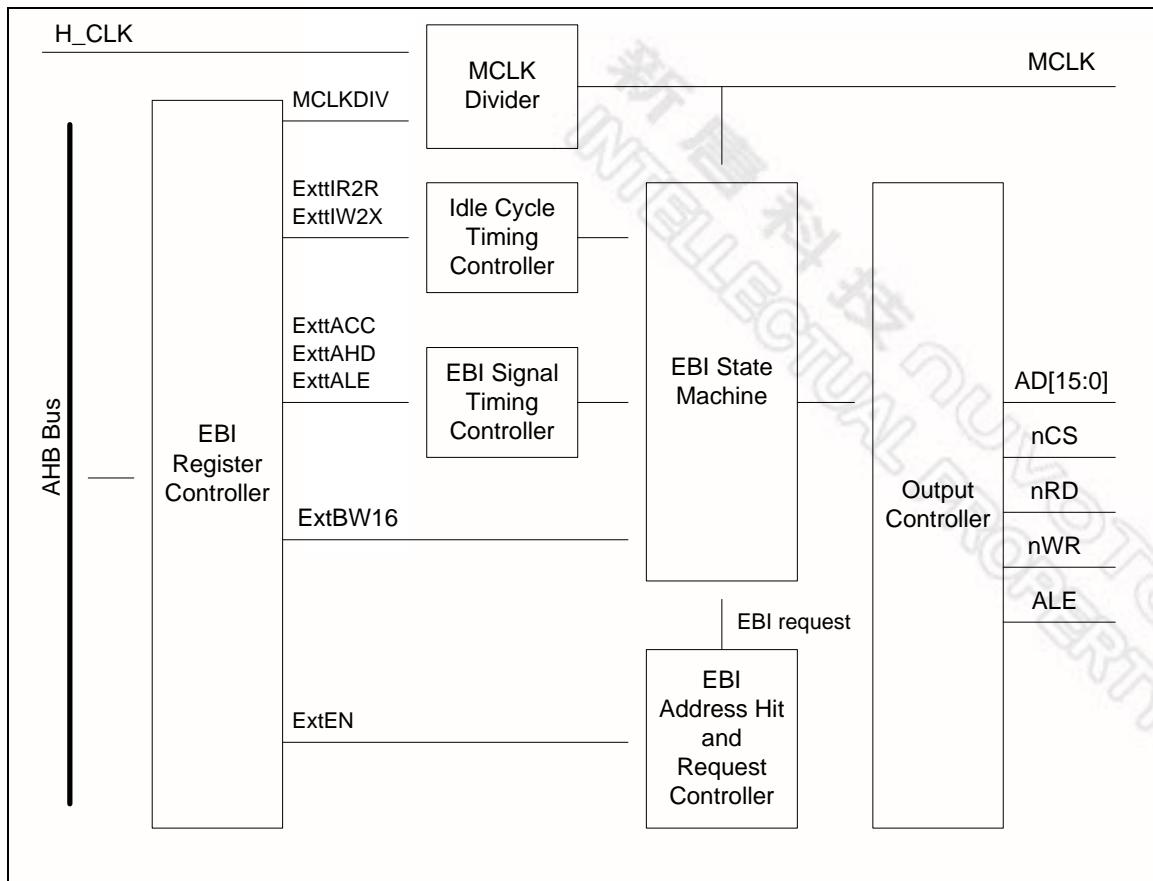


图 5-111 EBI 框图

5.19.4 功能描述

5.19.4.1 EBI 区域和地址命中

EBI 映射地址分布在 **0x6000_0000 ~ 0x6001_FFFF**，总共内存空间为 **128Kbyte**。当系统请求地址在 EBI 的内存空间内时，相应的 EBI 芯片选择信号有效，EBI 状态机工作。

对于 8-位设备 (64Kbyte)，EBI 同时映射该 64Kbyte 设备到 **0x6000_0000 ~ 0x6000_FFFF** 和 **0x6001_0000 ~ 0x6001_FFFF**。

5.19.4.2 EBI 数据宽度连接

EBI 支持具有多路地址总线和数据总线的设备。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单元锁存地址。这样，管脚 **ALE** 需要连接到锁存器上锁存地址值。管脚 **AD** 为锁存器的输入，锁存器的输出连接到外部设备的数据总线上。对于 16-位设备，**AD [15:0]** 由地址与 16-位数据共用。对于 8-位器件，仅 **AD [7:0]** 由地址与 8-位数据共用，**AD [15:8]** 作地址，直接与 8-位设备连接。

对于 8-位数据宽度，芯片系统地址**[15:0]** 作为设备地址**[15:0]**。对于 16-位数据宽度，芯片系统地址**[16:1]** 作为设备地址**[15:0]**，芯片系统地址位 **[0]** 无作用。

EBI 位宽	系统地址 (AHBADR)	EBI 地址 (AD)
8-bit	AHBADR[15:0]	AD[15:0]
16-bit	AHBADR[16:1]	AD[15:0]

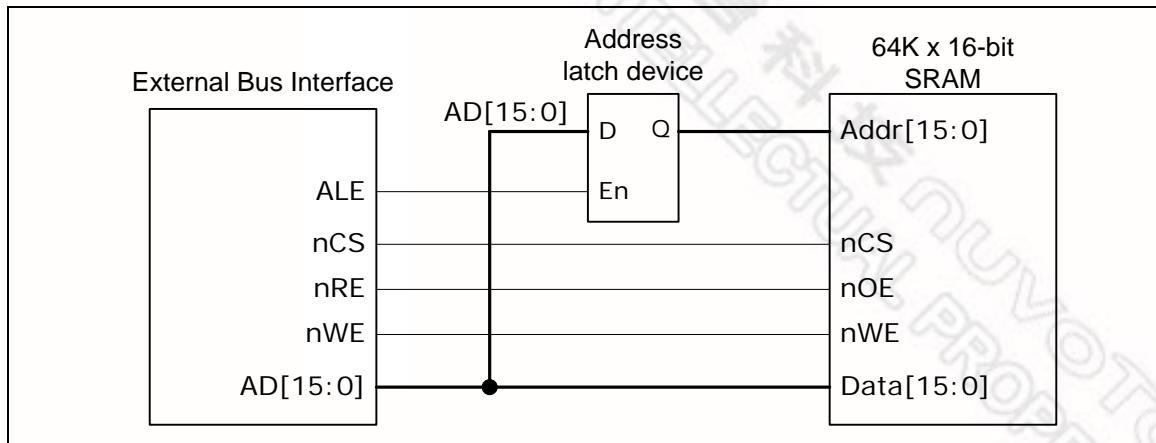


图 5-112 16-位 EBI 数据宽度与 16-位设备的连接

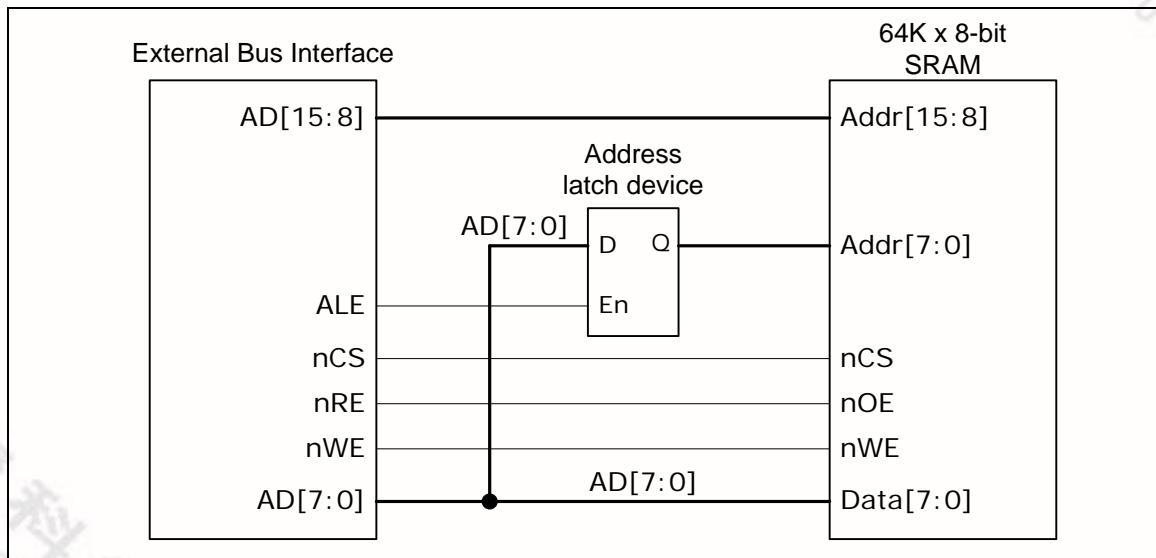


图 5-113 8-位 EBI 数据宽度和 8-位设备的连接

当系统访问数据宽度大于 EBI 的数据宽度，EBI 控制器将访问一次以上以完成操作。例如，如果系统通过 EBI 设备请求 32-位数据，当 EBI 为8-位数据宽度时，EBI 控制器将访问 4 次完成操作。

5.19.4.3 EBI 操作控制

MCLK 控制

当 EBI 工作时，芯片内所有的 EBI 信号将通过 MCLK 进行同步。当芯片以较低工作频率连接到外部设备，MCLK 可以通过设定寄存器 EBICON 的 MCLKDIV 最多分频到 HCLK/32。因此，芯片可以适用于宽频率范围的 EBI 设备。如果 MCLK 设置为 HCLK/1，则 EBI 信号与 MCLK 的正边沿同步，否则与 MCLK 的负边沿同步。

操作与访问时序控制

开始访问时，片选 (nCS) 置低并等待一个 MCLK 时间用于地址建立时间 (tASU) 以使地址稳定。地址稳定后 ALE 置高并保持一段时间 (tALE) 用于地址锁存。地址锁存后，ALE 置低并等待一个 MCLK 的周期用于锁存保持时间 (tLHD) 和另一个插入到地址保持时间之后的 MCLK 的时间 (tA2D) 用于总线转换（地址到数据）。然后当读访问时 nRD 置低或写访问时 nWR 置低。在保持访问时间(tACC) 后 nRD 或nWR 置高用于读输出稳定或写完成。之后，EBI 信号保持数据访问时间 (tAHD)，片选置高，地址由当前访问控制释放。

EBI 控制器提供了灵活的时序控制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，通过设置寄存器 EXTIME 的 ExttAHD, tAHD 可以在 1~8 MCLK 周期调节，通过设置寄存器 EXTIME 的 ExttACC, tACC可以在 1~32 MCLK 周期调节，通过设置寄存器 EBICON 的 tALE, tALE可以在 1~8 MCLK 周期调节。

参数	值	单元	描述
tASU	1	MCLK	地址锁存建立时间
tALE	1 ~ 8	MCLK	ALE 高时间。由 EBICON 的 ExttALE 控制
tLHD	1	MCLK	地址锁存保持时间
tA2D	1	MCLK	地址到数据的延时（总线转换时间）
tACC	1 ~ 32	MCLK	数据访问时间，由 EXTIME 的ExttACC 控制
tAHD	1 ~ 8	MCLK	数据访问保持时间，由 EXTIME 的 ExttAHD 控制
IDLE	0 ~ 15	MCLK	空闲周期，由 EXTIME 的 ExtIR2R 和 ExtIW2X 控制

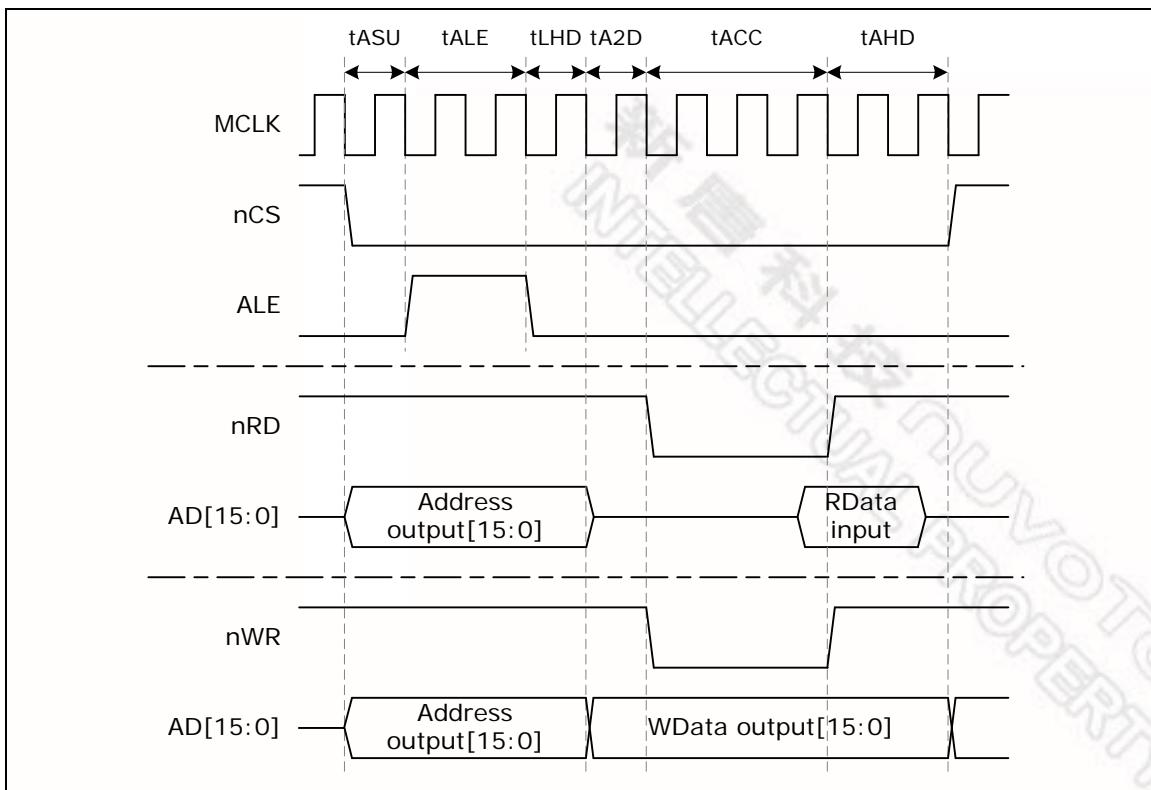


图 5-114 16-位数据宽度的时序控制波形

图 5-114 是一个设置 16-位数据宽度的例子。在该例中，AD 总线用作地址 [15:0] 和数据 [15:0]。当 ALE 置高，AD 为地址输出。在地址锁存后，ALE 置低并且在读取访问操作时，AD 总线转换成高阻以等待设备输出数据，或用于写数据输出。

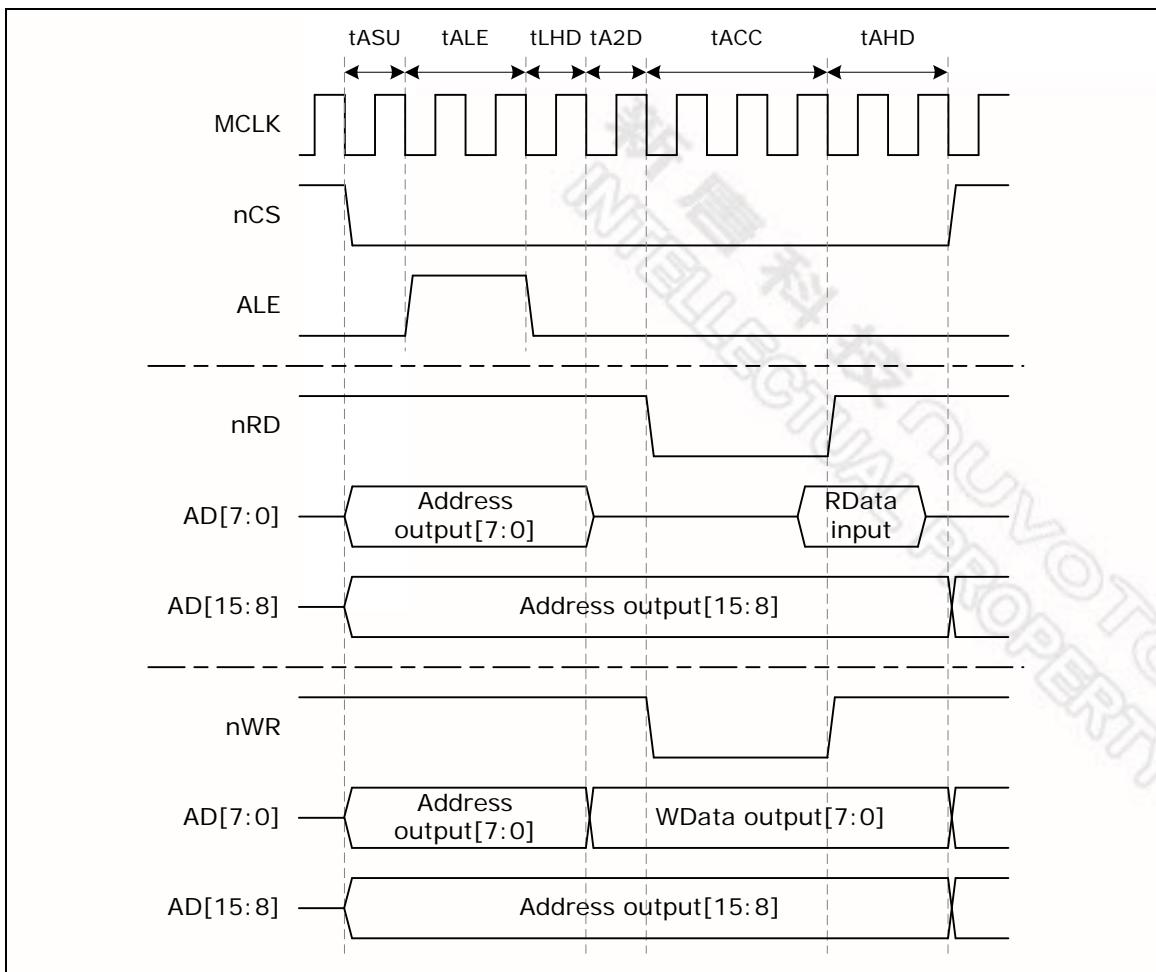


图 5-115 8-位数据宽度时序控制波形

图 5-115 是一个设置 8-位数据宽度的例子。8-位和 16-位数据宽度的不同之处在于 AD[15:8]。在8-位数据宽度的设置中，AD[15:8] 总为地址[15:8]输出，因此外部锁存只需要 8-位宽度。

插入空闲周期

当 EBI 连续访问时，如果器件访问时间比系统操作慢得多，可能会出现总线冲突。EBI 控制器支持额外空闲周期以解决该问题。在空闲周期，所有 EBI 的控制信号无效。图 5-116 为空闲周期。

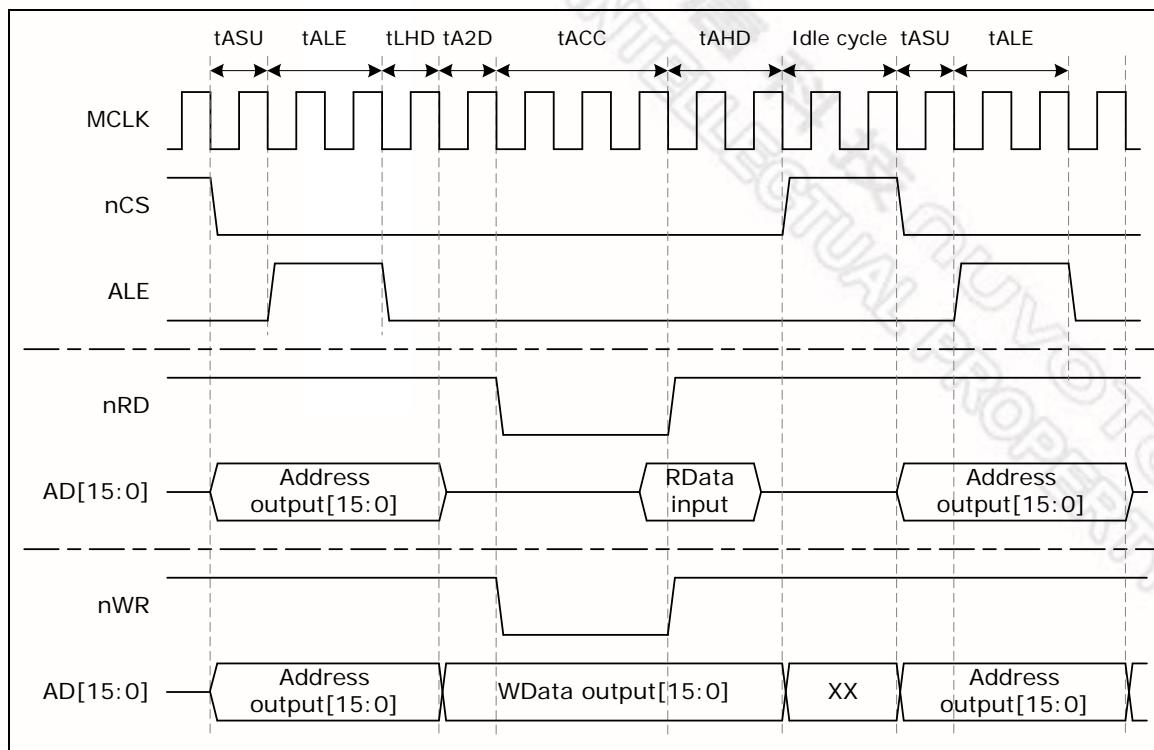


图 5-116 插入空闲周期的时序控制波形

有 2 个条件，EBI 可通过时序控制插入空闲周期：

1. 写访问后
2. 读访问后及下一个读访问之前

通过设置寄存器 EXTIME 的 ExtIw2X 和 ExtIr2R，空闲周期可被设定在 0~15 MCLK。



5.19.5 寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
EBI_BA = 0x5001_0000				
EBICON	EBI_BA+0x00	R/W	外部总线接口通用控制寄存器	0x0000_0000
EXTIME	EBI_BA+0x04	R/W	外部总线接口时序控制寄存器	0x0000_0000

5.19.6 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	外部总线接口通用控制寄存器	0x0000_0000

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

Bits	描述																	
[31:19]	Reserved	保留																
[18:16]	ExttALE	ALE 的扩展时间 通过 ExttALE 控制 ALE 宽度 (tALE) 来锁存地址 $tALE = (ExttALE+1)*MCLK$																
[15:11]	Reserved	保留																
[10:8]	MCLKDIV	外部输出时钟分频器 EBI 输出时钟频率由 MCLKDIV 控制, 见下表: <table border="1" style="margin-left: 20px;"> <tr> <th>MCLKDIV</th> <th>Output clock (MCLK)</th> </tr> <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>11X</td> <td>default</td> </tr> </table> 注: 输出时钟的默认值是 HCLK/1	MCLKDIV	Output clock (MCLK)	000	HCLK/1	001	HCLK/2	010	HCLK/4	011	HCLK/8	100	HCLK/16	101	HCLK/32	11X	default
MCLKDIV	Output clock (MCLK)																	
000	HCLK/1																	
001	HCLK/2																	
010	HCLK/4																	
011	HCLK/8																	
100	HCLK/16																	
101	HCLK/32																	
11X	default																	
[7:2]	Reserved	保留																
[1]	ExtBW16	EBI 数据宽度 16-位 该位定义数据总线是 8-位还是 16-位																



		1 = EBI 数据宽度是 16-位 0 = EBI 数据宽度是 8-位
[0]	ExtEN	EBI 使能 该位 EBI 的功能使能位 1 = 使能 EBI 功能 0 = 禁用 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
Reserved				ExtIR2R			
23	22	21	20	19	18	17	16
Reversed							
15	14	13	12	11	10	9	8
ExtIW2X				Reversed	ExttAHD		
7	6	5	4	3	2	1	0
ExttACC				Reversed			

Bits	描述	
[31:28]	Reserved	保留
[27:24]	ExtIR2R	读-读 之间的空闲状态周期 当读动作完成，下一个动作也是读时，插入空闲状态，nCS 返回高（如果 ExtIR2R 非0）。 空闲状态周期 = (ExtIR2R * MCLK)
[23:16]	Reserved	保留
[15:12]	ExtIW2X	写之后的空闲状态周期 当写动作完成，插入空闲状态，nCS 返回高（如果 ExtIW2X 非0）。 空闲状态周期 = (ExtIW2X * MCLK)
[11]	Reserved	保留
[10:8]	ExttAHD	EBI 数据访问保持时间 ExttAHD 定义数据访问保持时间 (tAHD). tAHD = (ExttAHD +1) * MCLK
[7:3]	ExttACC	EBI 数据访问时间 ExttACC 定义数据访问时间 (tACC). tACC = (ExttACC +1) * MCLK
[2:0]	Reserved	保留



6 FLASH 内存控制器 (FMC)

6.1 概述

NuMicro™ NUC100 系列配置了128/64/32K 字节的片上 Flash，用于应用程序内存 (APROM)，用户可通过 ISP 更新该部分。当芯片焊接在 PCB 上时，在系统编程 (ISP) 功能允许用户更新程序内存。芯片上电后，Cortex-M0 CPU 依据 Config0 中的启动选择 (CBS) 从 APROM 或 LDROM 中读取代码。此外，NuMicro™ NUC100 系列还提供额外的数据 Flash 用于用户在芯片断电之前存储一些应用所需的数据。对于 128K 字节的 APROM 设备，数据 flash 与原始的 128K 程序内存共享，其起始地址是可配置的，由用户在 Config1 中定义。对于 64K/32K 字节的 APROM 设备，数据 flash 固定为 4K。

6.2 特征

- 零等待状态，可达50 MHz的连续的地址读访问
- 128/64/32KB 应用程序内存 (APROM)
- 4KB 在系统编程 (ISP) 加载程序内存 (LDROM)
- 可配置或固定 4KB 的数据 flash，带有 512 字节页擦除单元
- 可配置 128K APROM 设备的数据 flash 起始地址
- 在系统编程 (ISP) 用于更新片上 Flash

6.3 框图

Flash 内存控制器包括 AHB 从接口，ISP 控制逻辑，写接口和 flash 宏接口时序控制逻辑。Flash 内存控制器的框图见下图：

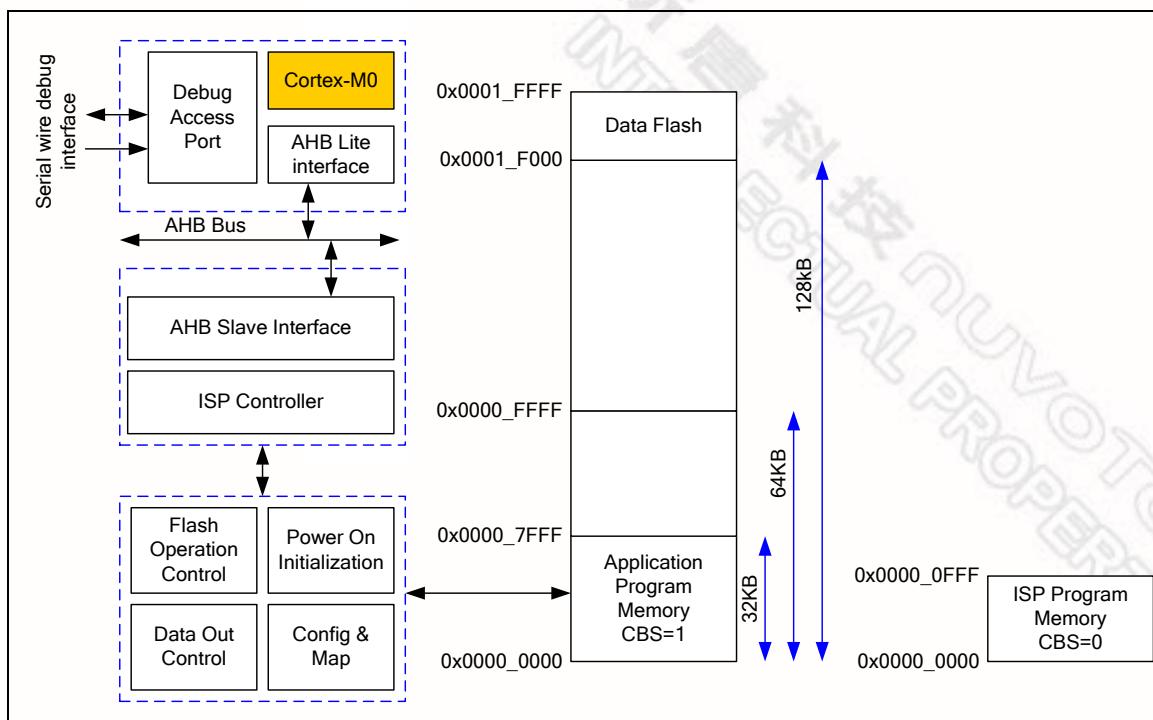


图 6-1 Flash 内存控制器框图

6.4 Flash 内存结构

NuMicro™ NUC100 系列 FLASH 内存由程序内存 (128/64/32KB), 数据 FLASH, ISP 载入程序内存, 及用户配置区域构成。用户配置区域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, 数据 FLASH 基地址等。它就像通电设置中的熔丝。在芯片上电过程中, 用户配置区域被从 FLASH 内存中载入到相应的控制寄存器。在芯片上 PCB 之前, 用户可以用烧写器根据应用需求来设置这些位。数据 FLASH 的开始地址和大小可由用户根据 128KB 的APROM的实际应用来定义。对于 64/32KB 的 APROM, 其大小为4KB, 开始地址固定为0x0001_F000。

区块名称	大小	开始地址	结束地址
AP-ROM	32/64/(128-0.5*N) KB	0x0000_0000	0x0000_7FFF (32KB) 0x0000_FFFF (64KB) DFBADR-1 (128KB if DFEN=0)
保留作将来之用	896KB	0x0002_0000	0x000F_FFFF
数据 Flash	4/4/0.5*N KB	0x0001_F000 DFBADR	0x0001_FFFF
LD-ROM	4 KB	0x0010_0000	0x0010_0FFF
用户配置	2 words	0x0030_0000	0x0030_0004

表 6-1 内存地址映射

Flash 内存组织结构如下所示：

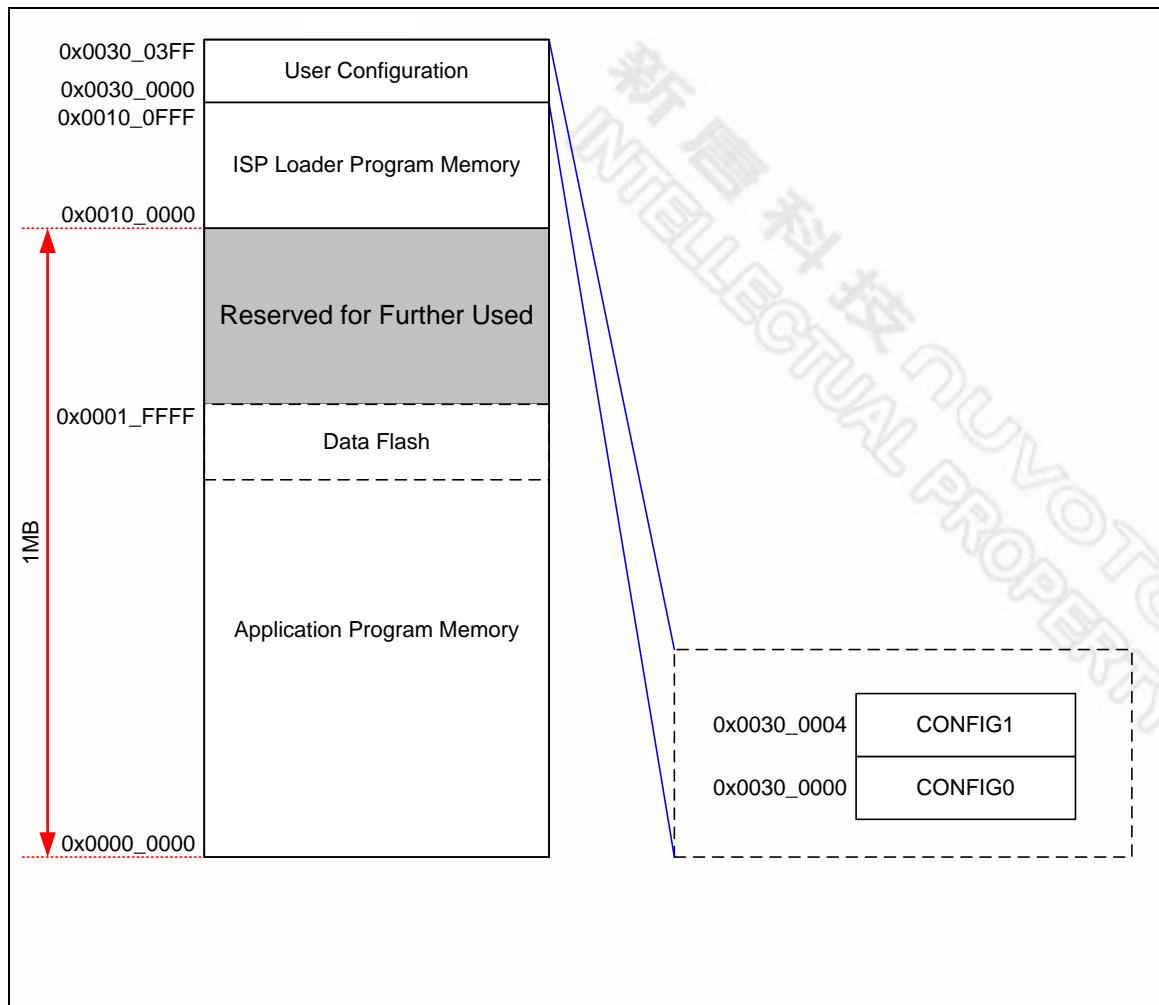


图 6-2 Flash 内存组织结构

6.5 启动选择

NuMicro™ NUC100 系列提供在系统编程 (ISP) 特征，允许用户直接更新 PCB 板上芯片中的程序。专用的 4KB 程序内存用于存储 ISP 固件。用户可以通过设置 Config0 中的 CBS 位来选择从 APROM 还是从 LDROM 取指令来开始程序。

6.6 数据 Flash

NuMicro™ NUC100 系列为用户提供数据 FLASH 用于存储数据，通过 ISP 过程读/写。每个擦除单元的大小是 512 字节。当要改变一个字节时，需要事先将所有的128个字拷贝到其他页或 SRAM 中。对于 128KB FLASH 器件，数据 FLASH 与应用程序共享 128KB 的内存，若 Config0 的 DFEN 位使能，数据 FLASH 的基地址由 DFBADR 定义，应用程序内存的大小为 $(128-0.5^*N)KB$ ，数据 FLASH 的大小为。对于 64/32KB APROM 器件，数据 FLASH 大小为 4KB，开始地址为 0x0001_F000。

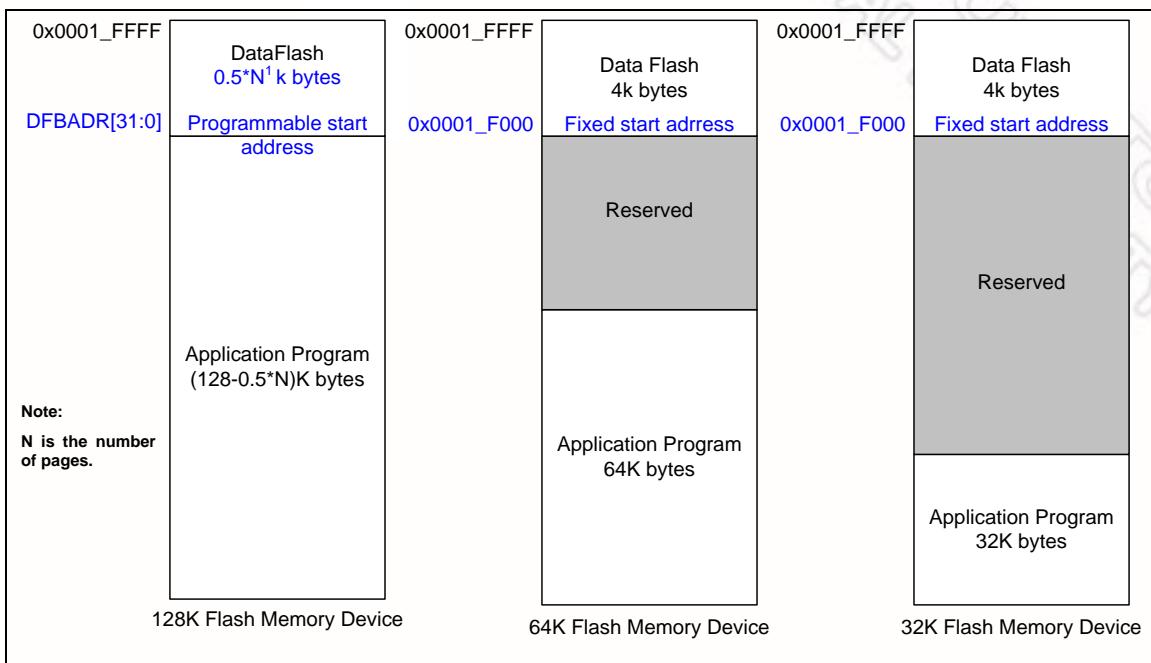


图 6-3 Flash 内存结构



6.7 用户配置

Config0 (Address = 0x0030_0000)

31	30	29	28	27	26	25	24
Reserved			CKF	Reserved	CFOSC		
23	22	21	20	19	18	17	16
CBODEN	CBOV1	CBOV0	CBORST	Reserved			
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CBS	Reserved				LOCK	DFEN	

Bits	描述															
[31:29]	Reserved	保留														
[28]	CKF	XT1 时钟过滤使能 0 = 禁用 XT1 时钟过滤 1 = 使能 XT1 时钟过滤														
[27]	Reserved	保留														
[26:24]	CFOSC	复位后, CPU 时钟源选择 <table border="1" style="margin-left: 20px;"> <tr><td>FOSC[2:0]</td><td>时钟源</td></tr> <tr><td>000</td><td>外部 4~24 MHz 高速晶振时钟</td></tr> <tr><td>111</td><td>内部 22.1184 MHz 高速振荡器时钟</td></tr> <tr><td>其他</td><td>保留</td></tr> </table> <p>任何复位发生后, CFOSC 的值将被加载到系统寄存器 CLKSEL0.HCLK_S[2:0]。</p>	FOSC[2:0]	时钟源	000	外部 4~24 MHz 高速晶振时钟	111	内部 22.1184 MHz 高速振荡器时钟	其他	保留						
FOSC[2:0]	时钟源															
000	外部 4~24 MHz 高速晶振时钟															
111	内部 22.1184 MHz 高速振荡器时钟															
其他	保留															
[23]	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]	欠压电压选择 <table border="1" style="margin-left: 20px;"> <tr><td>CBOV1</td><td>CBOV0</td><td>Brown-Out voltage</td></tr> <tr><td>1</td><td>1</td><td>4.5 V</td></tr> <tr><td>1</td><td>0</td><td>3.8 V</td></tr> <tr><td>0</td><td>1</td><td>2.7 V</td></tr> <tr><td>0</td><td>0</td><td>2.2 V</td></tr> </table>	CBOV1	CBOV0	Brown-Out voltage	1	1	4.5 V	1	0	3.8 V	0	1	2.7 V	0	0	2.2 V
CBOV1	CBOV0	Brown-Out voltage														
1	1	4.5 V														
1	0	3.8 V														
0	1	2.7 V														
0	0	2.2 V														

[20]	CBORST	欠压复位使能 0 = 上电后，使能欠压复位 1 = 上电后，禁用欠压复位
[19:8]	Reserved	保留
[7]	CBS	芯片启动选项 0 = 芯片从 LDROM 启动 1 = 芯片从 APROM 启动
[6:2]	Reserved	保留
[1]	LOCK	安全加密 0 = 加密 Flash 数据 1 = 解除 Flash 数据加密 当 flash 数据被加密，只有设备 ID, Config0 和 Config1 可被烧写器和 ICP 通过串口调试接口读取。其他数据锁定为 0xFFFFFFFF。无论数据是否锁定，ISP 都可以读取数据。
[0]	DFEN	数据 Flash 使能 (该位只支持 128KB APROM 器件) 0 = 使能数据 flash 1 = 禁用数据 flash

**Config1 (Address = 0x0030_0004)**

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

Bits	描述	
[31:20]	Reserved	保留（强行给这些保留位写入 0x00）
[19:0]	DFBADR	数据 Flash 基地址 （该寄存器仅支持 128KB APROM 设备） 对于 128KB APROM 设备，其数据 flash 基地址由用户定义。因为片上 flash 擦除单元为 512 字节，所以强制保持 bit 8-0 为0。这种配置仅对 128KB 的flash 设备有效。

6.8 在系统编程 (ISP)

程序内存和数据 FLASH 支持硬件编程和在系统编程 (ISP)。硬件编程模式采用 gang 烧写器，以方便量产阶段降低编程成本和时间。但是若产品还在开发阶段或终端用户需要升级固件时，硬件编程模式不是很方便，ISP 模式能更好地适用于这种情况。NuMicro™ NUC100 系列支持 ISP 模式，即通过软件控制来对设备重新编程。而且，这也使得更新固件得以广泛应用。

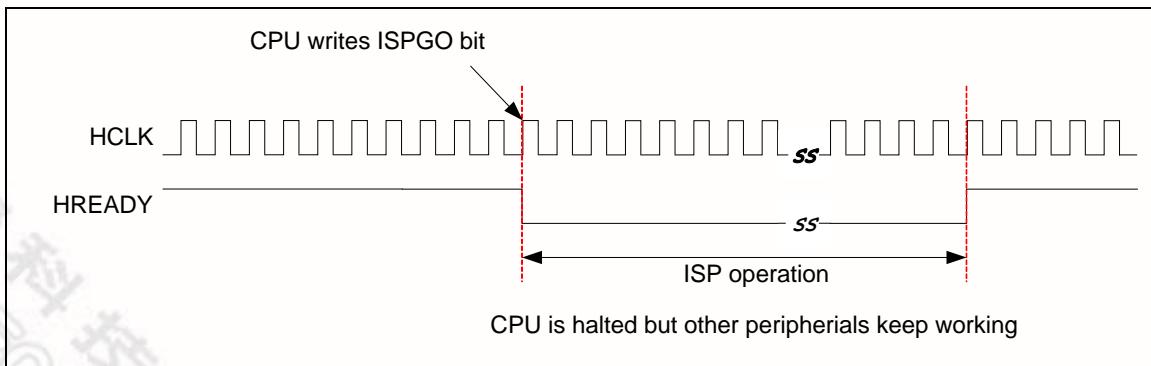
ISP 可以在没有将微控器从系统中取下来的情况下执行编程。各种接口使得 LDROM 更容易更新程序代码。最常用的方法是通过 UART 和在 LDROM 中的固件执行 ISP，一般来说，PC 都是通过串口传输新的 APROM 代码。LDROM 固件接收后，通过 ISP 命令重新对 APROM 编程。Nuvoton 提供用于 NuMicro™ NUC100 系列的 ISP 固件和 PC 应用程序。这使得用户可以通过 Nuvoton ISP 工具非常容易的执行 ISP。

6.8.1 ISP 程序

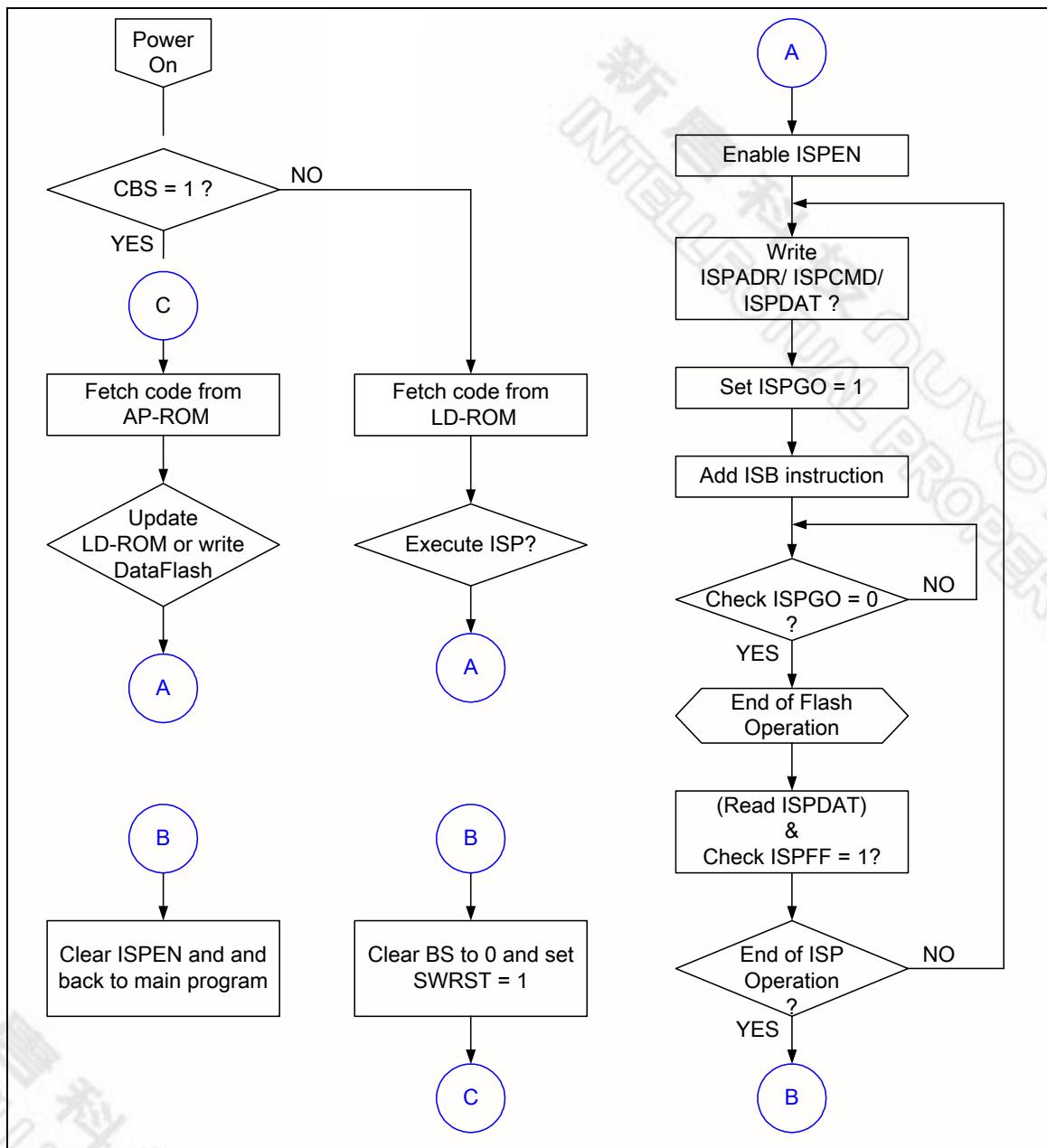
NuMicro™ NUC100 系列支持根据初始定义的用户配置位 (CBS) 从 APROM 或 LDROM 启动。如果用户想更新 APROM 中的应用程序，可以写 $BS=1$ ，并开始软件复位使芯片从 LDROM 启动。开始 ISP 功能的第一步是向 ISPPEN 位写 1。在写 ISPCON 寄存器之前，S/W 需要向全局控制寄存器 (GCR, 0x5000_0100) 中的寄存器 REGWRPROT 写 0x59, 0x16 和 0x88。这个过程用于保护 FLASH 内存免受在上电或断电期间无意识的写操作而造成的损坏。

在软件写 ISPGO 位后，要检查几个错误条件。如果错误条件产生，ISP 操作失败，ISP 失败标志置位，ISPFF 标志由 S/W 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。如果 ISPFF 被设置为 1，建议在每次 ISP 操作后，由软件检查 ISPFF 位并将其清零。

当 ISPGO 位置位，CPU 将等待 ISP 操作结束。在此期间，外设仍然像一样往常保持工作。如果任何中断请求发生，CPU 将不会响应中断直到 ISP 执行完成，ISPGO 位将会被硬件自动清除。用户通过检查 ISPGO 位可以知道 ISP 操作是否完成。用户需要添加 ISB 指令紧接着置位 ISPGO 的指令，从而保证紧跟 ISP 操作的指令正确执行。



注：NuMicro™ NUC100 系列允许用户通过 ISP 更新 CONFIG 的值。





ISP 模式	ISPCMD			ISPADR			ISPDAT
	FOEN	FCEN	FCTRL[3:0]	A21	A20	A[19:0]	
FLASH Page Erase	1	0	0010	0	A20	Address in A[19:0]	x
FLASH Program	1	0	0001	0	A20	Address in A[19:0]	Data in D[31:0]
FLASH Read	0	0	0000	0	A20	Address in A[19:0]	Data out D[31:0]
CONFIG Page Erase	1	0	0010	1	1	Address in A[19:0]	x
CONFIG Program	1	0	0001	1	1	Address in A[19:0]	Data in D[31:0]
CONFIG Read	0	0	0000	1	1	Address in A[19:0]	Data out D[31:0]

表 6-2 ISP 模式

6.9 Flash 控制寄存器映射

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

寄存器	偏移量	R/W	描述	复位后的值
Base Address (FMC_BA) : 0x5000_C000				
ISPCON	FMC_BA+0x00	R/W	ISP 控制寄存器	0x0000_0000
ISPADR	FMC_BA+0x04	R/W	ISP 地址寄存器	0x0000_0000
ISPDAT	FMC_BA+0x08	R/W	ISP 数据寄存器	0x0000_0000
ISPCMD	FMC_BA+0x0C	R/W	ISP 命令寄存器	0x0000_0000
ISPTRG	FMC_BA+0x10	R/W	ISP 触发寄存器	0x0000_0000
DFBADR	FMC_BA+0x14	R	数据 Flash 起始地址 (AP ROM 的大小小于 128KB)	0x0001_F000
DFBADR	FMC_BA+0x14	R	数据 Flash 起始地址 (AP ROM 的大小等于 128KB)	0x0000_0000
FATCON	FMC_BA+0x18	R/W	Flash 访问窗口控制寄存器	0x0000_0000



6.10 Flash 控制寄存器描述

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

Bits	描述	
[31:7]	Reserved	保留
[6]	ISPFF	ISP 失败标志 (写保护位) 当 ISP 满足下列条件时，该位由硬件置位： (1) APROM 写入本身（如果 APUEN = 0） (2) LDROM 写入本身 (3) 如果 CFGUEN 设置为 0，CONFIG 被擦除/编程 (4) 目的地址无效，比如超过正常范围 写 1 清除该位。
[5]	LDUEN	LDROM 更新使能 (写保护位) LDROM 更新使能位。 1 = 当芯片在 APROM 中运行时，LDROM 可以更新。 0 = LDROM 不能被更新
[4]	CFGUEN	使能由 ISP 更新配置-位 (写保护位) 1 = 使能 ISP 更新配置-位 0 = 禁用 ISP 更新配置-位
[3]	APUEN	APROM 更新使能 (写保护位) 1 = 当芯片运行在 APROM，APROM 可以更新 0 = 当芯片运行在 APROM，APROM 不能更新
[2]	Reserved	保留

[1]	BS	启动选择 (写保护位) 置位/清零该位选择下次是由 LDROM 启动还是由 APROM 启动，该位可作为MCU启动状态标志，用于检查芯片是由 LDROM 还是 APROM 启动。这一位在复位发生时（除了 CPU 复位 (RSTS_CPU 为 1) 或系统复位 (RSTS_SYS)）被初始化为 Config0 的 CBS 位的反转值，在其他复位时保持不变 1 = 从 LDROM 启动 0 = 从 APROM 启动
[0]	ISPEN	ISP 使能 (写保护位) ISP 功能使能位。设置该位使能 ISP 功能。 1 = 使能 ISP 功能 0 = 禁用 ISP 功能

**ISP 地址寄存器 (ISPADR)**

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

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

Bits	描述	
[31:0]	ISPADR	ISP 地址 NuMicro™ NUC100 系列最大内置 32Kx32 的 flash，仅支持字编程。执行 ISP 操作时，ISPADR[1:0] 必须保持 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	ISP 数据 ISP 操作之前，写数据到该寄存器 ISP 读操作后，可从该寄存器读数据

ISP 命令寄存器 (ISPCMD)

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

31	30	29	28	27	26	25	24	
Reserved								
23	22	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		FOEN	FCEN	FCTRL				

Bits	描述																												
[31:6]	Reserved	保留																											
[5]	FOEN	ISP 命令 ISP 命令表如下:																											
[4]	FCEN	操作模式 FOEN FCEN FCTRL[3:0]																											
[3:0]	FCTRL	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">读</td> <td style="padding: 2px;">0</td> </tr> <tr> <td style="padding: 2px;">编程</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">1</td> </tr> <tr> <td style="padding: 2px;">页擦除</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">0</td> <td style="padding: 2px;">1</td> <td style="padding: 2px;">0</td> </tr> </table>							读	0	0	0	0	0	0	编程	1	0	0	0	0	1	页擦除	1	0	0	0	1	0
读	0	0	0	0	0	0																							
编程	1	0	0	0	0	1																							
页擦除	1	0	0	0	1	0																							

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

Bits	描述	
[31:1]	Reserved	保留
[0]	ISPGO	<p>ISP 开始触发 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	偏移量	R/W	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	<p>数据 Flash 基地址</p> <p>该寄存器为数据 FLASH 开始地址寄存器。该寄存器只读。</p> <p>对于128 KB flash 内存的器件，数据 FLASH 的大小由用户配置定义，在芯片上电后寄存器的内容由 Config1 加载。但对于 64/32 KB的器件，地址固定为 0x0001_F000。</p>

Flash 访问时间控制寄存器 (FATCON)

寄存器	偏移量	R/W	描述	复位后的值
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
Reserved							
23	22	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			LFOM	FATS[2:0]			FPSEN

Bits	描述																			
[31:5]	Reserved	保留																		
[4]	LFOM	<p>低频优化模式 (写保护位) 当芯片操作频率低于 25 MHz时，通过设置该位为 1 芯片可以更高效的工作。 1 = 使能低频优化模式 0 = 禁用低频优化模式</p>																		
[3:1]	FATS	<p>Flash 访问时间窗口选择 (写保护位) 这些位用于决定 flash 读放大器持续有效时间。</p> <table border="1" style="margin-left: 20px;"> <tr> <th>FATS</th> <th>访问时间窗口 (ns)</th> </tr> <tr> <td>000</td> <td>40</td> </tr> <tr> <td>001</td> <td>50</td> </tr> <tr> <td>010</td> <td>60</td> </tr> <tr> <td>011</td> <td>70</td> </tr> <tr> <td>100</td> <td>80</td> </tr> <tr> <td>101</td> <td>90</td> </tr> <tr> <td>110</td> <td>100</td> </tr> <tr> <td>111</td> <td>Reserved</td> </tr> </table>	FATS	访问时间窗口 (ns)	000	40	001	50	010	60	011	70	100	80	101	90	110	100	111	Reserved
FATS	访问时间窗口 (ns)																			
000	40																			
001	50																			
010	60																			
011	70																			
100	80																			
101	90																			
110	100																			
111	Reserved																			
[0]	FPSEN	<p>Flash 省电使能 (写保护位) 如果CPU 时钟低于 24 MHz, S/W 可以使能省电功能。 1 = 使能 flash 省电功能 0 = 禁用 flash 省电功能</p>																		



新唐科技 NUVOTON
INTELLECTUAL PROPERTY

新唐科技 NUVOTON
INTELLECTUAL PROPERTY

7 电气特性

7.1 绝对最大额定值

参数	符号	最小值	最大值	单位
直流电源电压	VDD-VSS	-0.3	+7.0	V
输入电压	VIN	VSS-0.3	VDD+0.3	V
振荡器频率	1/t _{CLCL}	4	24	MHz
工作温度	TA	-40	+85	°C
贮存温度	TST	-55	+150	°C
VDD 最大流入电流		-	120	mA
VSS 最大流出电流			120	mA
单一 I/O 管脚最大灌电流			35	mA
单一 I/O 管脚最大拉电流			35	mA
所有 I/O 管脚最大灌电流总和			100	mA
所有 I/O 管脚最大拉电流总和			100	mA

注：上表所列的条件中，其极限值可能对器件的提升和稳定有反作用。

7.2 DC 电气特性

7.2.1 NuMicro™ NUC130/NUC140 DC 电气特性

(VDD-VSS=3.3 V, TA = 25°C, FOSC = 50 MHz 除非其他特别说明)

参数	符号	规格				测试条件
		最小值	典型值	最大值	单位	
工作电压	V _{DD}	2.5		5.5	V	V _{DD} = 2.5 V ~ 5.5 V up to 50 MHz
电源地	V _{SS} AV _{SS}	-0.3			V	
LDO 输出电压	V _{LDO}	-10%	2.5	+10%	V	V _{DD} > 2.7 V
模拟工作电压	AV _{DD}	0		V _{DD}	V	
模拟参考电压	V _{ref}	0		AV _{DD}	V	
普通模式下的工作电流 @ 50 MHz	I _{DD1}		51		mA	V _{DD} = 5.5 V @ 50 MHz, enable all IP and PLL, XTAL=12 MHz
	I _{DD2}		25		mA	V _{DD} = 5.5 V @ 50 MHz, disable all IP and enable PLL, XTAL=12 MHz
	I _{DD3}		48		mA	V _{DD} = 3 V @ 50 MHz, enable all IP and PLL, XTAL=12 MHz
	I _{DD4}		23		mA	V _{DD} = 3 V @ 50 MHz, disable all IP and enable PLL, XTAL=12 MHz
普通模式下的工作电流 @ 12 MHz	I _{DD5}		19		mA	V _{DD} = 5.5 V @ 12 MHz, enable all IP and disable PLL, XTAL=12 MHz
	I _{DD6}		7		mA	V _{DD} = 5.5 V @ 12 MHz, disable all IP and disable PLL, XTAL=12 MHz
	I _{DD7}		17		mA	V _{DD} = 3 V @ 12 MHz, enable all IP and disable PLL, XTAL=12 MHz

参数	符号	规格				测试条件
		最小值	典型值	最大值	单位	
普通模式下的工作电流 @ 4 MHz	I _{DD8}		6		mA	V _{DD} = 3 V@12 MHz, disable all IP and disable PLL, XTAL=12 MHz
	I _{DD9}		11		mA	V _{DD} = 5 V@4 MHz, enable all IP and disable PLL, XTAL=4 MHz
	I _{DD10}		3		mA	V _{DD} = 5 V@4 MHz, disable all IP and disable PLL, XTAL=4 MHz
	I _{DD11}		10		mA	V _{DD} = 3 V@4 MHz, enable all IP and disable PLL, XTAL=4 MHz
空闲模式下的工作电流 @ 50 MHz	I _{IDLE1}		2.5		mA	V _{DD} = 3 V@4 MHz, disable all IP and disable PLL, XTAL=4 MHz
	I _{IDLE2}		35		mA	V _{DD} = 5.5 V@50 MHz, enable all IP and PLL, XTAL=12 MHz
	I _{IDLE3}		15		mA	V _{DD} = 5.5 V@50 MHz, disable all IP and enable PLL, XTAL=12 MHz
	I _{IDLE4}		33		mA	V _{DD} = 3 V@50 MHz, enable all IP and PLL, XTAL=12 MHz
空闲模式下的工作电流 @ 12 MHz	I _{IDLE5}		10		mA	V _{DD} = 5.5 V@12 MHz, enable all IP and disable PLL, XTAL=12 MHz
	I _{IDLE6}		4.5		mA	V _{DD} = 5.5 V@12 MHz, disable all IP and disable PLL, XTAL=12 MHz
	I _{IDLE7}		9		mA	V _{DD} = 3 V@12 MHz, enable all IP and disable PLL, XTAL=12 MHz

参数	符号	规格				测试条件
		最小值	典型值	最大值	单位	
空闲模式下的工作电流 @ 4 MHz	I _{IDLE8}		3.5		mA	V _{DD} = 3 V @ 12 MHz, disable all IP and disable PLL, XTAL=12 MHz
	I _{IDLE9}		4		mA	V _{DD} = 5 V @ 4 MHz, enable all IP and disable PLL, XTAL=4 MHz
	I _{IDLE10}		2.5		mA	V _{DD} = 5 V @ 4 MHz, disable all IP and disable PLL, XTAL=4 MHz
	I _{IDLE11}		3.5		mA	V _{DD} = 3 V @ 4 MHz, enable all IP and disable PLL, XTAL=4 MHz
掉电模式下的工作电流	I _{PWD1}		12		μA	V _{DD} = 5.5 V, RTC OFF, No load @ Disable BOV function
	I _{PWD2}		9		μA	V _{DD} = 3.3 V, RTC OFF, No load @ Disable BOV function
	I _{PWD3}				μA	V _{DD} = 5.5 V, RTC run , No load @ Disable BOV function
	I _{PWD4}				μA	V _{DD} = 3.3 V, RTC run , No load @ Disable BOV function
PA, PB, PC, PD, PE 输入电流 (准双向模式)	I _{IN1}		-50	-60	μA	V _{DD} = 5.5 V, V _{IN} = 0 V or V _{IN} =V _{DD}
/RESET ^[1] 管脚输入电流	I _{IN2}	-55	-45	-30	μA	V _{DD} = 3.3 V, V _{IN} = 0.45 V
PA, PB, PC, PD, PE 输入漏电流	I _{LK}	-2	-	+2	μA	V _{DD} = 5.5 V, 0 < V _{IN} < V _{DD}
PA~PE 逻辑 1至 0 转换时的电流 (准双向模式)	I _{TL} ^[3]	-650	-	-200	μA	V _{DD} = 5.5 V, V _{IN} <2.0 V
PA, PB, PC, PD, PE 输入低电压 (TTL 输入)	V _{IL1}	-0.3	-	0.8	V	V _{DD} = 4.5 V
		-0.3	-	0.6		V _{DD} = 2.5 V
PA, PB, PC, PD, PE 输入高电压 (TTL 输入)	V _{IH1}	2.0	-	V _{DD} +0.2	V	V _{DD} = 5.5 V
		1.5	-	V _{DD} +0.2		V _{DD} = 3.0 V
PA, PB, PC, PD, PE 输入低电压 (Schmitt 输入)	V _{IL2}	-0.5	-	0.4 V _{DD}	V	

参数	符号	规格				测试条件
		最小值	典型值	最大值	单位	
PA, PB, PC, PD, PE 输入高电压 (Schmitt 输入)	V_{IH2}	0.6 V_{DD}	-	$V_{DD}+0.5$	V	
PA~PE 迟滞电压 (Schmitt 输入)	V_{HY}		0.2 V_{DD}		V	
XT1 ^[*2] 管脚输入低电压	V_{IL3}	0	-	0.8	V	$V_{DD} = 4.5\text{ V}$
		0	-	0.4		$V_{DD} = 3.0\text{ V}$
XT1 ^[*2] 管脚输入高电压	V_{IH3}	3.5	-	$V_{DD}+0.2$	V	$V_{DD} = 5.5\text{ V}$
		2.4	-	$V_{DD}+0.2$		$V_{DD} = 3.0\text{ V}$
X32I ^[*2] 管脚输入低电压	V_{IL4}	0	-	0.4	V	
X32I ^[*2] 管脚输入高电压	V_{IH4}	1.7		2.5	V	
/RESET 管脚负向阈值电压 (Schmitt 输入)	V_{ILS}	-0.5	-	0.3 V_{DD}	V	
/RESET 管脚正向阈值电压 (Schmitt 输入)	V_{IHS}	0.7 V_{DD}	-	$V_{DD}+0.5$	V	
PA, PB, PC, PD, PE 拉电流 (准双向模式)	I_{SR11}	-300	-370	-450	μA	$V_{DD} = 4.5\text{ V}, V_S = 2.4\text{ V}$
	I_{SR12}	-50	-70	-90	μA	$V_{DD} = 2.7\text{ V}, V_S = 2.2\text{ V}$
	I_{SR12}	-40	-60	-80	μA	$V_{DD} = 2.5\text{ V}, V_S = 2.0\text{ V}$
PA, PB, PC, PD, PE 拉电流 (推挽模式)	I_{SR21}	-20	-24	-28	mA	$V_{DD} = 4.5\text{ V}, V_S = 2.4\text{ V}$
	I_{SR22}	-4	-6	-8	mA	$V_{DD} = 2.7\text{ V}, V_S = 2.2\text{ V}$
	I_{SR22}	-3	-5	-7	mA	$V_{DD} = 2.5\text{ V}, V_S = 2.0\text{ V}$
PA, PB, PC, PD, PE 灌电流 (准双向和推挽模式)	I_{SK1}	10	16	20	mA	$V_{DD} = 4.5\text{ V}, V_S = 0.45\text{ V}$
	I_{SK1}	7	10	13	mA	$V_{DD} = 2.7\text{ V}, V_S = 0.45\text{ V}$
	I_{SK1}	6	9	12	mA	$V_{DD} = 2.5\text{ V}, V_S = 0.45\text{ V}$
BOV_VL [1:0] =00b 的欠压电压	$V_{BO2.2}$	2.1	2.2	2.3	V	
BOV_VL [1:0] =01b 的欠压电压	$V_{BO2.7}$	2.6	2.7	2.8	V	
BOV_VL [1:0] =10b 的欠压电压	$V_{BO3.8}$	3.6	3.8	4.0	V	
BOV_VL [1:0] =11b 的欠压电压	$V_{BO4.5}$	4.3	4.5	4.7	V	
BOD 电压的迟滞范围	V_{BH}	30	-	150	mV	$V_{DD} = 2.5\text{ V}\sim 5.5\text{ V}$
带隙电压	V_{BG}	1.20	1.26	1.32	V	$V_{DD} = 2.5\text{ V}\sim 5.5\text{ V}$

注:

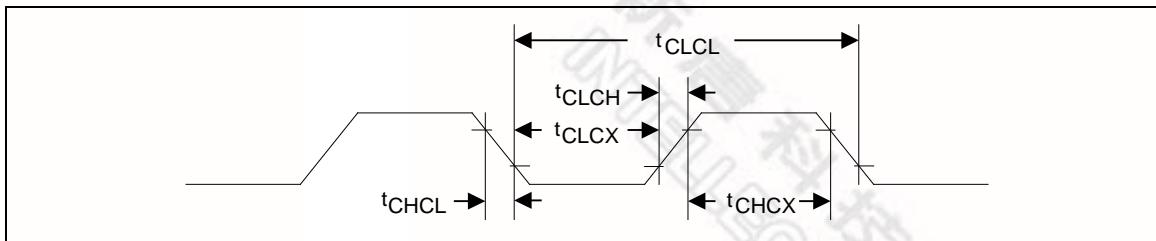
1. /RESET 管脚为 Schmitt 触发输入。



2. 晶振输入为 CMOS 输入。
3. 当 PA, PB, PC, PD 和 PE 管脚被外部由 1 驱动到 0 时, 可以作为转换电流的源。在 $V_{DD} = 5.5$ V 条件下, 当 V_{IN} 接近 2 V, 转换电流达到它的最大值。

7.3 AC 电气特性

7.3.1 外部 4~24 MHz 高速振荡器



注：占空比为 50%.

符号	参数	条件	最小值	典型值	最大值	单位
t_{CHCX}	时钟高电平时间		20	-	-	nS
t_{CLCX}	时钟低电平时间		20	-	-	nS
t_{CLCH}	时钟上升沿时间		-	-	10	nS
t_{CHCL}	时钟下降沿时间		-	-	10	nS

7.3.2 外部 4~24 MHz 高速晶振

参数	条件	最小值	典型值	最大值	单位
输入时钟频率	外部晶振	4	12	24	MHz
温度	-	-40	-	85	°C
VDD	-	2.5	5	5.5	V

7.3.2.1 典型晶振应用电路

晶振	C1	C2	R
4 MHz ~ 24 MHz	不需要	不需要	不需要

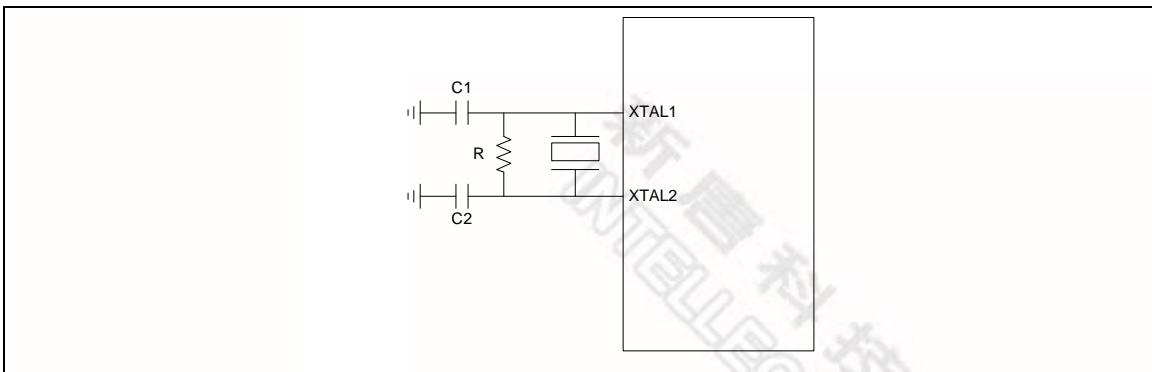


图 7-1 典型晶振应用电路



7.3.3 外部 32.768 kHz 低速晶振

参数	条件	最小值	典型值	最大值	单位
输入时钟频率	外部晶振	-	32.768	-	kHz
温度	-	-40	-	85	°C
VDD	-	2.5	-	5.5	V

7.3.4 内部 22.1184 MHz 高速振荡器

参数	条件	最小值	典型值	最大值	单位
工作电压 ^[1]	-	2.5	-	5.5	V
中心频率	-	-	22.1184	-	MHz
校验内部振荡器频率	+25°C; V _{DD} =5 V	-1	-	+1	%
	-40°C~+85°C; V _{DD} =2.5 V~5.5 V	-3	-	+3	%
工作电流	V _{DD} =5 V	-	500	-	uA

7.3.5 内部 10 kHz 低速振荡器

参数	条件	最小值	典型值	最大值	单位
工作电压 ^[1]	-	2.5	-	5.5	V
中心频率	-	-	10	-	kHz
校验内部振荡器频率	+25°C; V _{DD} =5 V	-30	-	+30	%
	-40°C~+85°C; V _{DD} =2.5 V~5.5 V	-50	-	+50	%

注：内部工作电压来自 LDO。

7.4 模拟量特性

7.4.1 12-位 SARADC 规格

符号	参数	最小值	典型值	最大值	单位
-	分辨率	-	-	12	Bit
DNL	非线性差分误差	-	±3	-	LSB
INL	非线性积分误差	-	±4	-	LSB
EO	补偿误差	-	±1	10	LSB
EG	增益误差 (传输增益)	-	1	1.005	-
-	一致性	Guaranteed			
FADC	ADC 时钟频率	-	-	16	MHz
TCAL	校准时间	-	127	-	Clock
TS	采样时间	-	7	-	Clock
TADC	转换时间	-	13	-	Clock
FS	采样率	-	-	600	K SPS
VLDO	工作电压	-	2.5	-	V
VADD		3	-	5.5	V
IDD	工作电流 (平均)	-	0.5	-	mA
IDDA		-	1.5	-	mA
VREF	参考电压	-	VDDA	-	V
IREFP	参考电流 (平均)	-	1	-	mA
VIN	参考电压	0	-	VREF	V
CIN	电容	-	5	-	pF



7.4.2 LDO 规格和电源管理

参数	最小值.	典型值	最大值	单位	备注
输入电压	2.7	5	5.5	V	V_{DD} 输入电压
输出电压	-10%	2.5	+10%	V	$V_{DD} > 2.7\text{ V}$
温度	-40	25	85	°C	
静态电流 (PD=0)	-	100	-	uA	
静态电流 (PD=1)	-	5	-	uA	
Iload (PD=0)	-	-	100	mA	
Iload (PD=1)	-	-	100	uA	
Cbp	-	10	-	uF	Resr=1ohm

注:

1. 建议接一颗 10uF 或更大的电容和一颗 100nF 旁路电容在 VDD 与 VSS 之间。
2. 为保证电源稳定, 要在 LDO 与 VSS 之间接一颗 10uF 或更大的电容。



7.4.3 低压复位说明

参数	条件	最小值	典型值	最大值	单位
工作电压	-	1.7	-	5.5	V
静态电流	VDD5V=5.5 V	-	-	5	μA
温度	-	-40	25	85	°C
阈值电压	温度=25°	1.7	2.0	2.3	V
	温度=-40°	-	2.4	-	V
	温度=85°	-	1.6	-	V
迟滞	-	0	0	0	V

7.4.4 欠压检测说明

参数	条件	最小值	典型值	最大值	单位
工作电压	-	2.5	-	5.5	V
静态电流	AVDD=5.5 V	-	-	125	μA
温度	-	-40	25	85	°C
欠压电压	BOV_VL[1:0]=11	4.3	4.5	4.7	V
	BOV_VL [1:0]=10	3.6	3.8	4.0	V
	BOV_VL [1:0]=01	2.6	2.7	2.8	V
	BOV_VL [1:0]=00	2.1	2.2	2.3	V
迟滞	-	30	-	150	mV

7.4.5 上电复位说明 (5 V)

参数	条件	最小值	典型值	最大值	单位
温度	-	-40	25	85	°C
复位电压	V+	-	2	-	V
静态电流	Vin>复位电压	-	1	-	nA



7.4.6 温度传感器说明

参数	条件	最小值.	典型值	最大值	单位
电源 ^[1]		2.5	-	5.5	V
温度		-40	-	125	°C
电流消耗		6.4	-	10.5	uA
增益			-1.76		mV/°C
偏移量	Temp=0 °C		720		mV

注：内部工作电压来自 LDO。

7.4.7 比较器说明

参数	条件	最小值	典型值	最大值	单位
温度	-	-40	25	85	°C
VDD	-	2.4	3	5.5	V
VDD 电流	20 uA@VDD=3 V	-	20	40	uA
输入偏移电压	-	-	5	15	mV
输出漂移（偏差）	-	0.1	-	VDD-0.1	V
共模输入范围	-	0.1	-	VDD-1.2	V
DC 增益	-	-	70	-	dB
延迟	@VCM=1.2 V and VDIFF=0.1 V	-	200	-	ns
比较电压	20 mV@VCM=1 V 50 mV@VCM=0.1 V 50 mV@VCM=VDD-1.2 @10 mV for non-hysteresis	10	20	-	mV
迟滞	One bit control W/O and W. hysteresis @VCM=0.4 V ~ VDD-1.2 V	-	±10	-	mV
唤醒时间	@CINP=1.3 V CINN=1.2 V	-	-	2	us



7.4.8 USB PHY 说明

7.4.8.1 USB DC 电气特性

符号	参数	条件	最小值	典型值	最大值	单位
V_{IH}	输入高 (driven)		2.0			V
V_{IL}	输入低				0.8	V
V_{DI}	差分输入	$ PADP-PADM $	0.2			V
V_{CM}	差分同模范围	Includes V_{DI} range	0.8		2.5	V
V_{SE}	单端接收器阈值		0.8		2.0	V
	接收器迟滞			200		mV
V_{OL}	输出低 (driven)		0		0.3	V
V_{OH}	输出高 (driven)		2.8		3.6	V
V_{CRS}	输出信号串扰电压		1.3		2.0	V
R_{PU}	上拉电阻		1.425		1.575	kΩ
R_{PD}	下拉电阻		14.25		15.75	kΩ
V_{TRM}	上行端口上的上拉电阻的极限电压(RPU)		3.0		3.6	V
Z_{DRV}	驱动输出阻抗	稳态驱动*		10		Ω
C_{IN}	发射器电容	Pin to GND			20	pF

*驱动输出阻抗不包括串联电阻阻抗

7.4.8.2 USB 全速驱动器电气特性

符号	参数	条件	最小值	典型值	最大值	单位
T_{FR}	上升时间	$C_L=50p$	4		20	ns
T_{FF}	下降时间	$C_L=50p$	4		20	ns
T_{FRFF}	上升和下降时间比值	$T_{FRFF}=T_{FR}/T_{FF}$	90		111.11	%

7.4.8.3 USB 电源功耗

符号	参数	条件	最小值	典型值	最大值	单位
I_{VDDREG} (全速)	VDDD 和 VDDREG 供给电流 (稳态)	待机		50		uA
		输入模式				uA
		输出模式				uA



新唐科技 NUVOTON
INTELLECTUAL PROPERTY

新唐科技 NUVOTON
INTELLECTUAL PROPERTY

7.5 SPI 动态特性

符号	参数	最小值	典型值	最大值	单位
SPI 主机模式 (VDD = 4.5V ~ 5.5V, 30pF 负载电容)					
t _{DS}	数据准备时间	4	2	-	ns
t _{DH}	数据保持时间	0	-	-	ns
t _V	数据输出有效时间	-	7	11	ns
SPI 主机模式 (VDD = 3.0V ~ 3.6V, 30pF 负载电容)					
t _{DS}	数据准备时间	5	3	-	ns
t _{DH}	数据保持时间	0	-	-	ns
t _V	数据输出有效时间	-	13	18	ns
SPI 从机模式 (VDD = 4.5V ~ 5.5V, 30pF 负载电容)					
t _{DS}	数据准备时间	0	-	-	ns
t _{DH}	数据保持时间	2*PCLK+4	-	-	ns
t _V	数据输出有效时间	-	2*PCLK+11	2*PCLK+19	ns
SPI 从机模式 (VDD = 3.0V ~ 3.6V, 30pF 负载电容)					
t _{DS}	数据准备时间	0	-	-	ns
t _{DH}	数据保持时间	2*PCLK+6	-	-	ns
t _V	数据输出有效时间	-	2*PCLK+19	2*PCLK+25	ns

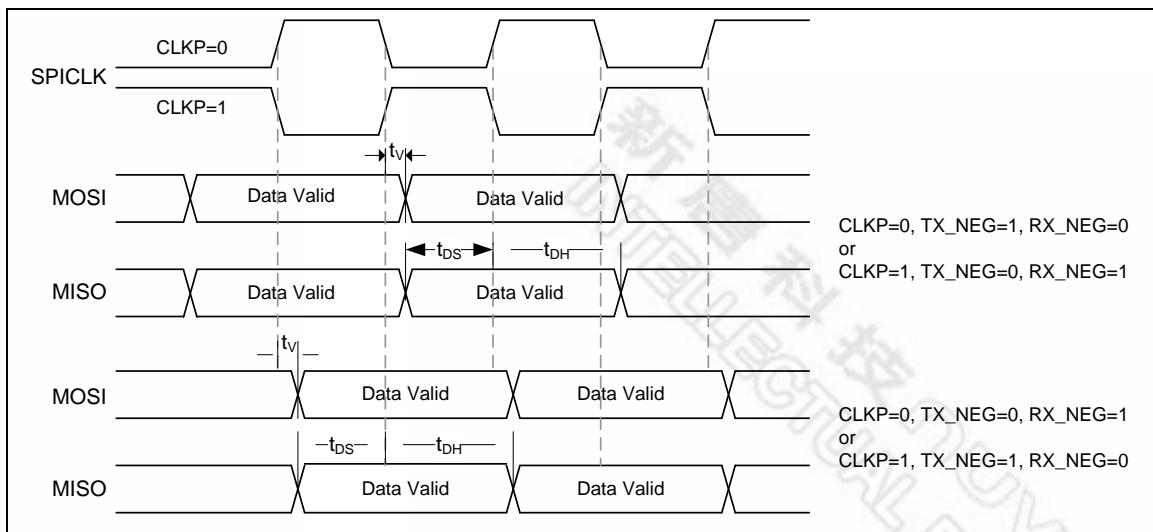


图 7-2 SPI 主机动态特性时序图

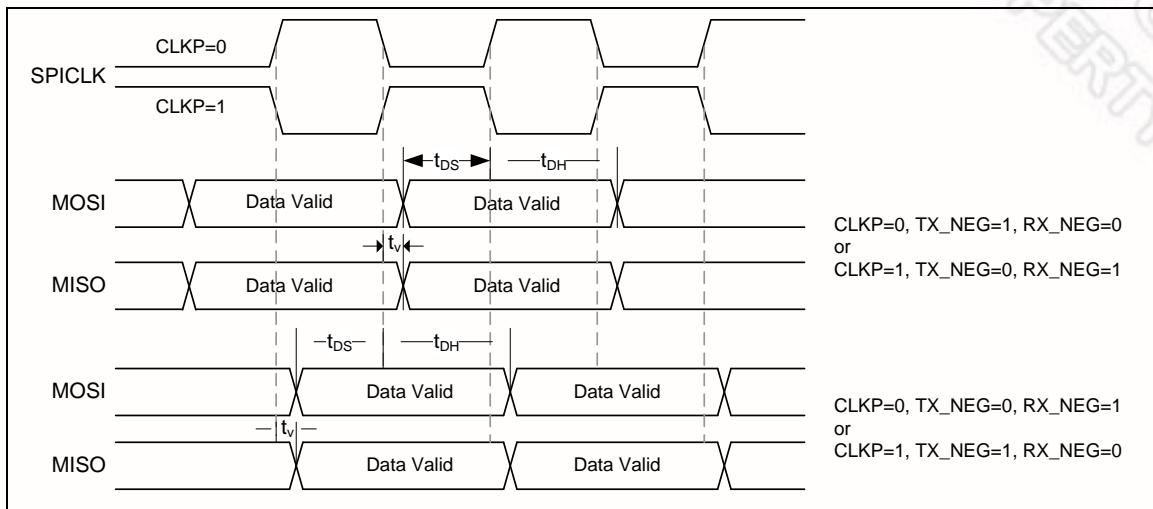
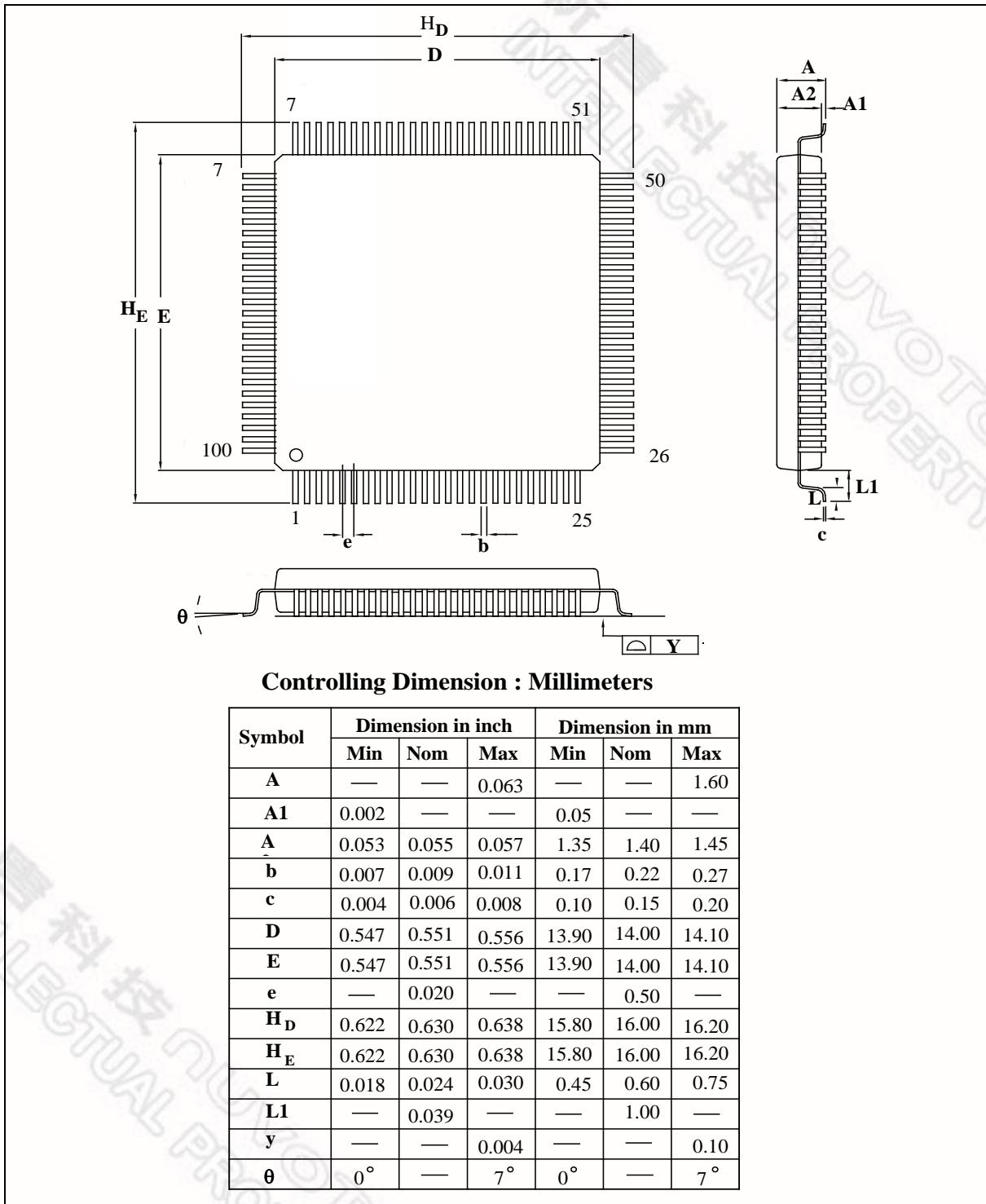


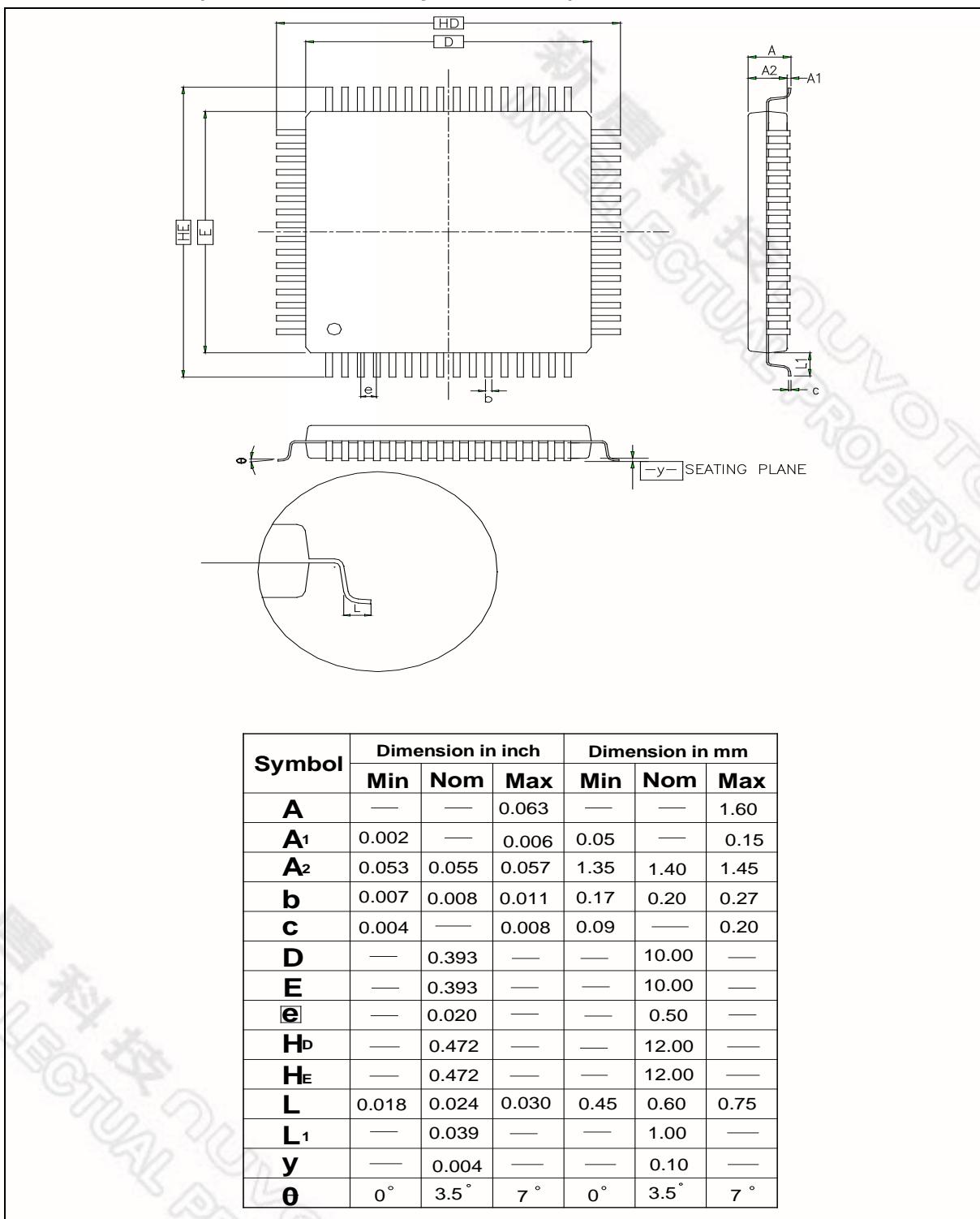
图 7-3 SPI 从机动态特性时序图

8 封装定义

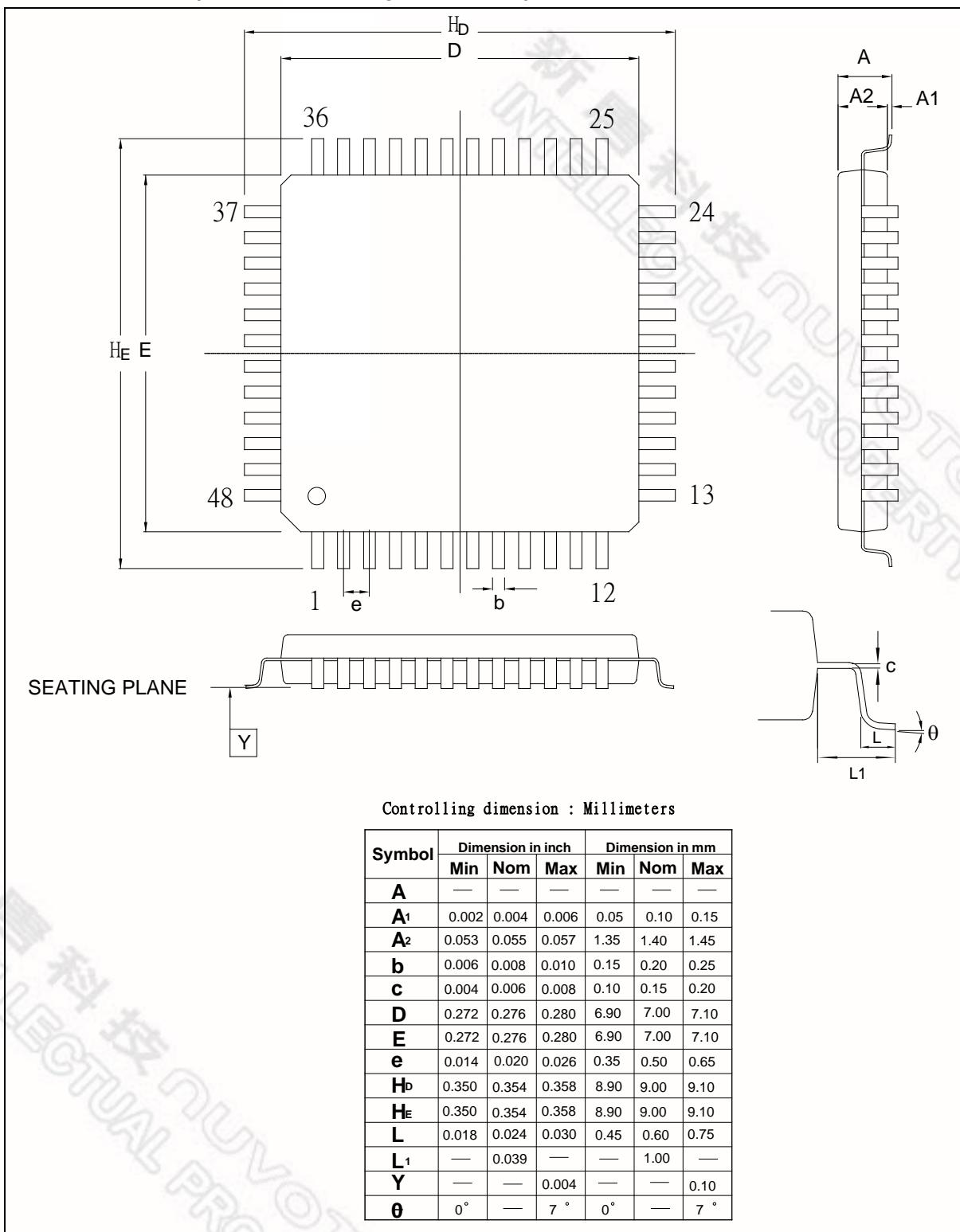
8.1 100L LQFP (14x14x1.4 mm footprint 2.0mm)



8.2 64L LQFP (10x10x1.4mm footprint 2.0 mm)



8.3 48L LQFP (7x7x1.4mm footprint 2.0mm)





9 版本历史

版本	日期	描述
V2.00	2011年 5月 4 日,	1. 该文档从“NUC100 series TRM.doc”分出来 2. 更新 CAN 到 BOSCH C-CAN
V2.01	2011年 6月 14 日	1. 修订 ISPCON.BS 和 ISPCON.ISPFF 的描述 2. 修改温度传感器规格 3. 修改复用管脚 T2EX, T3EX, nRD, nWR 的管脚表格位置 4. 更新 SPI 动态特性的标题 5. 更新 BOD 规格
V2.02	2014年 7月 3日	1. 更新格式 2. 修改UART_CLK / (A+2), A must >=7 3. 更新ET and PT 暂存器 4. 移除SPI FIFO 模式

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.